Feb. 6, 1968 R. D. HUNTER ETAL 3,368,205

CONTROL APPARATUS IN A DATA PROCESSING SYSTEM.

Filed April 14, 1965 3 Sheets-Sheet 1

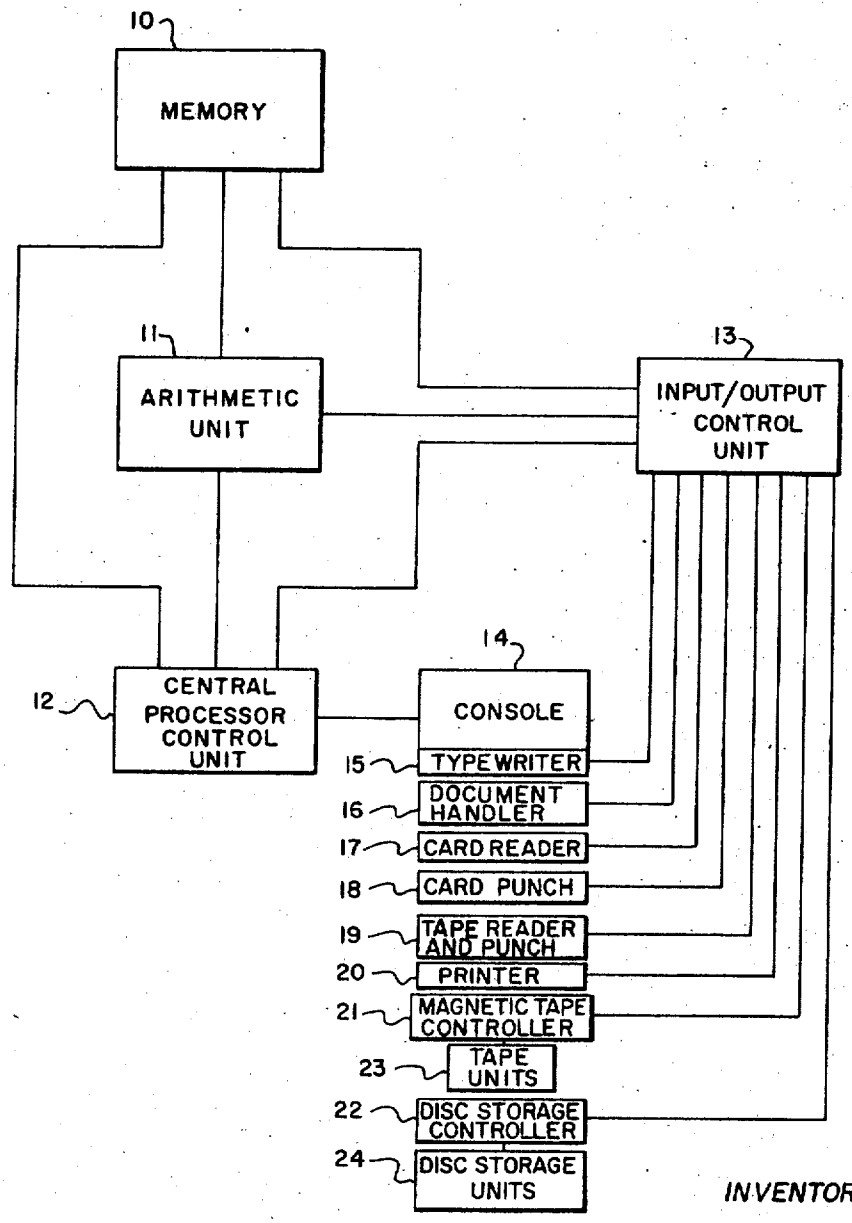


FIG. 1

DATA PROCESSING SYSTEM

INVENTORS.

ROBERT D. HUNTER
ROBERT A. PERRINE
JOHN E. WILHITE

BY

ATTORNEY

FIG. 2a

ALPHANUMERIC DATA WORD

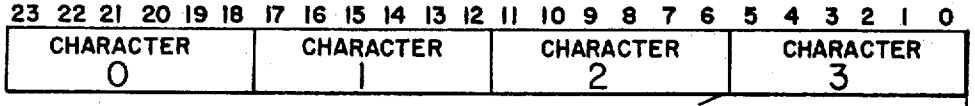
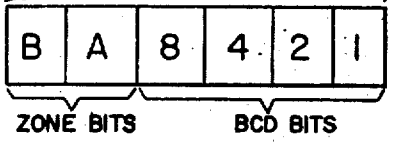


FIG. 2b

BINARY DATA WORD



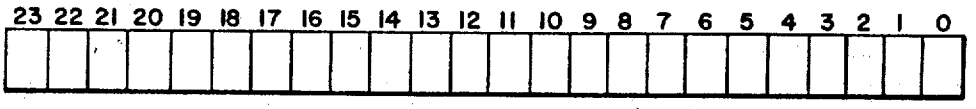


FIG. 2c

INSTRUCTION WORD

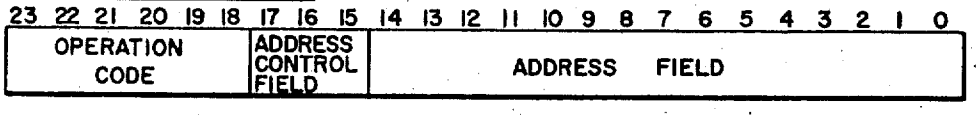


FIG. 2d

FIXED LOCATION INDEX WORD (WHEN USED FOR ADDRESS MODIFICATION)

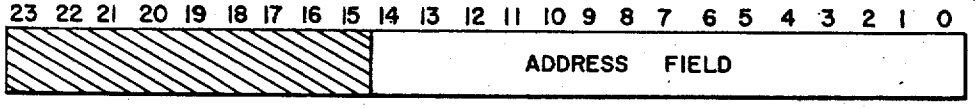


FIG. 2e

P-SEQUENCE AMS WORD

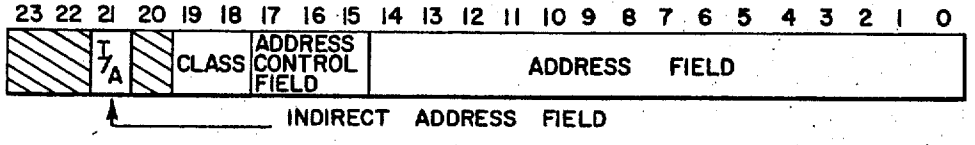


FIG.2f

REMOTE AMS WORD

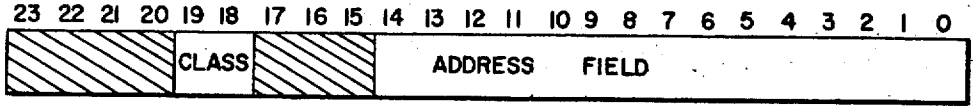


FIG.2g

INDIRECT ADDRESS WORD

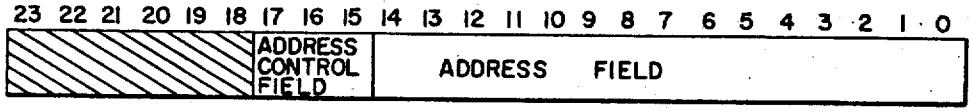


FIG.2h

INDIRECT AMS WORD

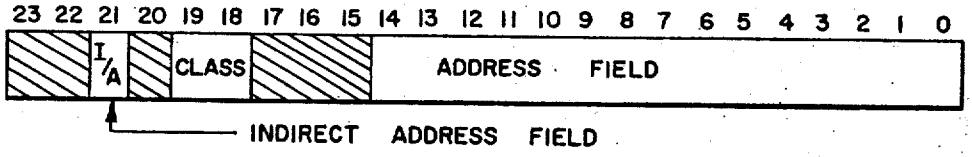


FIG.2i

P SEQUENCE SAS WORD

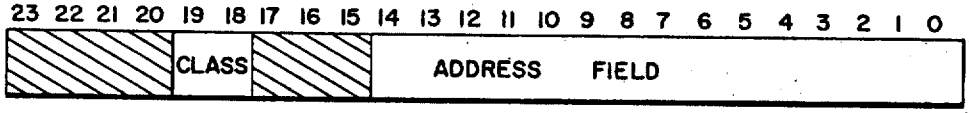


FIG.2j

REMOTE SAS WORD

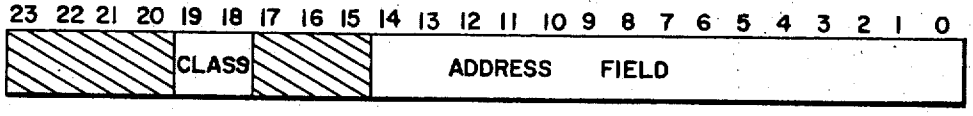
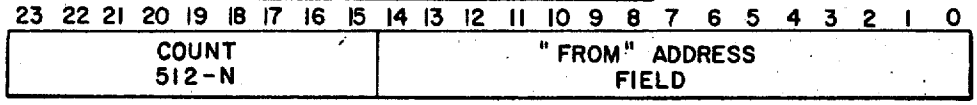


FIG. 2k

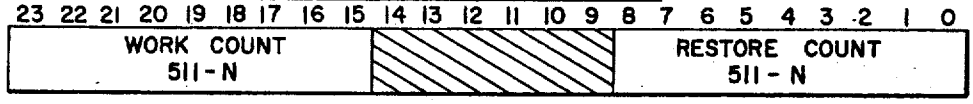
CONTROL WORD – INSTRUCTION 06(MOV)



N = NUMBER OF WORDS TO BE MOVED

FIG. 2l

CONTROL WORD – INSTRUCTION 16(BRC)



N = NUMBER OF TIMES BRANCH IS TAKEN

FIG. 2m

STATUS WORD – INSTRUCTION 67(CPO)

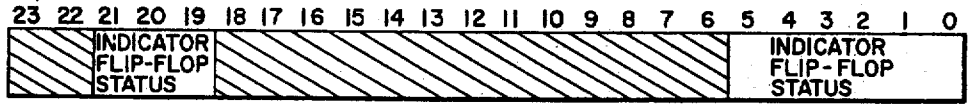
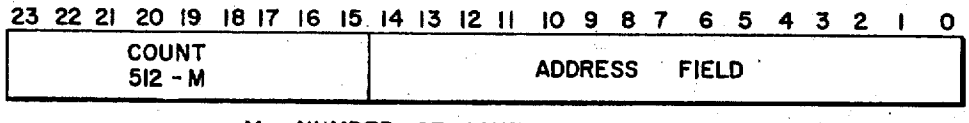


FIG. 2n

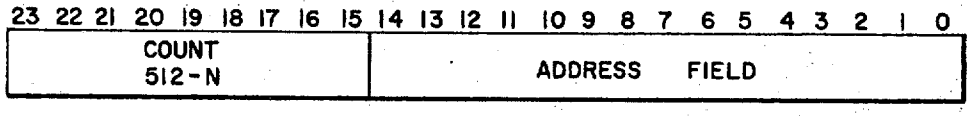
LIST POINTER WORD (LPW)



M = NUMBER OF CONTROL WORDS IN DATA CONTROL LIST.

FIG. 2o

DATA CONTROL WORD (DCW)



N = NUMBER OF CHARACTERS TO BE TRANSFERED BETWEEN MEMORY AND PERIPHERAL SUBSYSTEM

Feb. 6, 1968 R. D. HUNTER ETAL 3,368,205

CONTROL APPARATUS IN A DATA PROCESSING SYSTEM

Filed April 14, 1965 36 Sheets-Sheet 5

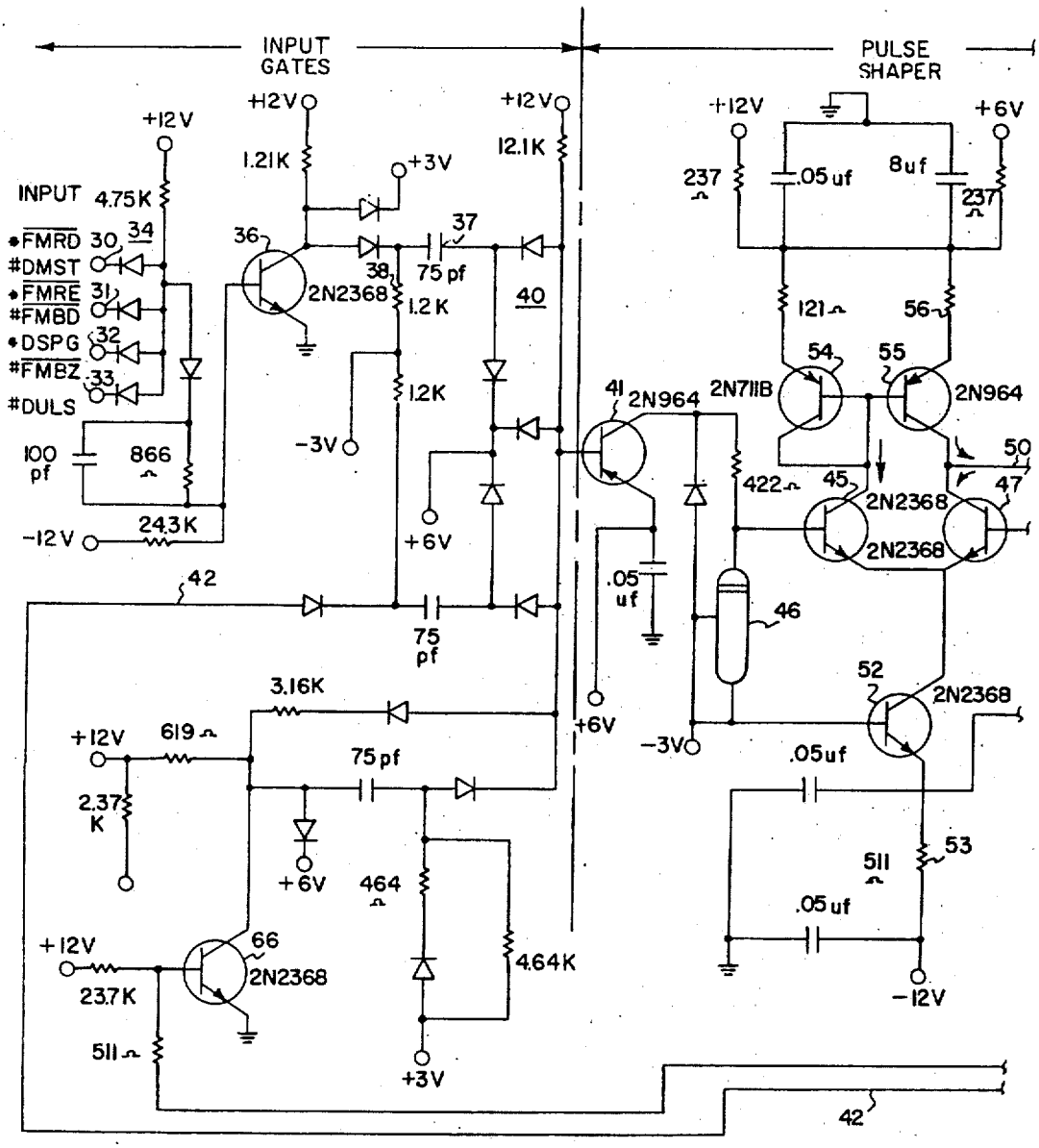


FIG. 3a

CLOCK GENERATOR

Feb. 6, 1968 R. D. HUNTER ETAL 3,368,205
CONTROL APPARATUS IN A DATA PROCESSING SYSTEM
Filed April 14, 1965 363 Sheets-Sheet 6

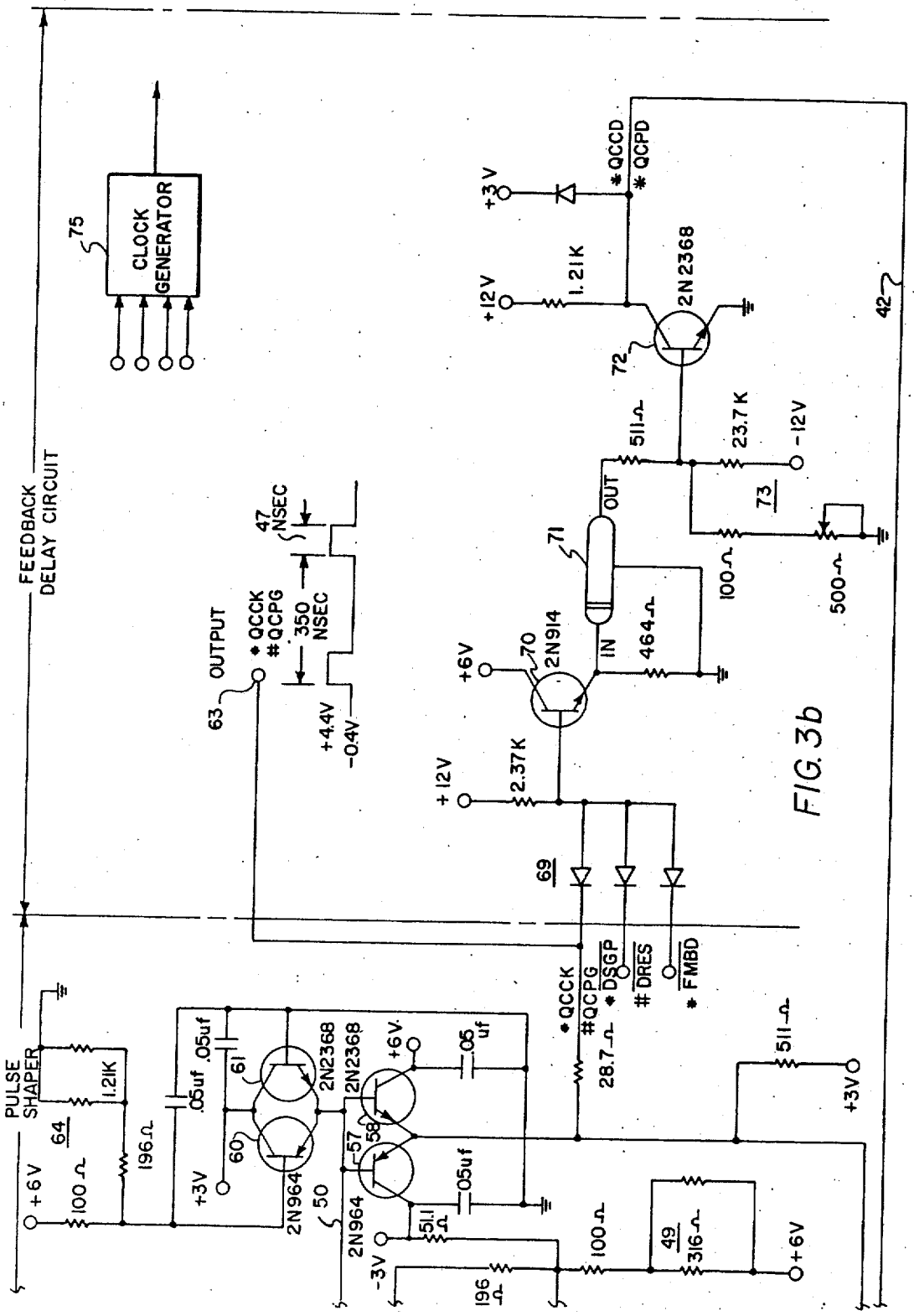


FIG. 3b

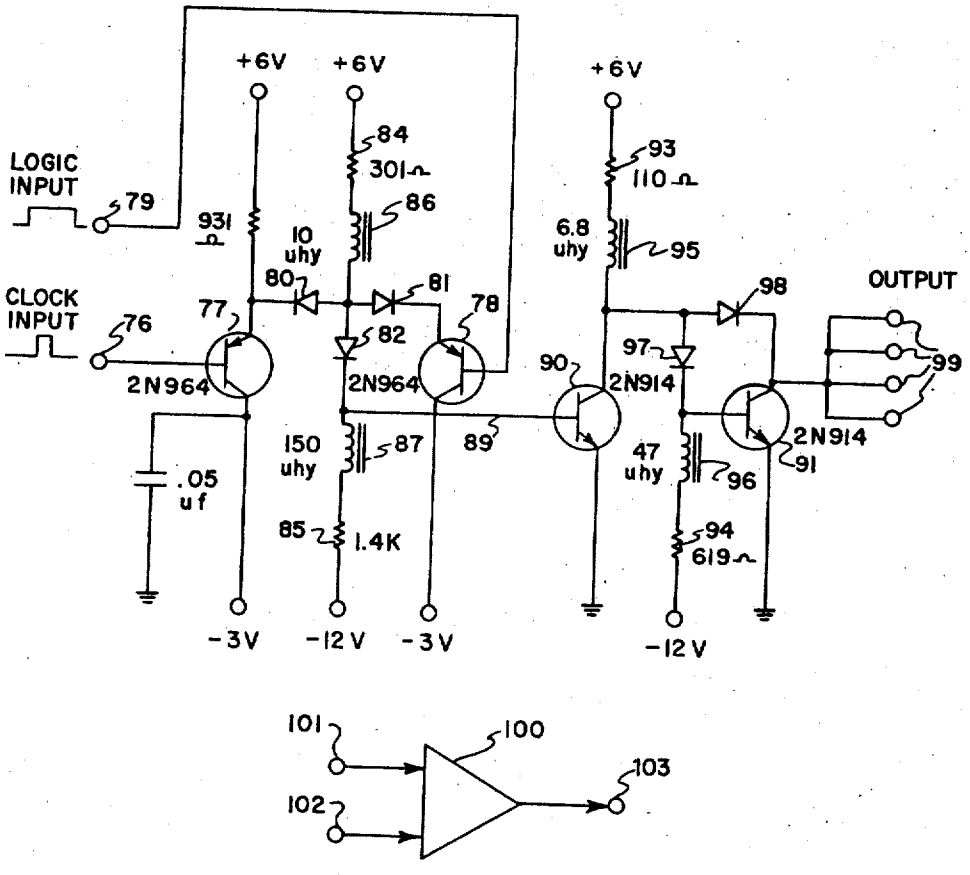


FIG. 4
GATED CLOCK
DISTRIBUTION DRIVER

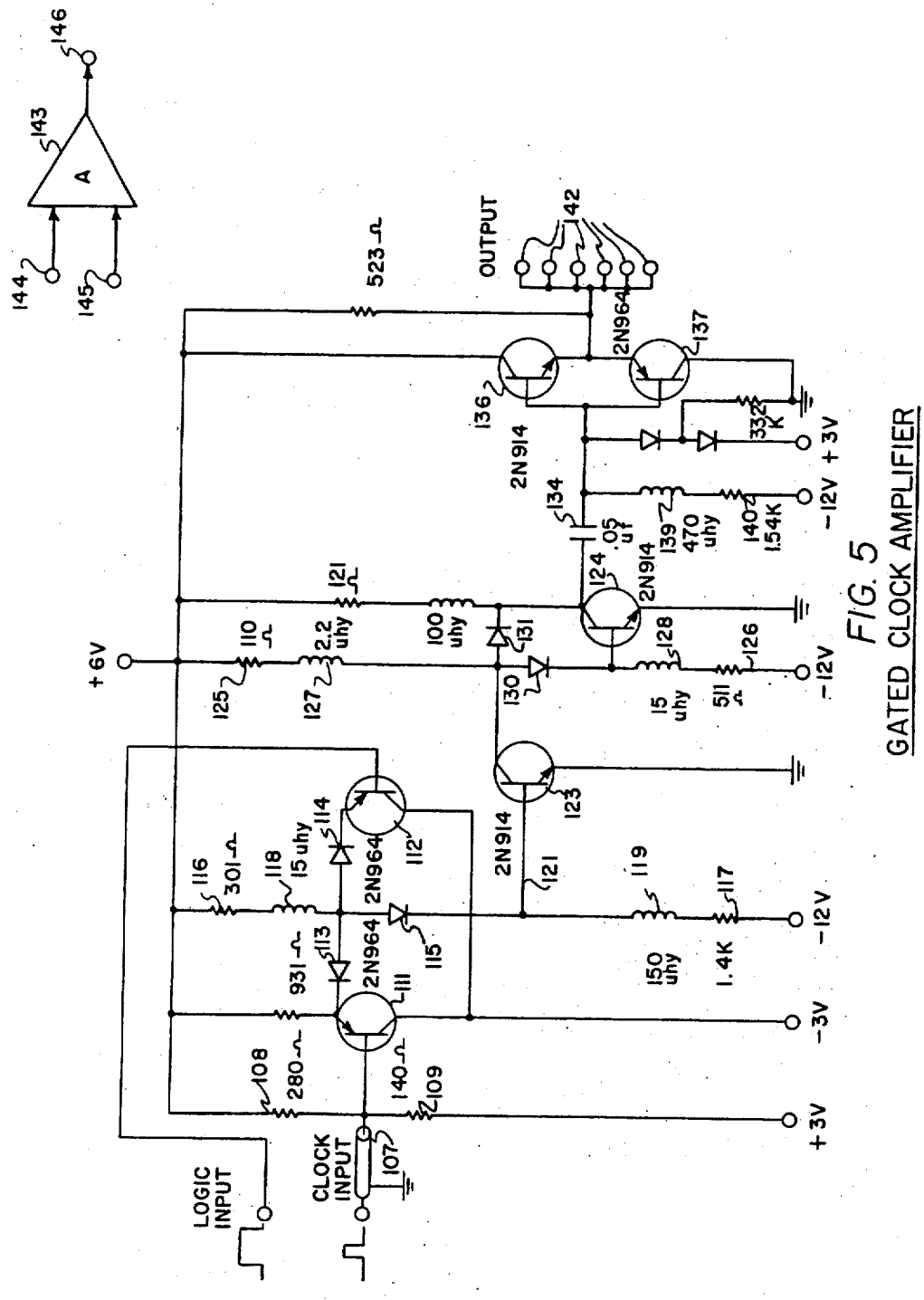


FIG. 5
GATED CLOCK AMPLIFIER

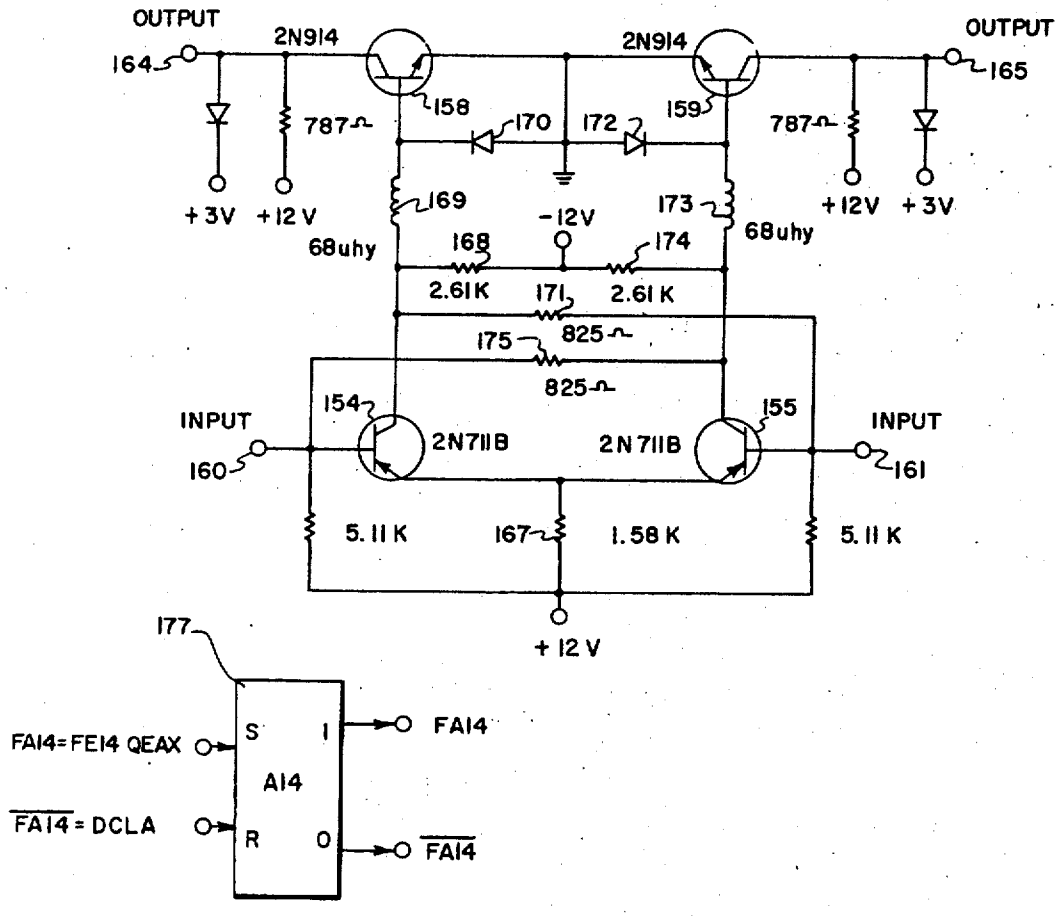


FIG. 6
FLIP-FLOP

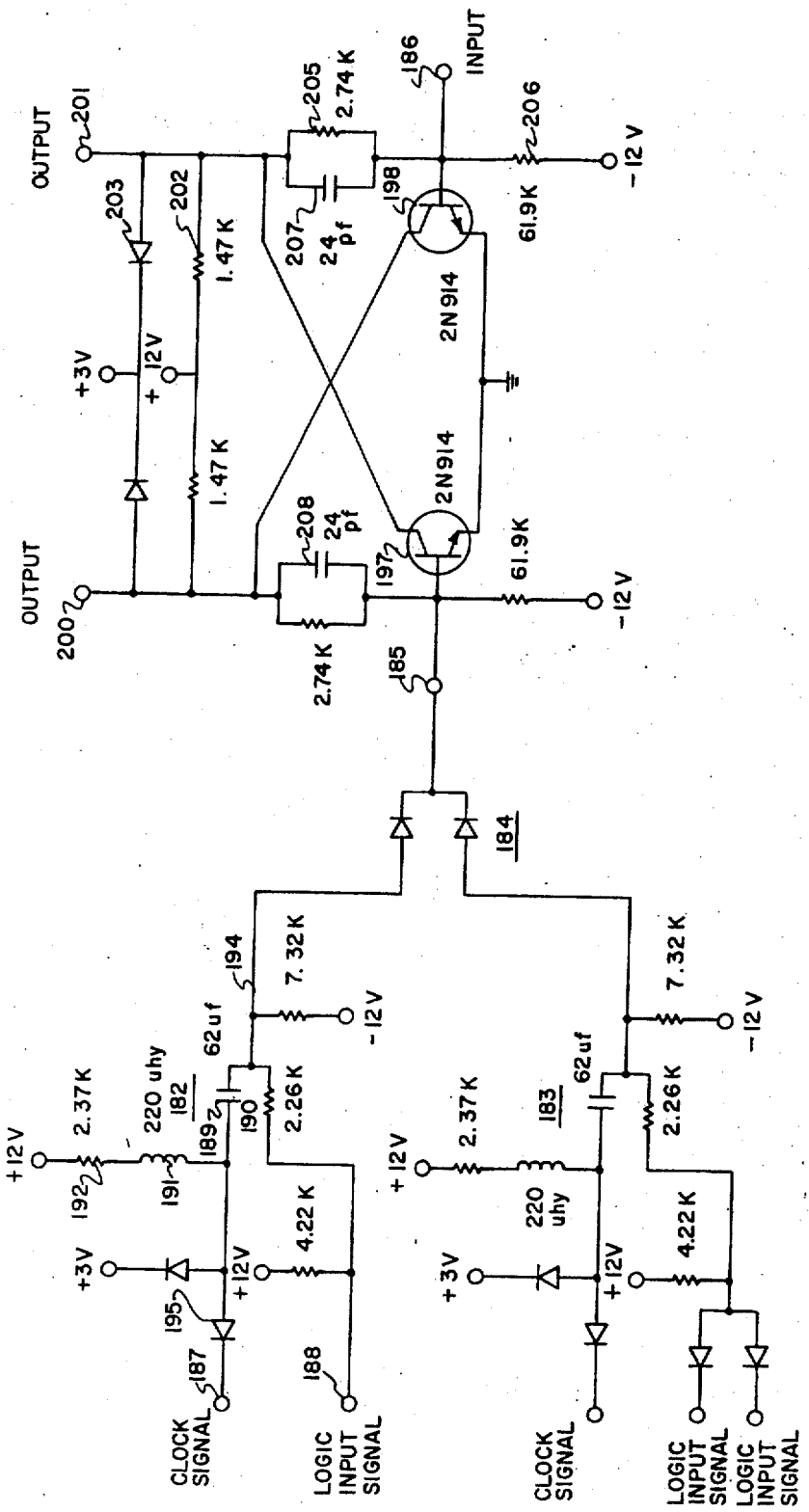


FIG. 7
PULSE PEDESTAL FLIP-FLOP

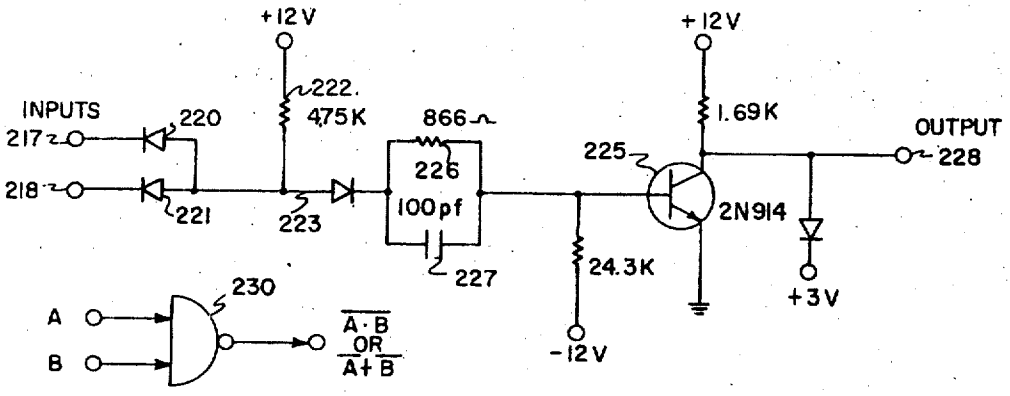
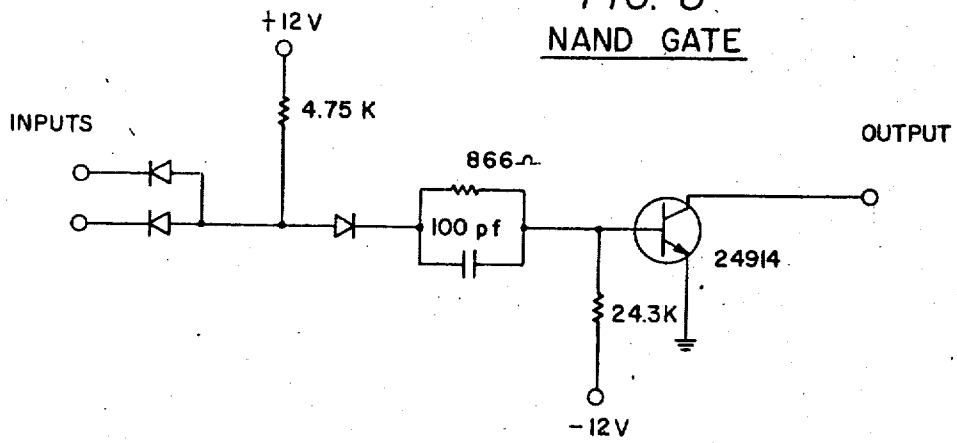


FIG. 8
NAND GATE



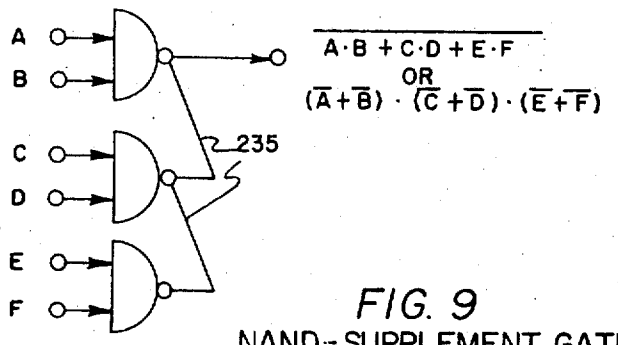


FIG. 9
NAND-SUPPLEMENT GATE

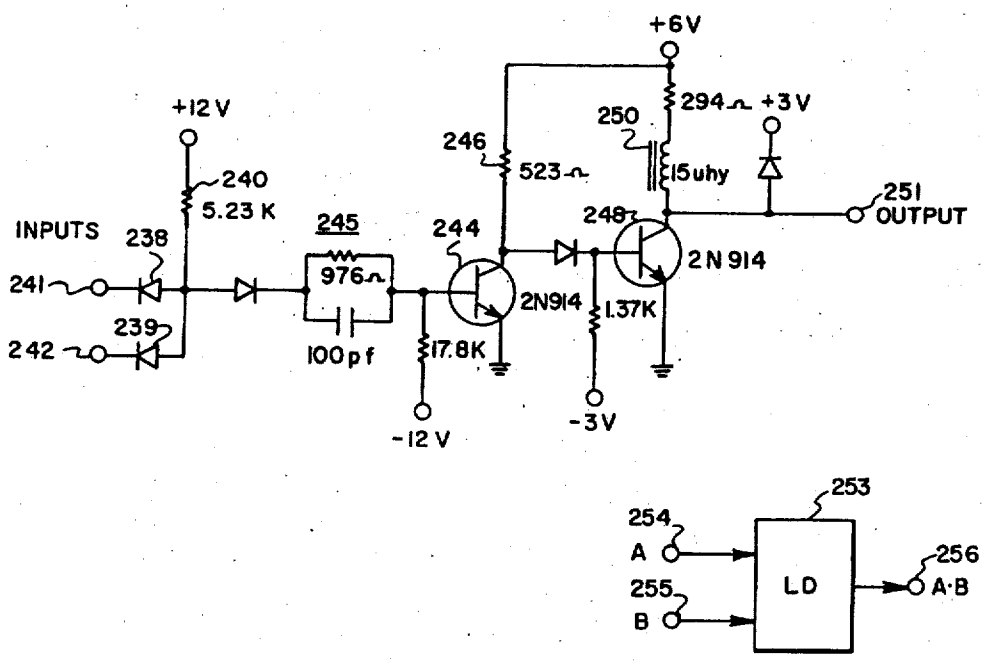


FIG. 10

LOGIC DRIVER

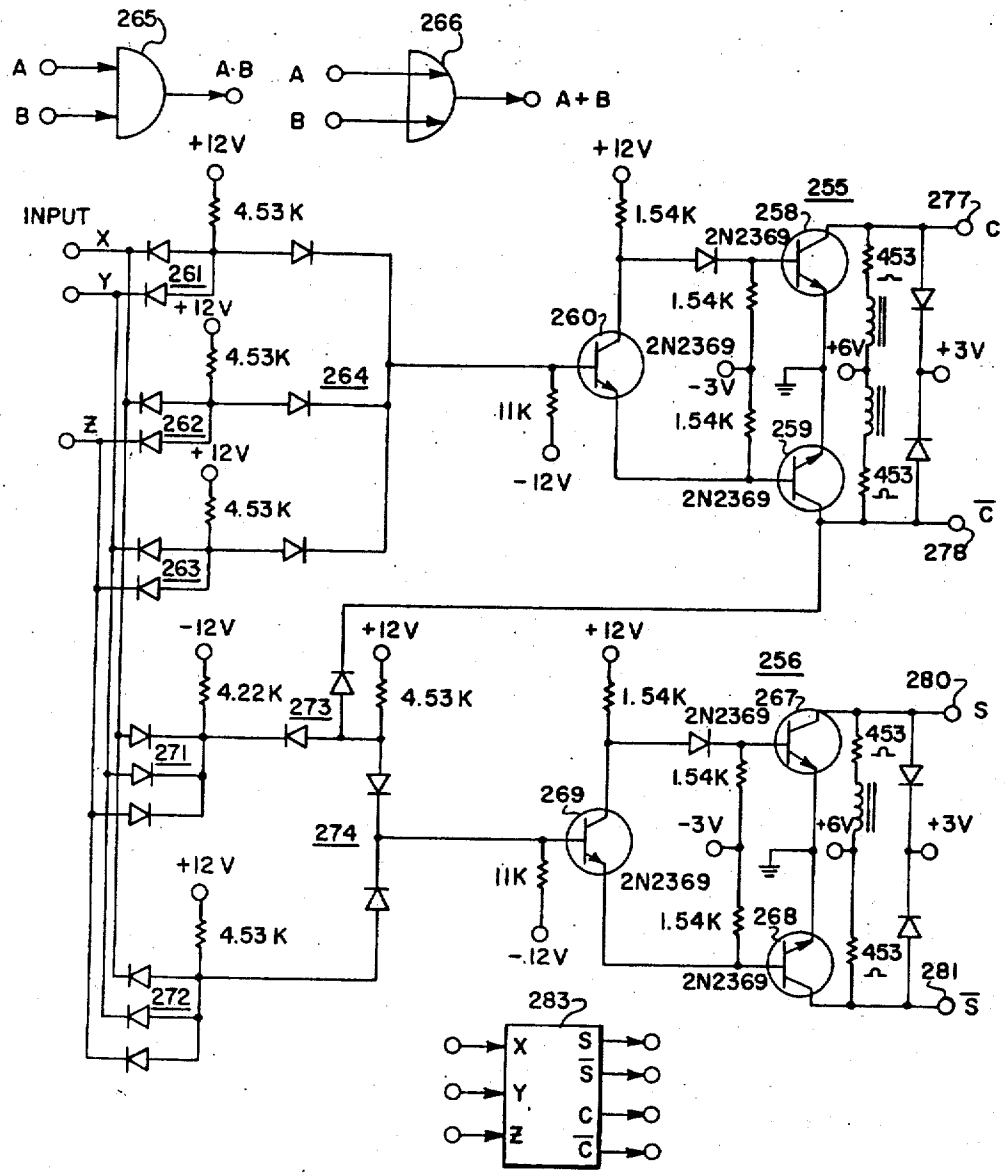


FIG. 11
FULL ADDER

CONTROL APPARATUS IN A DATA PROCESSING SYSTEM

Filed April 14, 1965 363 Sheets-Sheet 14

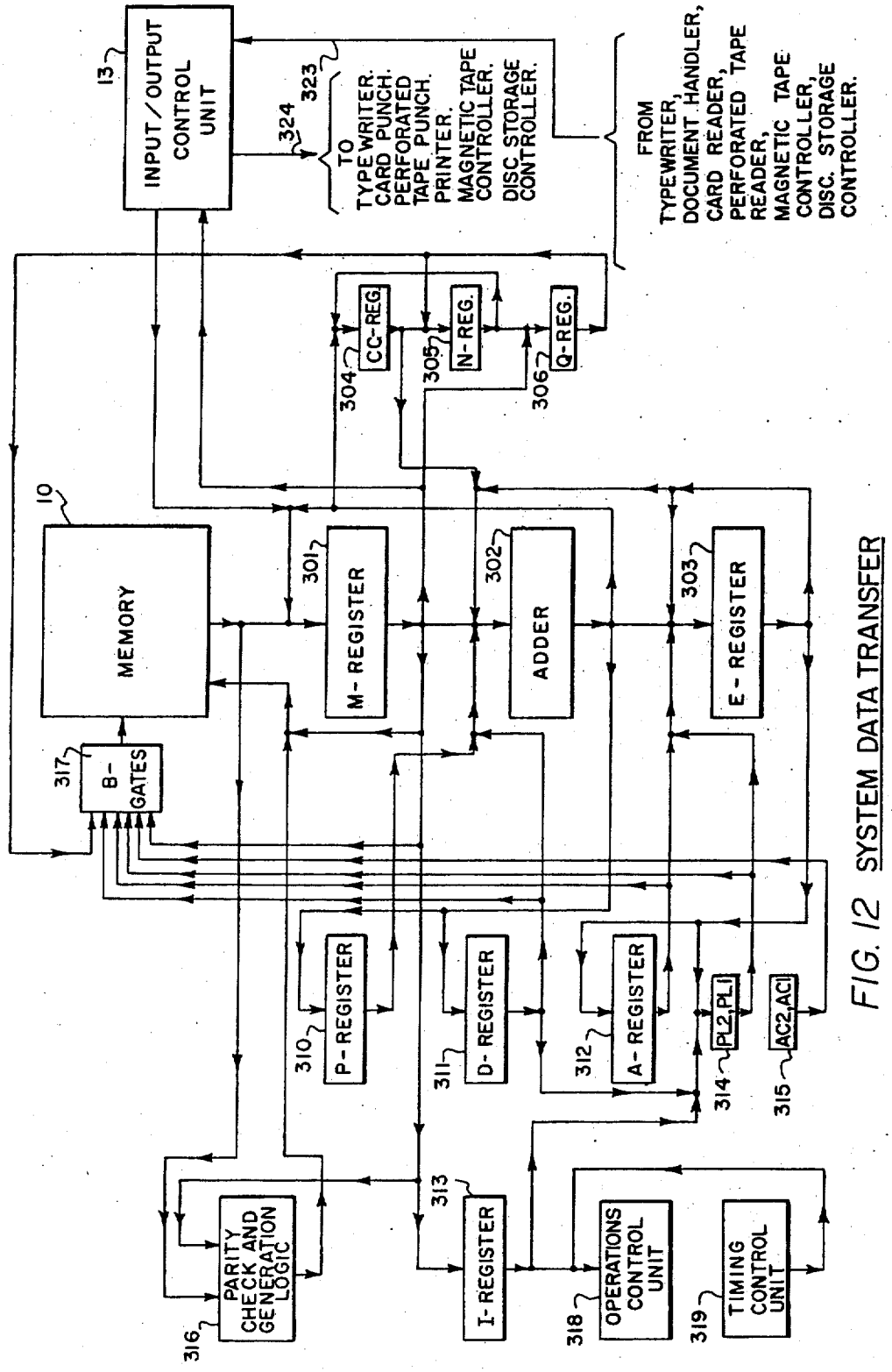


FIG. 12 SYSTEM DATA TRANSFER

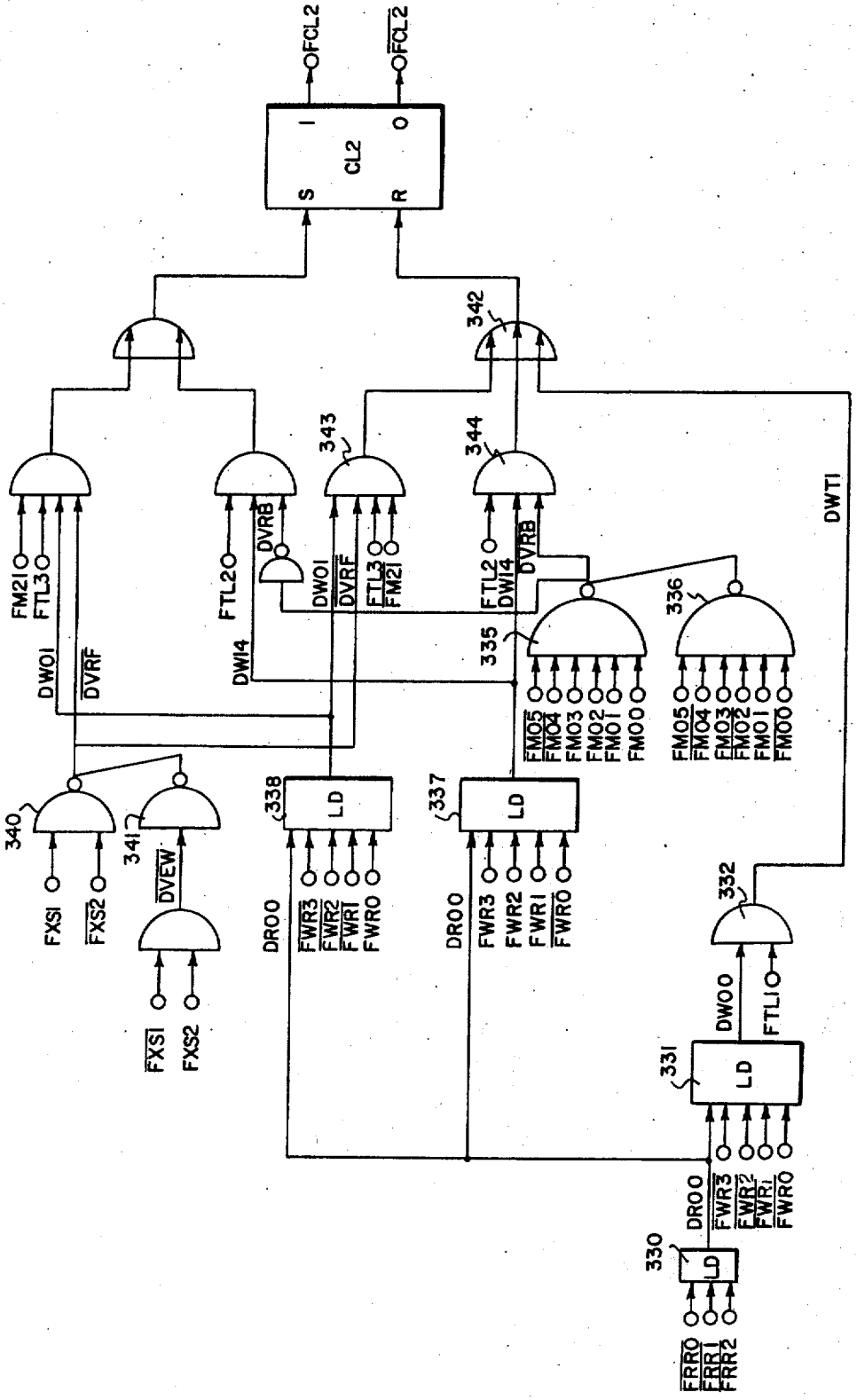

OFCL2
OFCL2
I
O
CL2
S
R
342
343
344
DWTI
FM21
FTL3
DWOI
DVRF
FTL2O
DVRB
DWI4
DWOI
DVRF
FTL3
FM21
FTL2
DWI4
DVRB
335
336
FM05
FM04
FM03
FM02
FM01
FM00
340
341
338
337
LD
FXS1
FXS2
DVEW
DR00
FWR3
FWR2
FWR1
FWR0
332
DW00
FTLIO
331
330
FRR0
FRR1
FRR2

FIG. 13

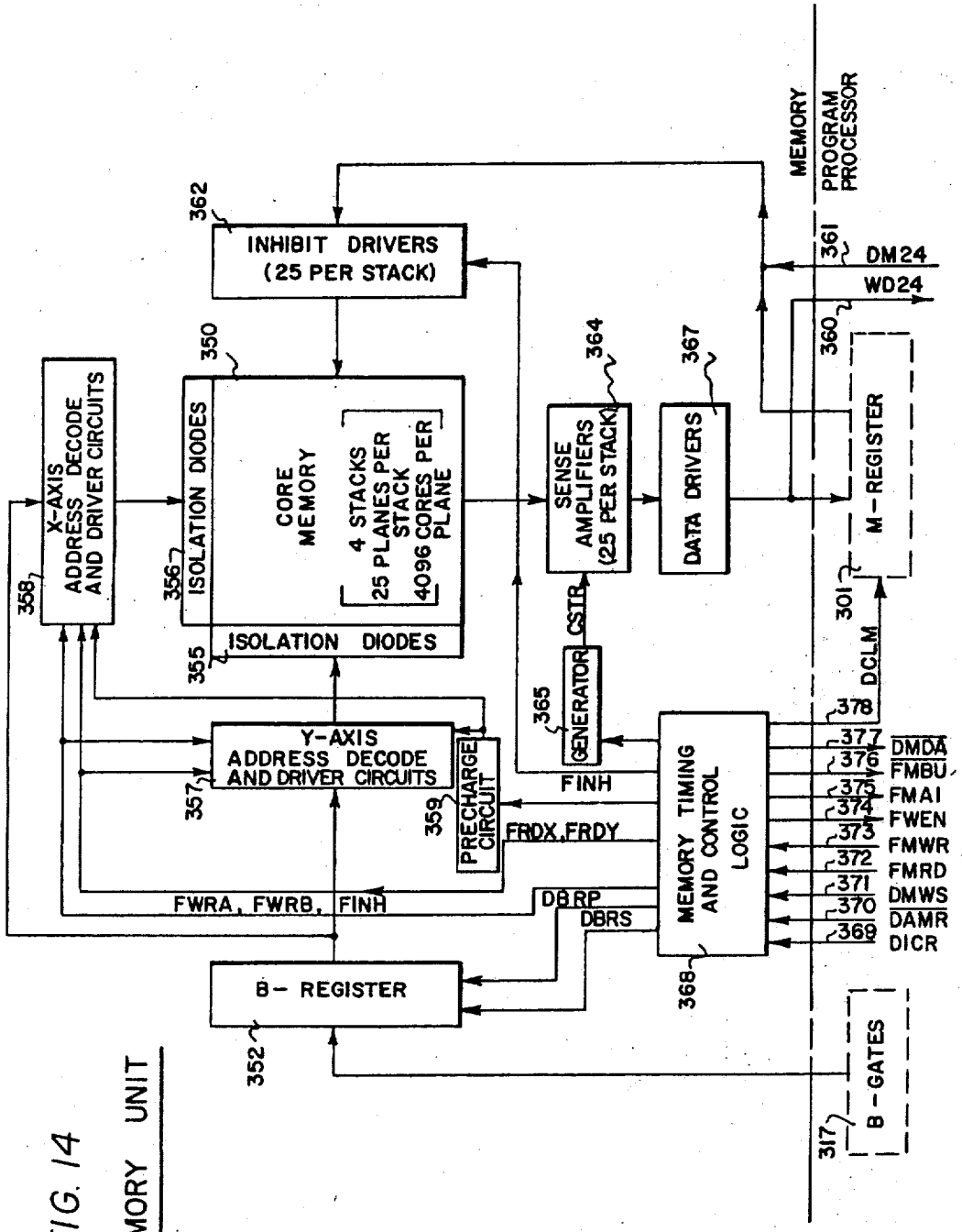


FIG. 14

16K MEMORY UNIT

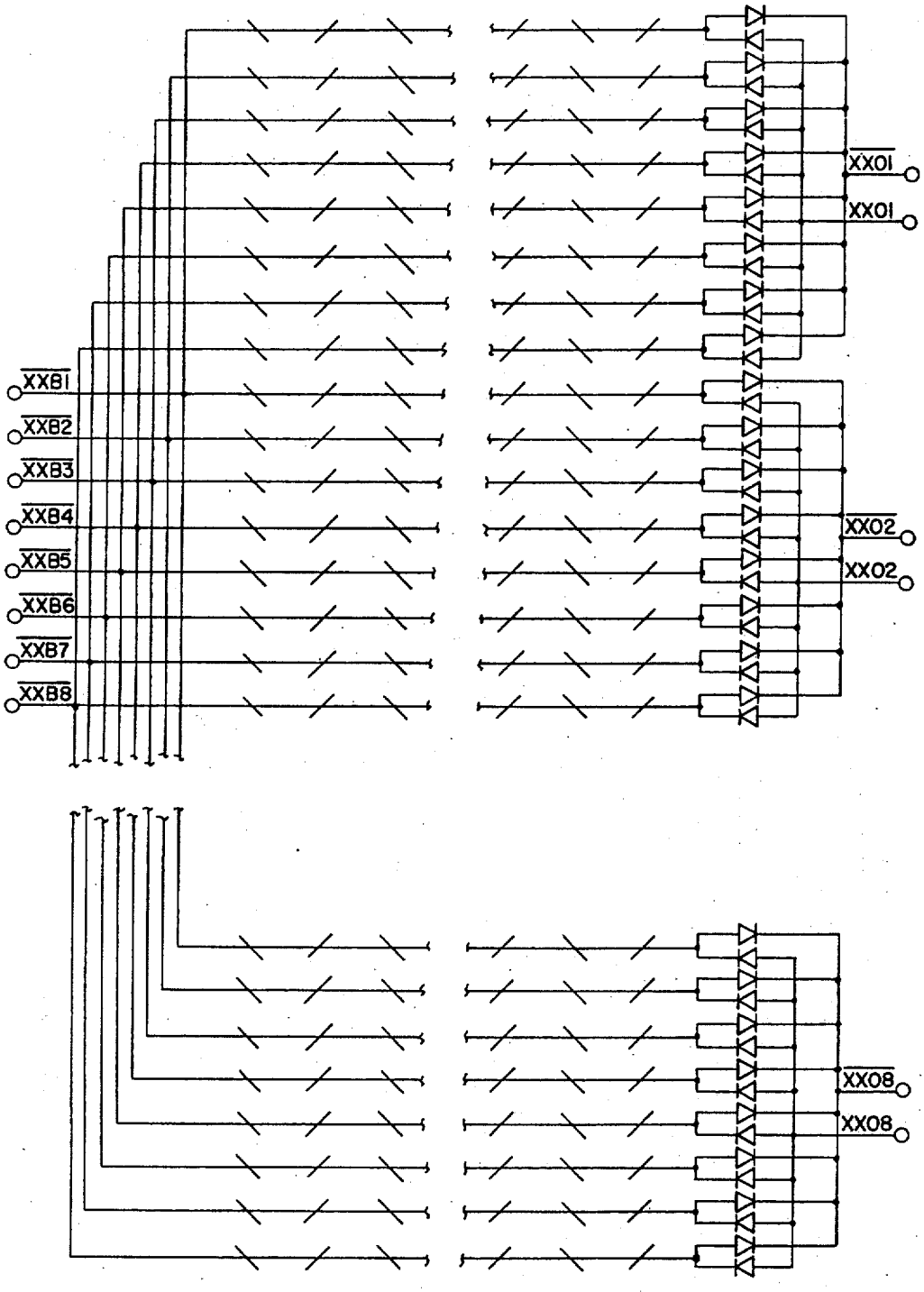


FIG. 15
MEMORY PLANE
ROW SELECTION

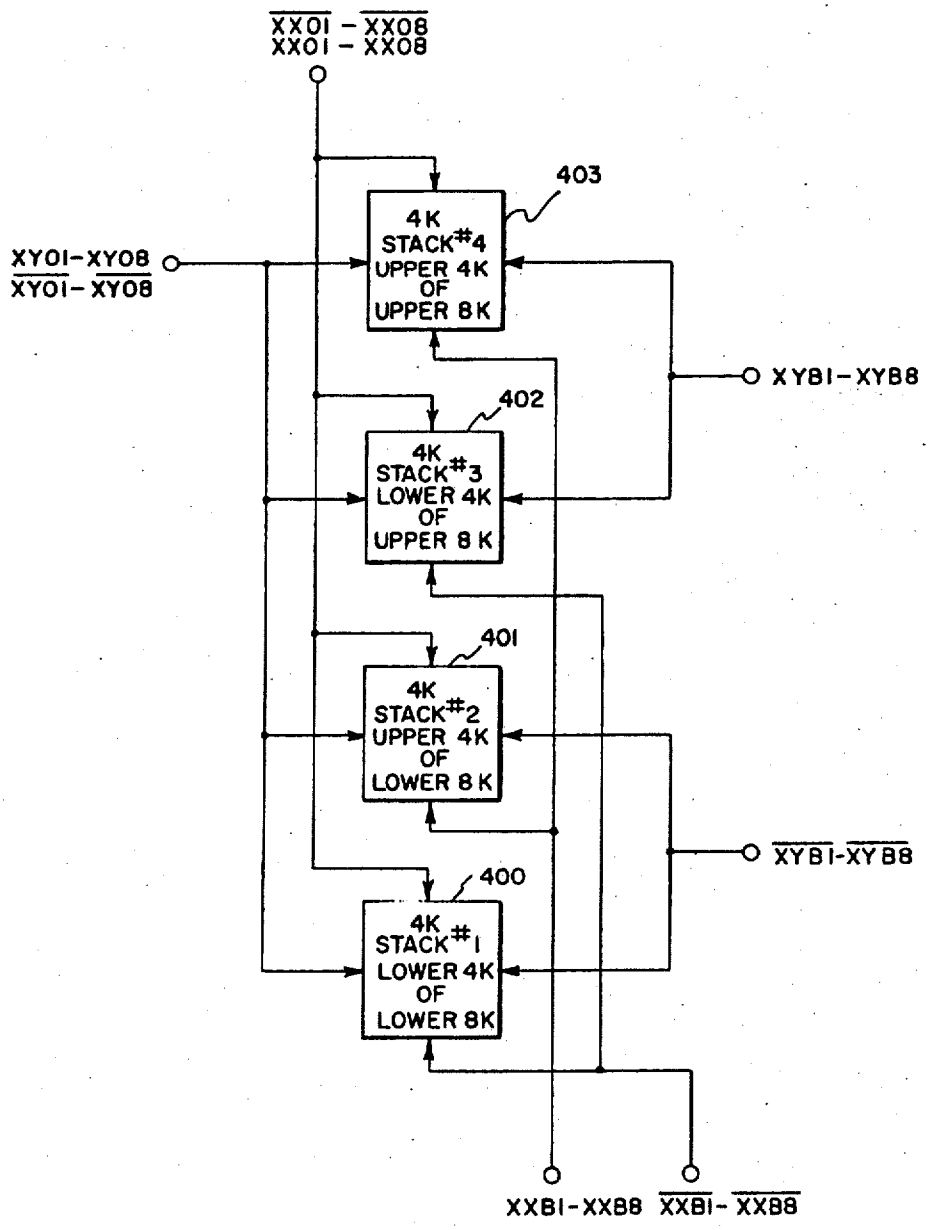


FIG. 16
16 K MEMORY UNIT ADDESSING

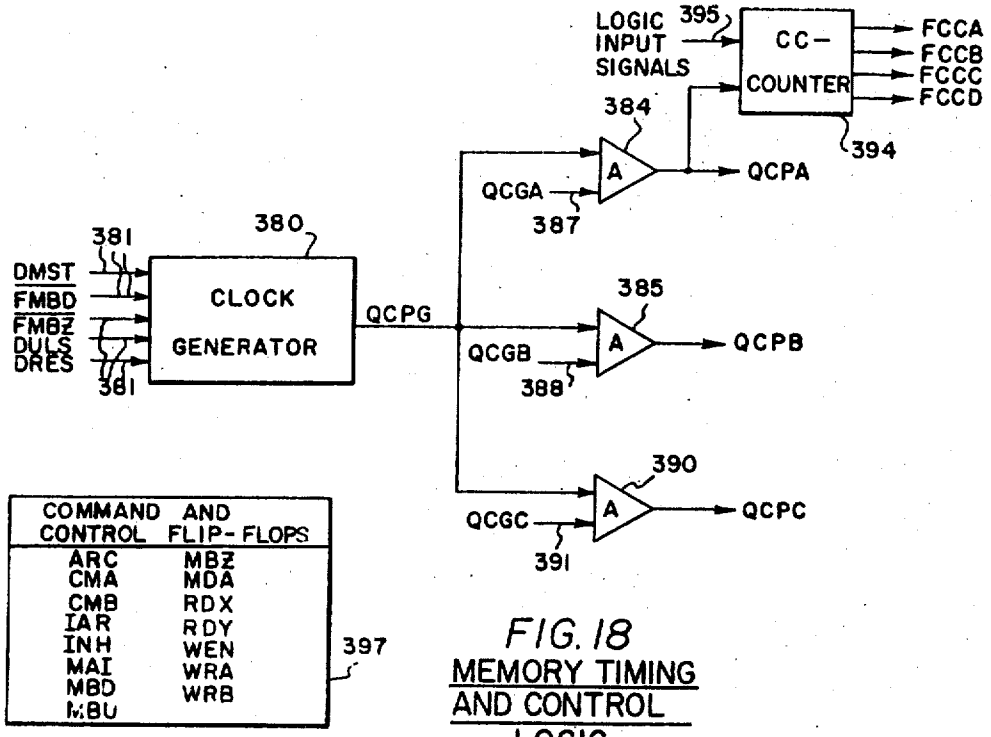


FIG. 18
MEMORY TIMING AND CONTROL LOGIC

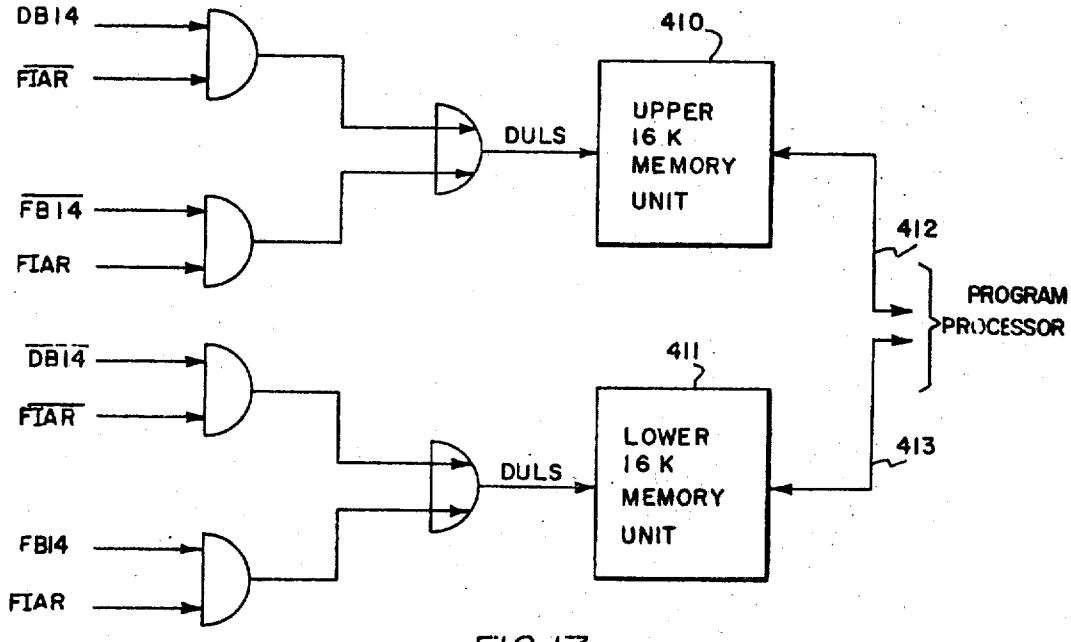


FIG. 17
32 K MEMORY

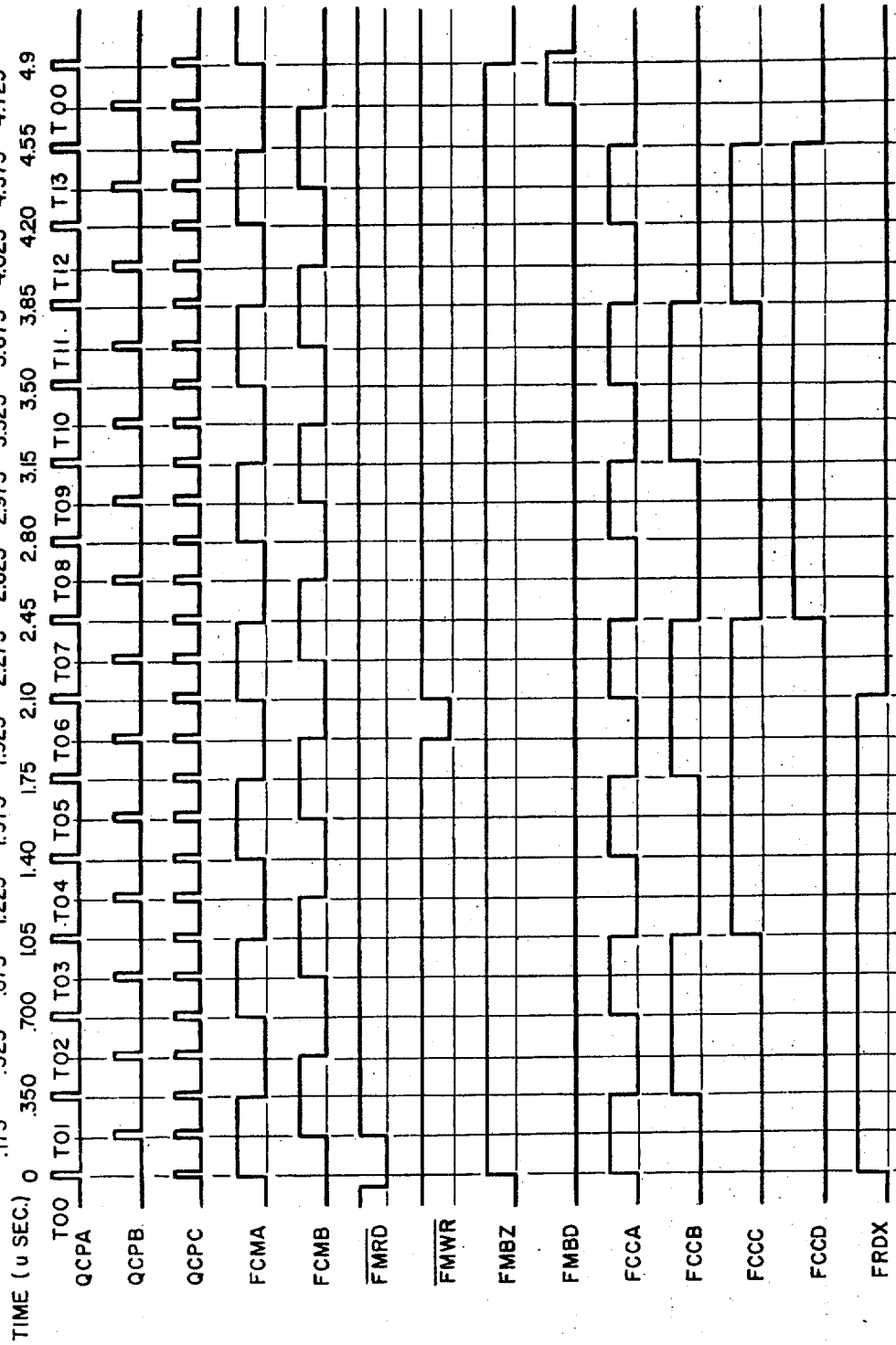


FIG. 19a MEMORY TIMING

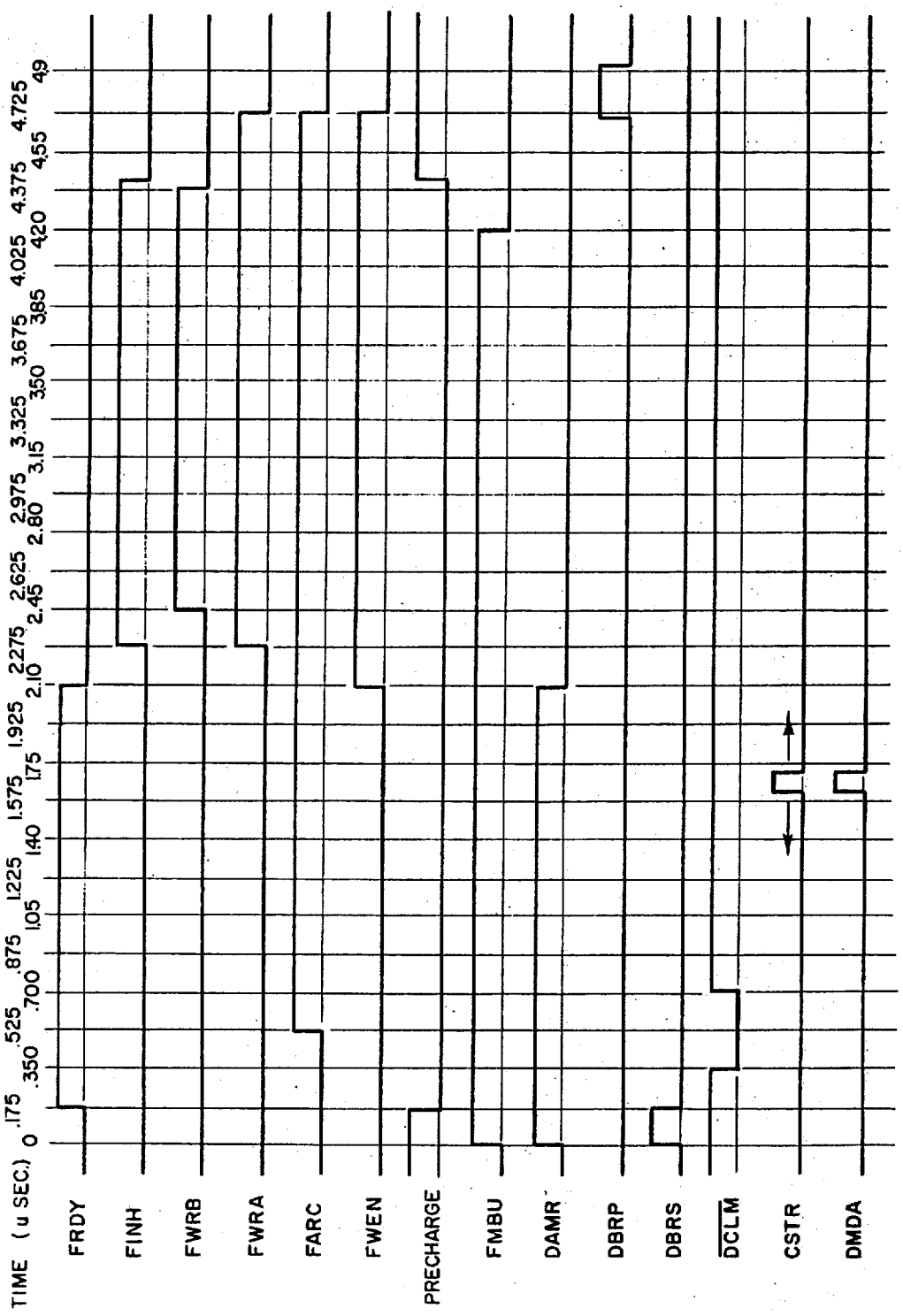
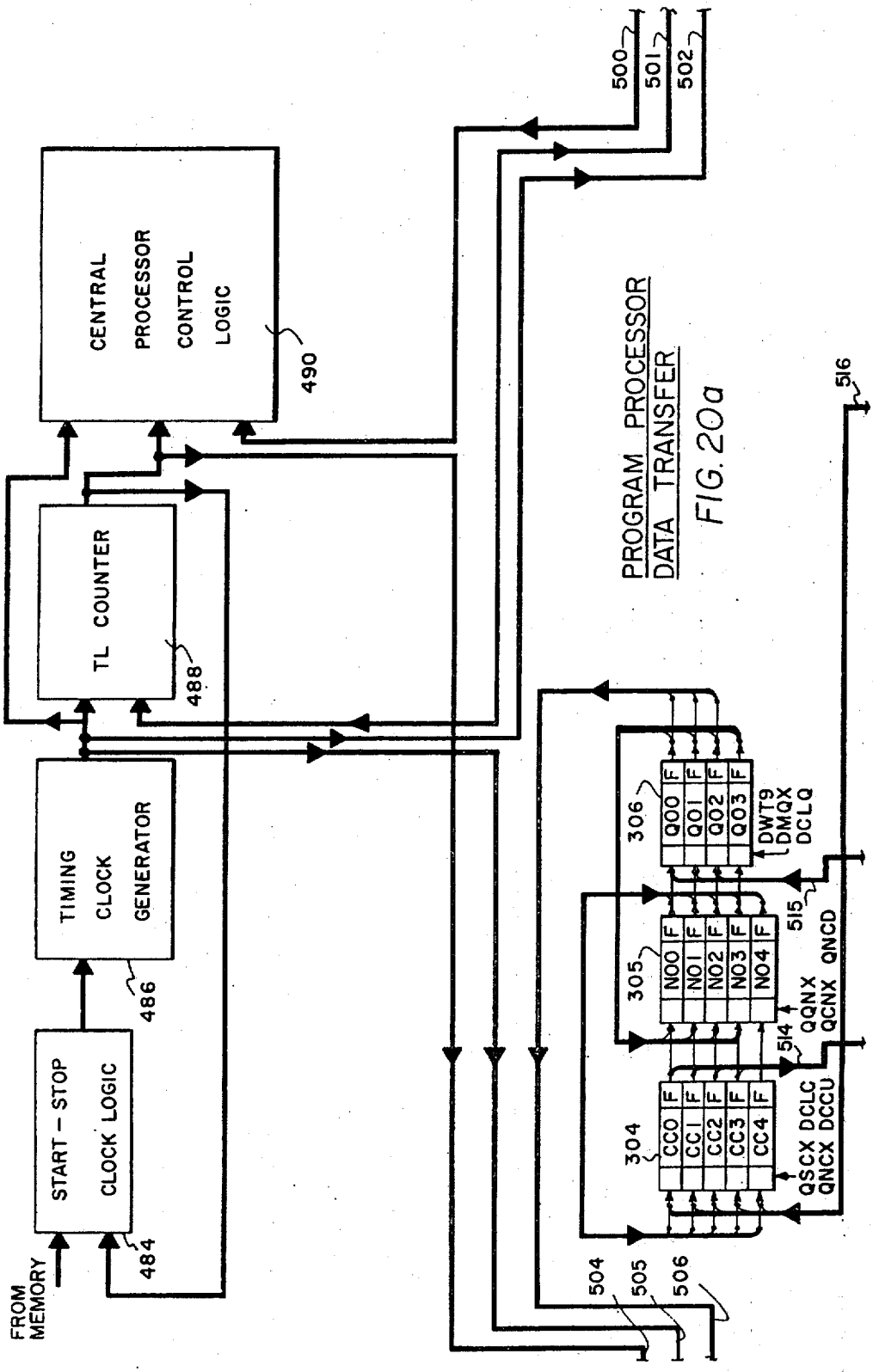

TIME (u SEC.)
0
.175
.350
.525
.700
.875
1.05
1.225
1.40
1.575
1.75
1.925
2.10
2.275
2.45
2.625
2.80
2.975
3.15
3.325
3.50
3.675
3.85
4.025
4.20
4.375
4.55
4.725
4.9
FRDY
FINH
FWRB
FWRA
FARC
FWEN
PRECHARGE
FMBU
DAMR
DBRP
DBRS
DCLM
CSTR
DMDA

FIG. 19b

CONTROL APPARATUS IN A DATA PROCESSING SYSTEM

Filed April 14, 1965 363 Sheets-Sheet 22



CONTROL APPARATUS IN A DATA PROCESSING SYSTEM

Filed April 14, 1965 3 Sheets-Sheet 23

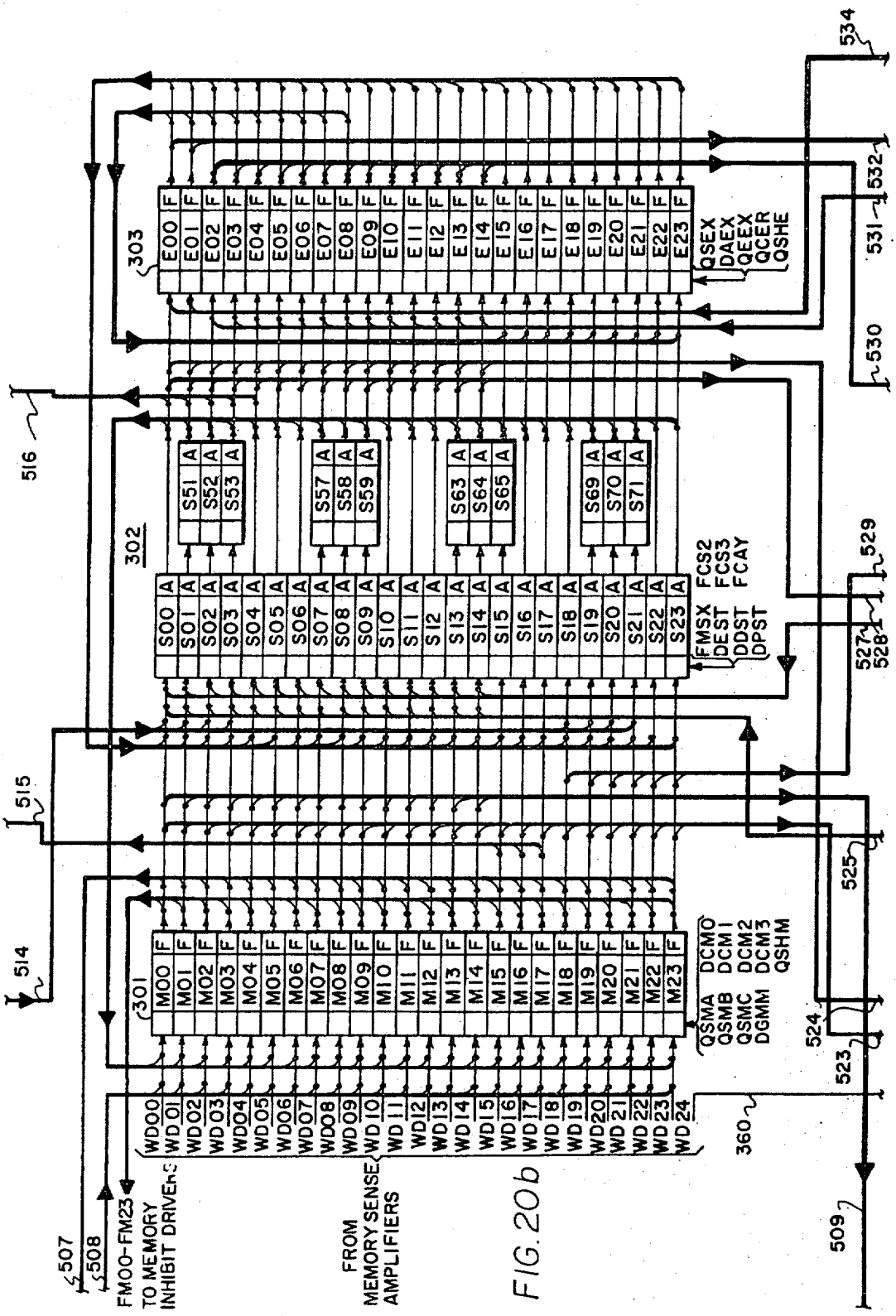


FIG. 20b

CONTROL APPARATUS IN A DATA PROCESSING SYSTEM

Filed April 14, 1965 363 Sheets-Sheet 24

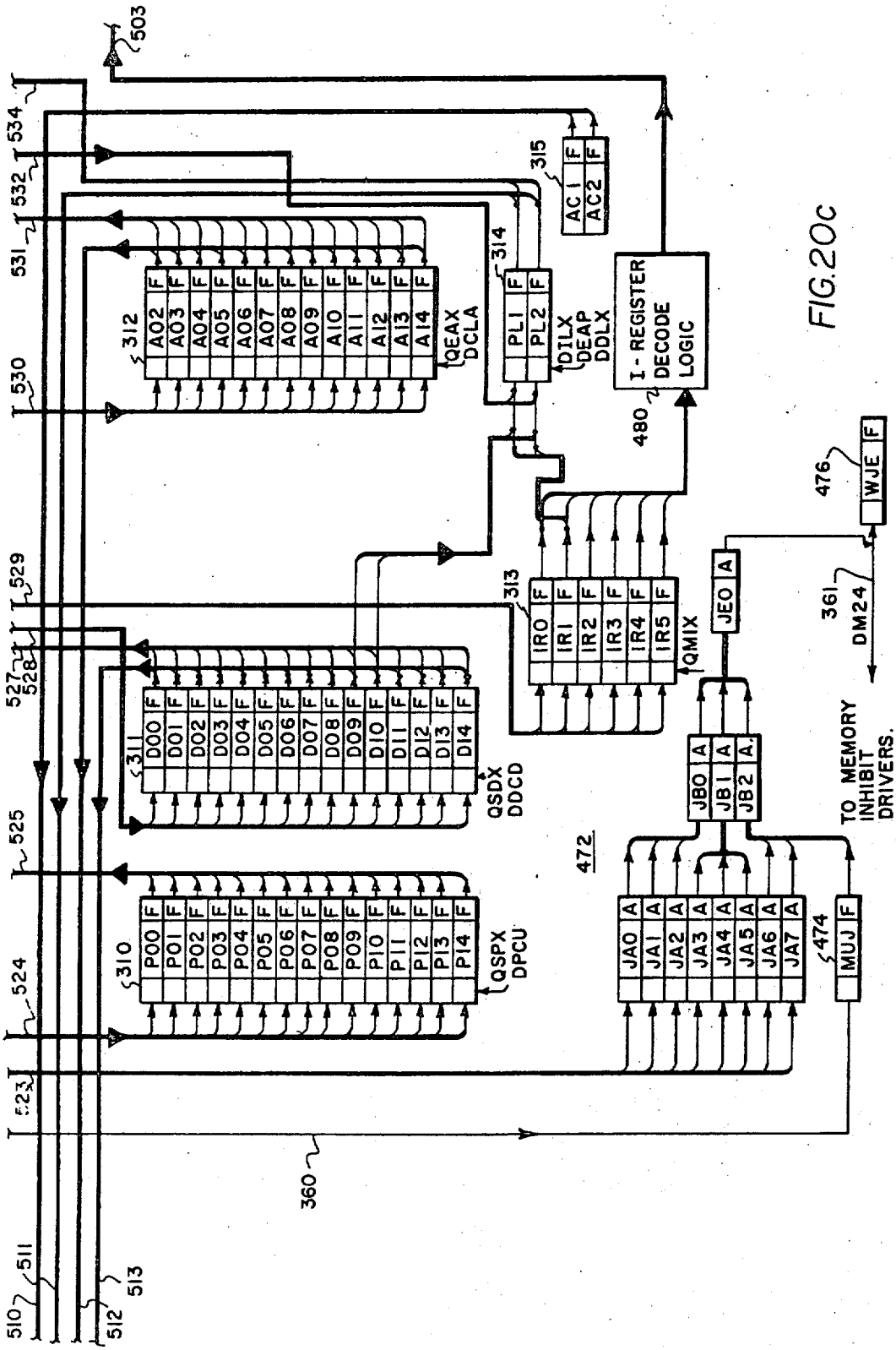


FIG. 20c

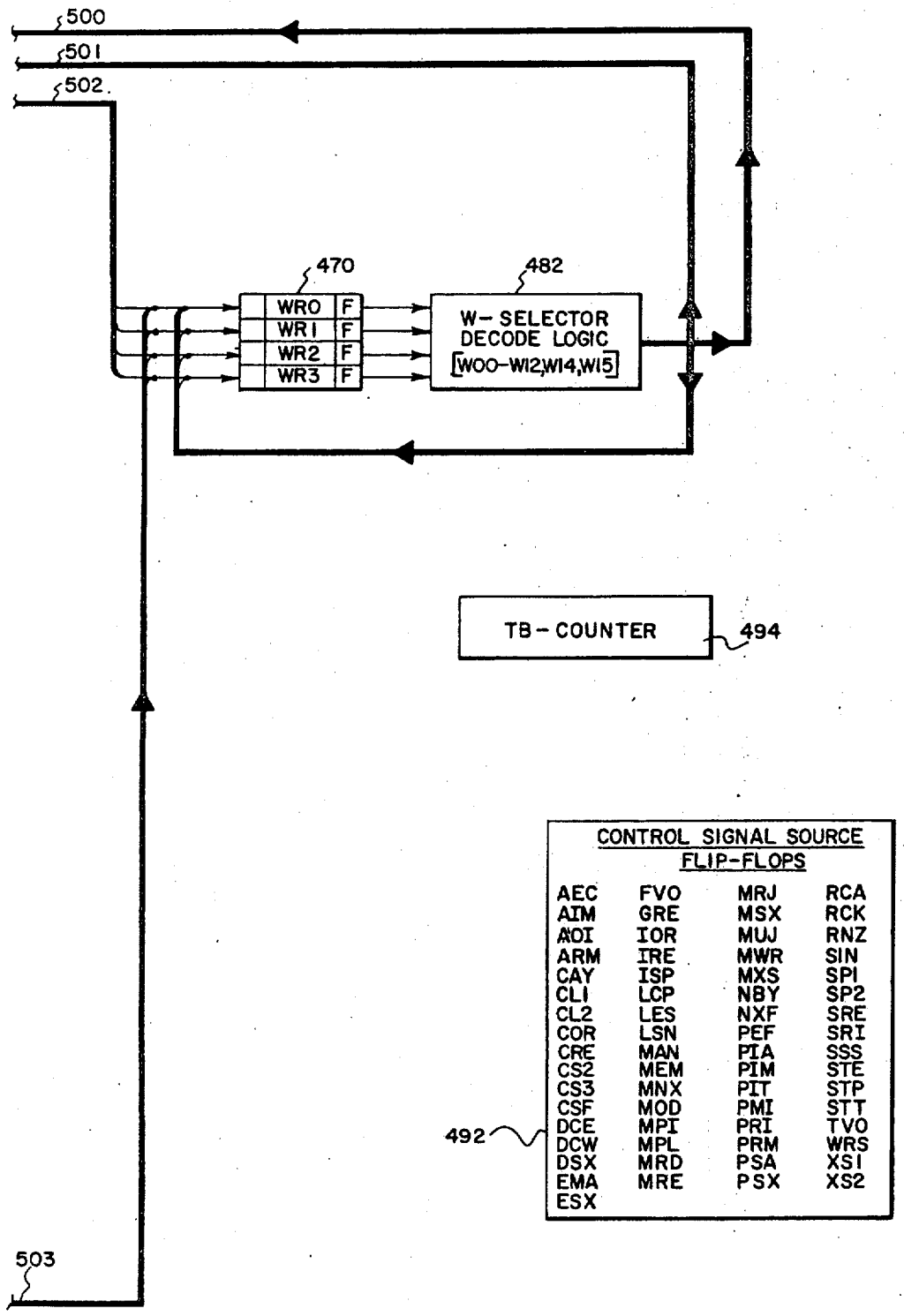

500
501
502
470
482
WR0 F
WR1 F
WR2 F
WR3 F
W-SELECTOR DECODE LOGIC [W00-W12,W14,W15]
TB-COUNTER
494
CONTROL SIGNAL SOURCE FLIP-FLOPS
AEC AIM AOI ARM CAY CLI CL2 COR CRE CS2 CS3 CSF DCE DCW DSX EMA ESX
FVO GRE IOR IRE ISP LCP LES LSN MAN MEM MNX MOD MPI MPL MRD MRE
MRJ MSX MUJ MWR MXS NBY NXF PEF PIA PIM PIT PMI PRI PRM PSA PSX
RCA RCK RNZ SIN SPI SP2 SRE SRI SSS STE STP STT TVO WRS XSI XS2
492
503

FIG. 20 d

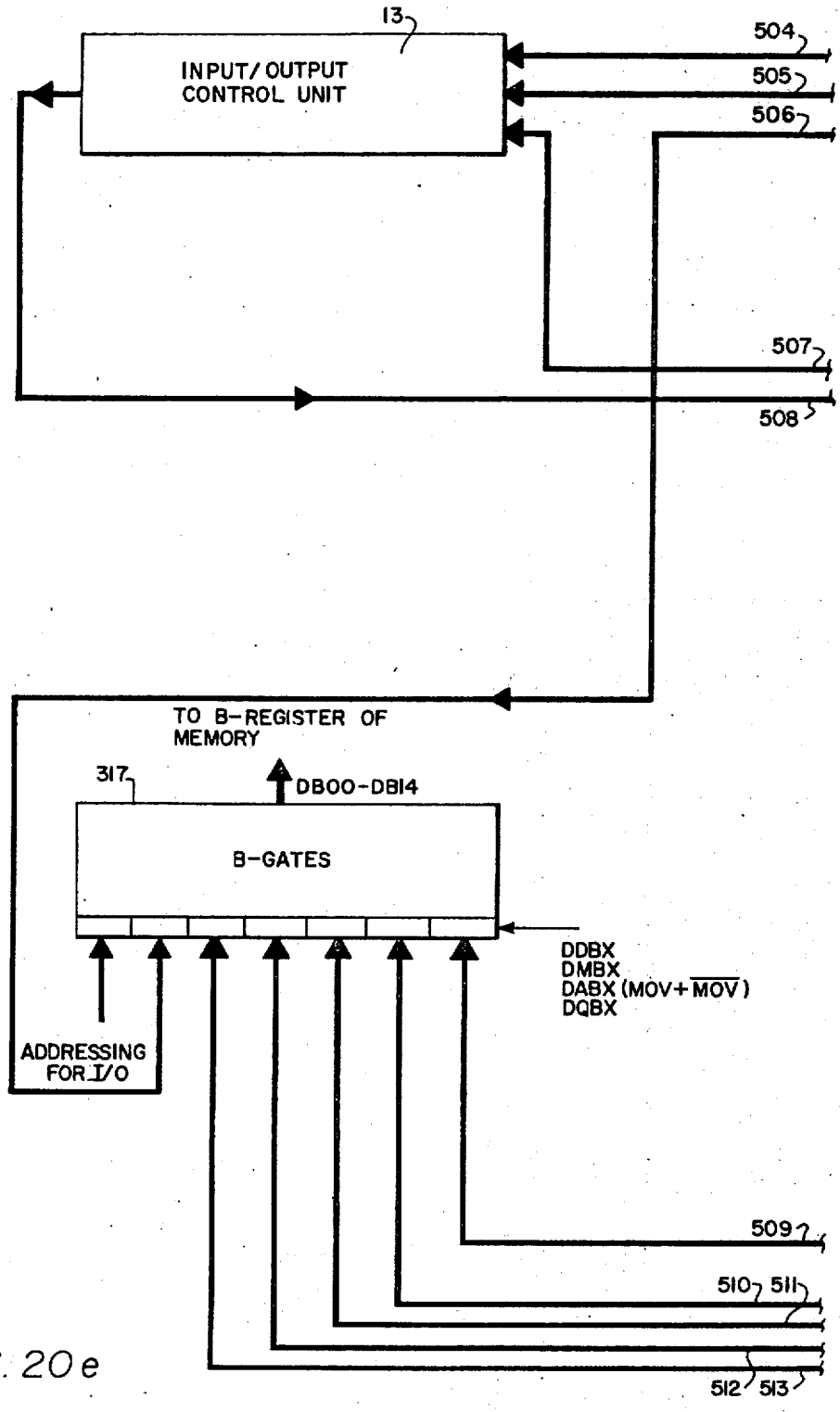

13
INPUT/OUTPUT CONTROL UNIT
504
505
506
507
508
TO B-REGISTER OF MEMORY
317
DB00-DB14
B-GATES
DDBX
DMBX
DABX (MOV+$\overline{MOV}$)
DQBX
ADDRESSING FOR I/O
509
510
511
512
513

FIG. 20e

CONTROL APPARATUS IN A DATA PROCESSING SYSTEM

Filed April 14, 1965 363 Sheets-Sheet 27

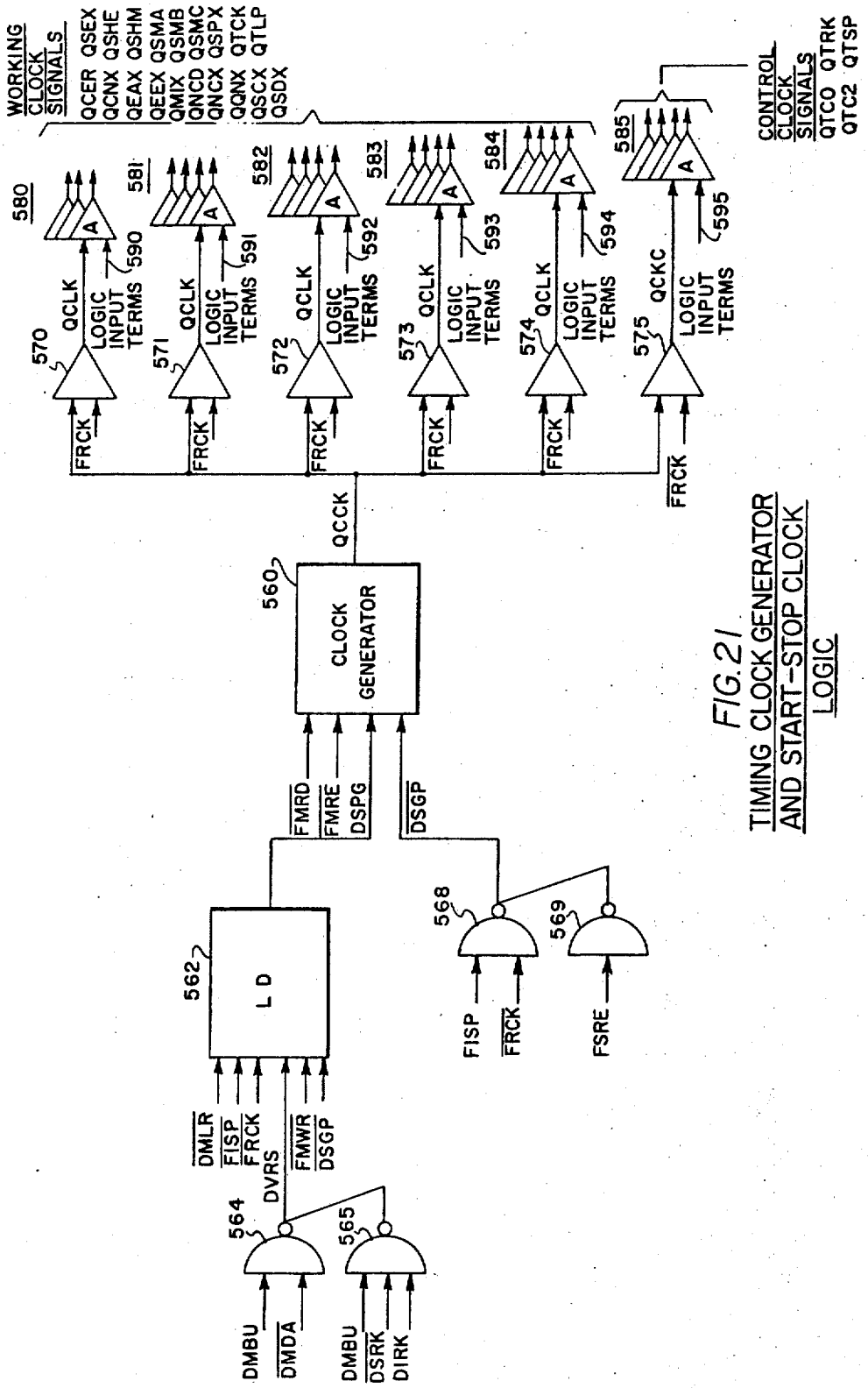


FIG. 21
TIMING CLOCK GENERATOR AND START-STOP CLOCK LOGIC

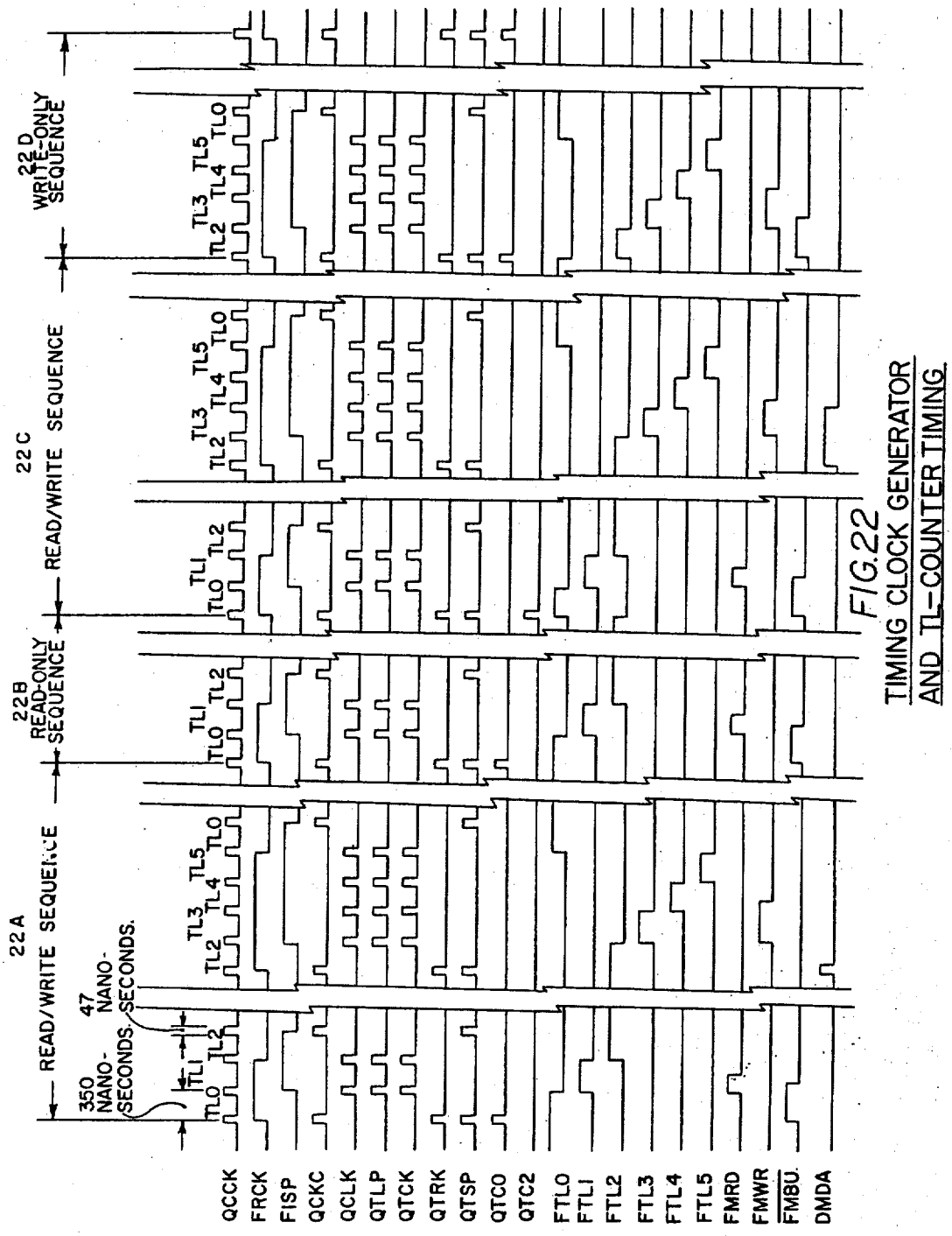


FIG.22
TIMING CLOCK GENERATOR AND TL-COUNTER TIMING

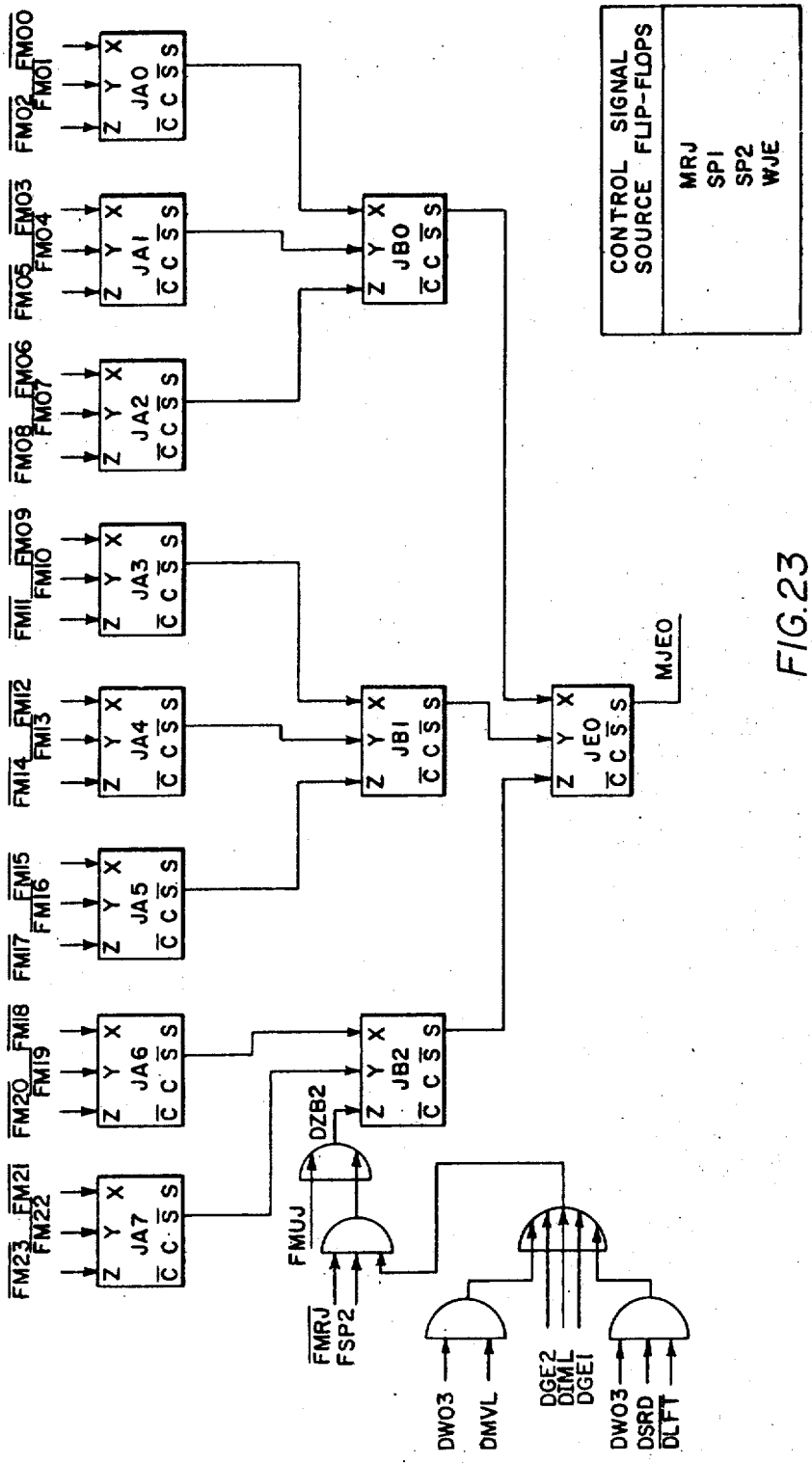


FIG. 23
PARITY CHECK AND GENERATION LOGIC

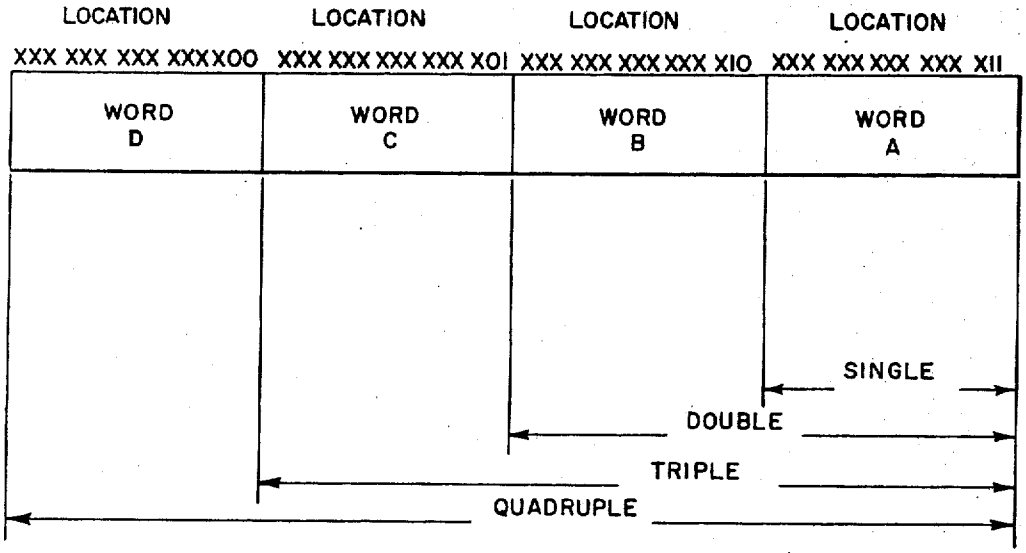
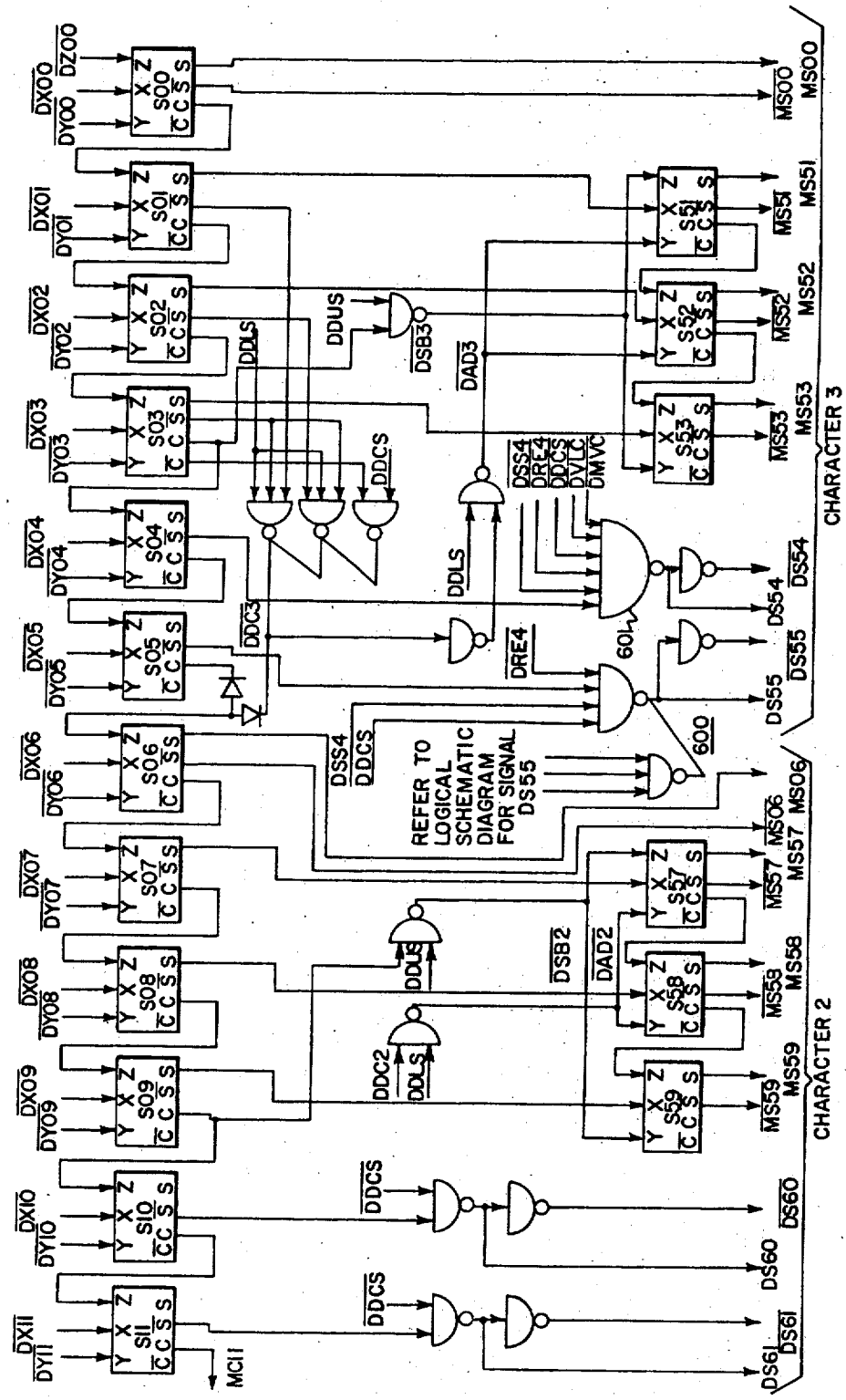


FIG. 24

RELOCATABLE ACCUMULATOR

FIG. 25a
ADDER



CONTROL APPARATUS IN A DATA PROCESSING SYSTEM

Filed April 14, 1965 363 Sheets-Sheet 32

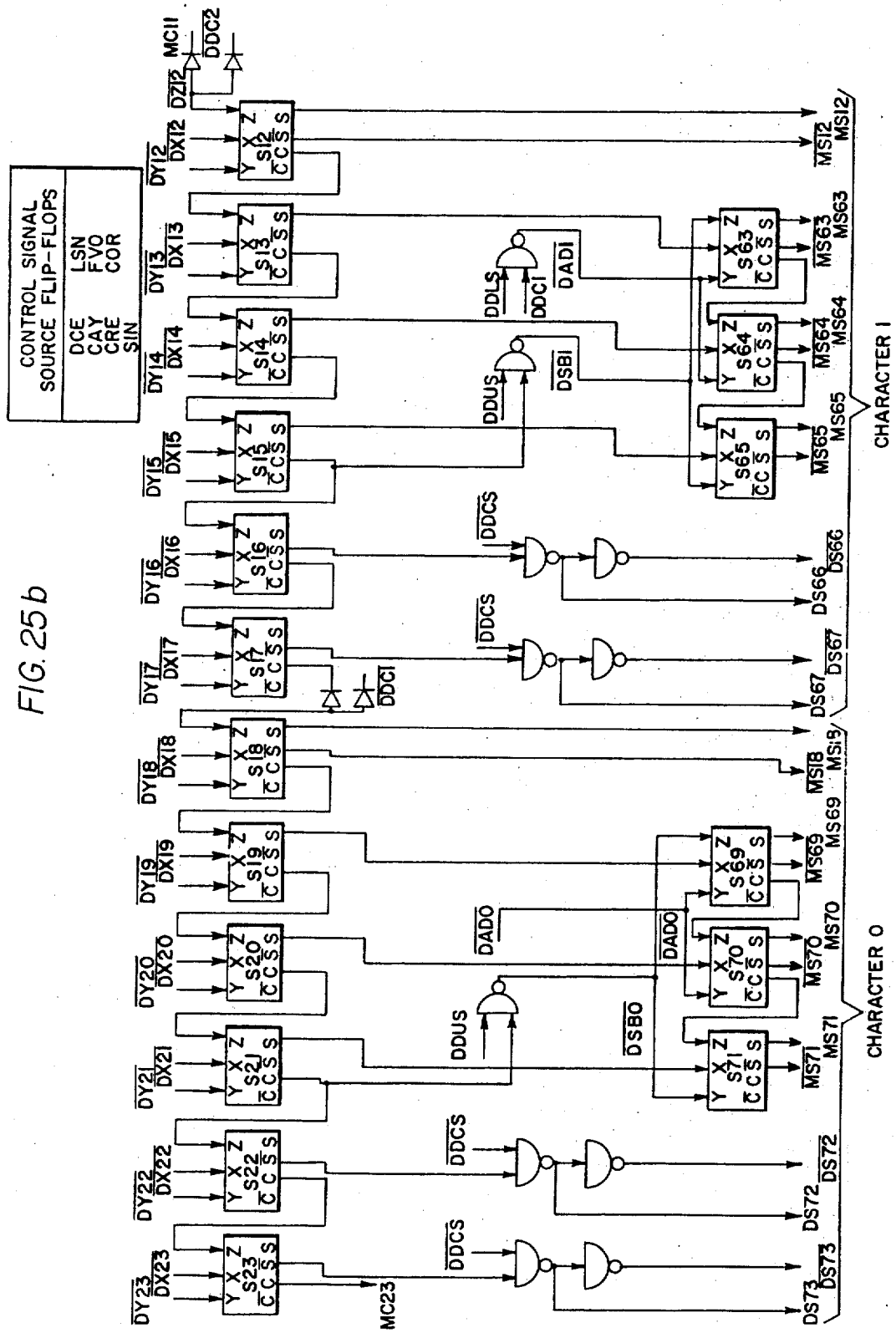


FIG. 25b

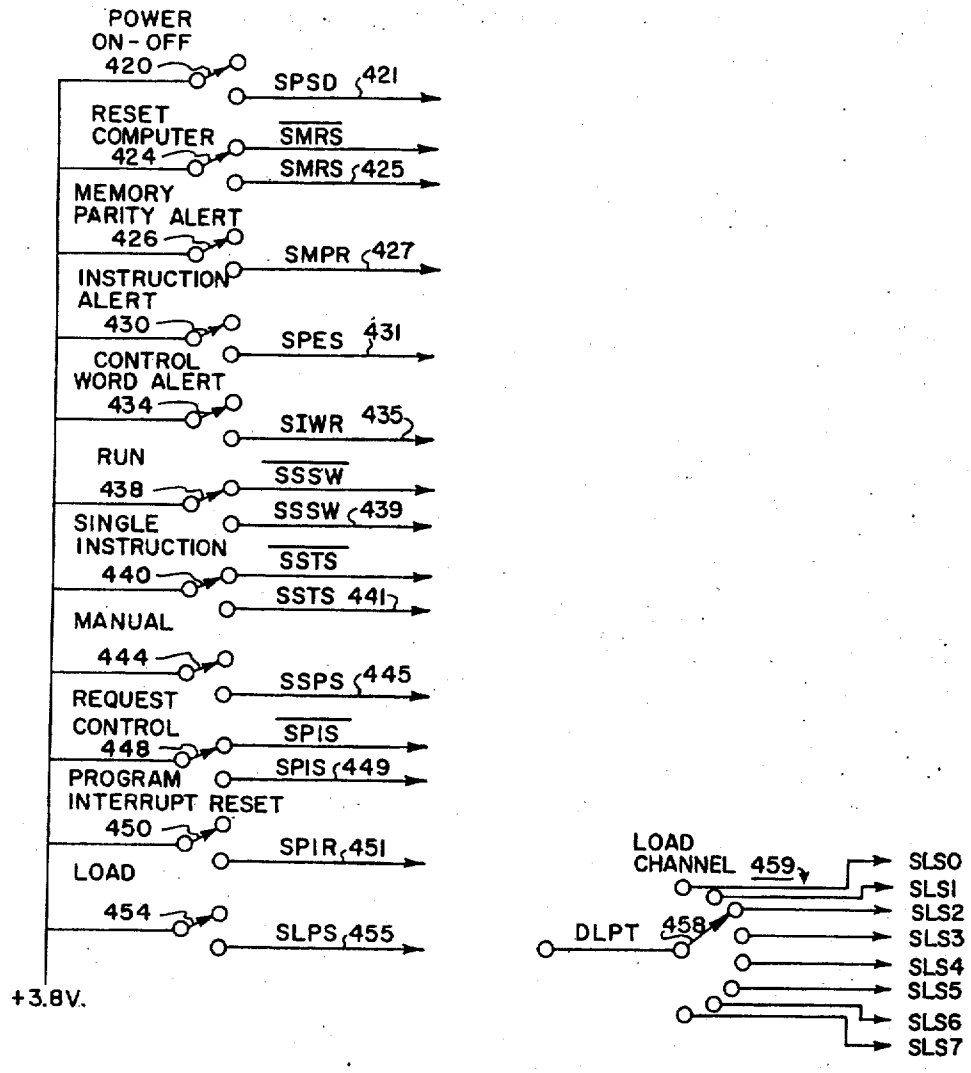
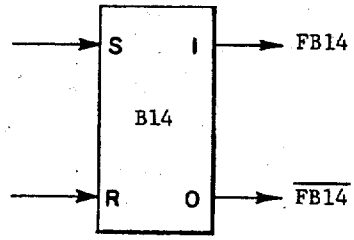


FIG. 26
CONTROL CONSOLE SWITCHES

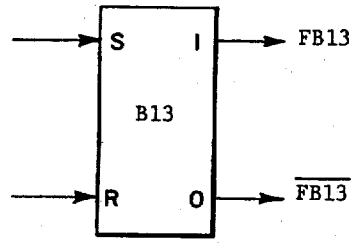
$FB14 = \overline{DB14}\ DBRS$

$\overline{FB14} = DBRP$



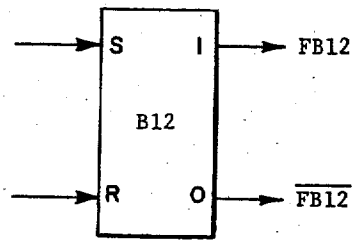
$FB13 = \overline{DB13}\ DBRS$

$\overline{FB13} = DBRP$



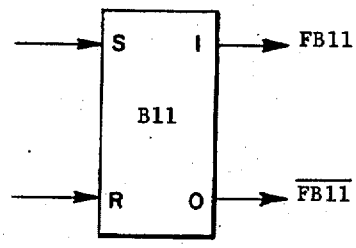
$FB12 = \overline{DB12}\ DBRS$

$\overline{FB12} = DBRP$



$FB11 = \overline{DB11}\ DBRS$

$\overline{FB11} = DBRP$

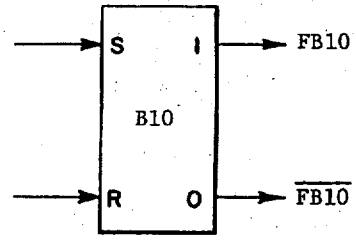


MEMORY

FIG. 27

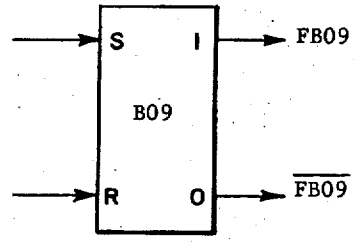
$FB10 = \overline{DB10}\ DBRS$

$\overline{FB10} = DBRP$



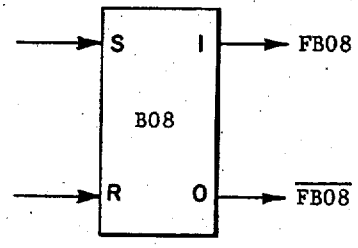
$FB09 = \overline{DB09}\ DBRS$

$\overline{FB09} = DBRP$



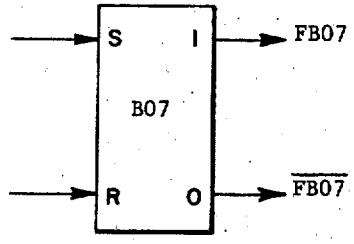
$FB08 = \overline{DB08}\ DBRS$

$\overline{FB08} = DBRP$



$FB07 = \overline{DB07}\ DBRS$

$\overline{FB07} = DBRP$

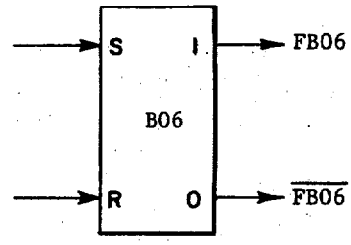


MEMORY

FIG. 28.

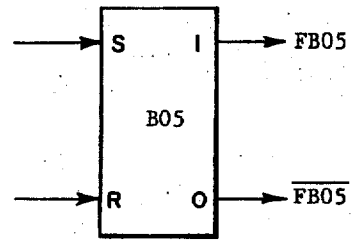
$FB06 = \overline{DB06}\ DBRS$

$\overline{FB06} = DBRP$



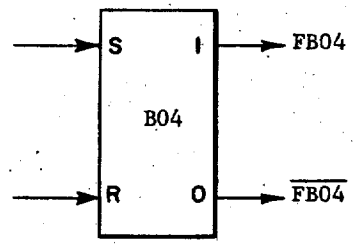
$FB05 = \overline{DB05}\ DBRS$

$\overline{FB05} = DBRP$



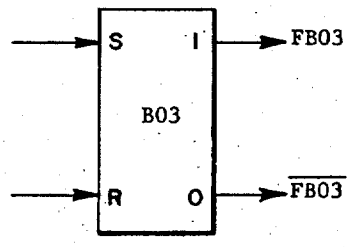
$FB04 = \overline{DB04}\ DBRS$

$\overline{FB04} = DBRP$



$FB03 = \overline{DB03}\ DBRS$

$\overline{FB03} = DBRP$

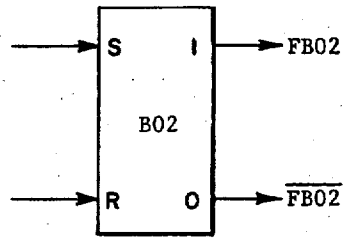


MEMORY

FIG. 29.

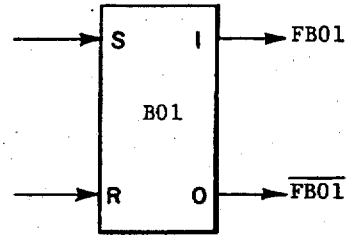
$FB02 = \overline{DB02}\ DBRS$

$\overline{FB02} = DBRP$



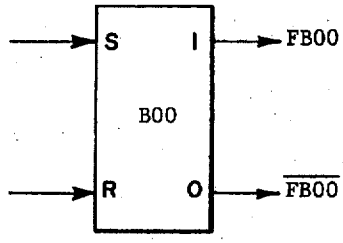
$FB01 = \overline{DB01}\ DBRS$

$\overline{FB01} = DBRP$



$FB00 = \overline{DB00}\ DBRS$

$\overline{FB00} = DBRP$

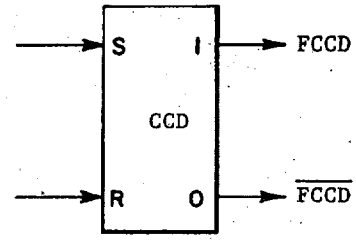


MEMORY

FIG. 30.

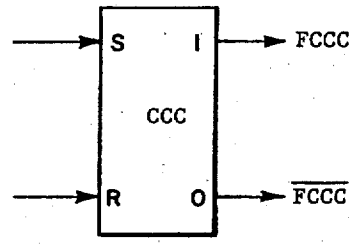
FCCD = FCCA FCCB FCCC FWEN QCPA

$\overline{FCCD}$ = DT13 QCPA + FMBD + DRES



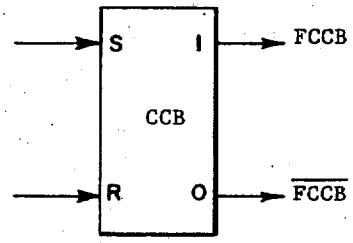
FCCC = (FCCA FCCB $\overline{FCCC}$ + DMWT) QCPA

$\overline{FCCC}$ = (DT07 + DT13) QCPA + DRES



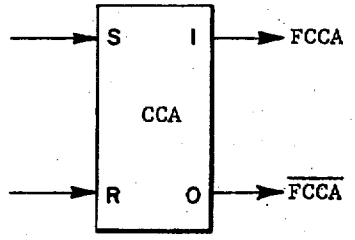
FCCB = (FCCA $\overline{FCCB}$ $\overline{DT13}$ + DMWT) QCPA

$\overline{FCCB}$ = FCCA FCCB QCPA + FMBD + DRES



FCCA = ($\overline{FCCA}$ + DMWT) QCPA

$\overline{FCCA}$ = FCCA QCPA + DRES

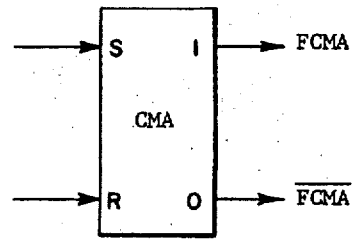


MEMORY

FIG. 31.

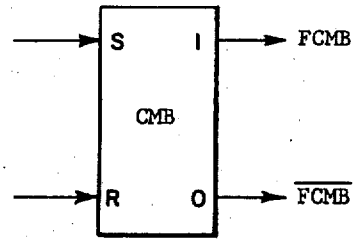
FCMA = $\overline{\text{FCMA}}$ QCPA

$\overline{\text{FCMA}}$ = FCMA QCPA + DRES



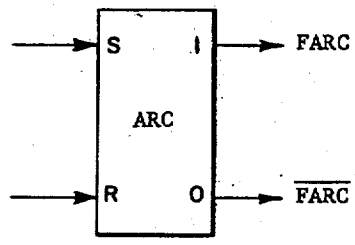
FCMB = $\overline{\text{FCMB}}$ QCPB

$\overline{\text{FCMB}}$ = FCMB QCPB + DRES



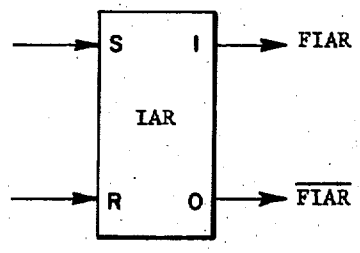
FARC = $\overline{\text{FCCA}}$ FCCB $\overline{\text{FCCC}}$ QCPB

$\overline{\text{FARC}}$ = DT00 QCPB + DRES



FIAR = DMWS QCPA

$\overline{\text{FIAR}}$ = (DT01 + DT08) QCPA + DRES

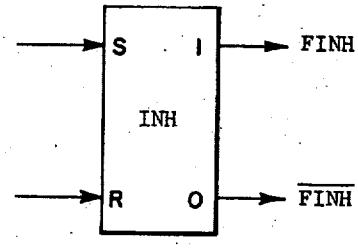


MEMORY

FIG. 32.

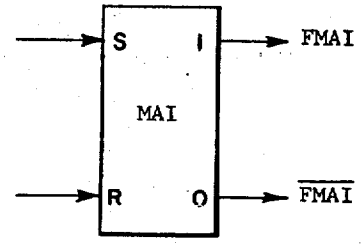
$FINH = (FCCA\ FCCB\ FCCC\ FWEN + DRES)\ QCPB$

$\overline{FINH} = \overline{FWRB}\ DT13 + DRES$



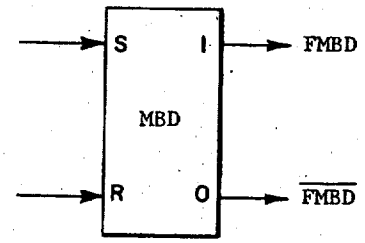
$FMAI = DIAD\ \overline{FIAR}\ \overline{FMBZ}\ \overline{FMBD}\ (FMWR + FMRD)$

$\overline{FMAI} = (\overline{DIAD} + FIAR)\ (FMWR + FMRD) + DRES$



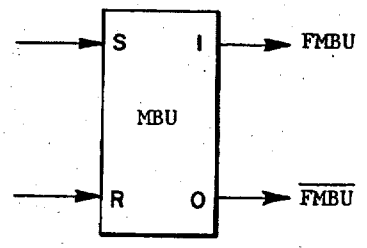
$FMBD = DT00\ QCPB$

$\overline{FMBD} = \overline{FMBZ} + DRES$



$FMBU = FCMA\ \overline{FARC}$

$\overline{FMBU} = \overline{FRDX}\ \overline{FWEN}\ FARC + FCCA\ FCCC\ FCCD\ \overline{FCMB} + DBRP$

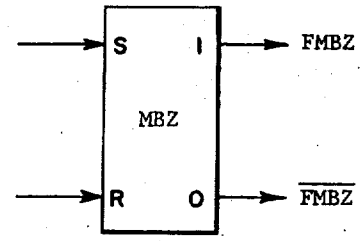


MEMORY

FIG.33.

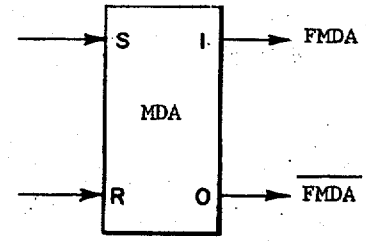
FMBZ = DMST $\overline{\text{FMBZ}}$ QCPC

$\overline{\text{FMBZ}}$ = FMBD QCPC + DRES



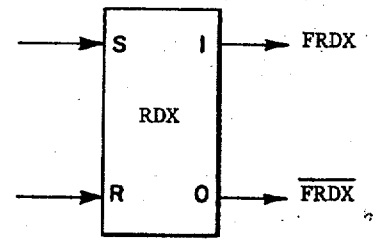
FMDA = $\overline{\text{FCCB}}$ FCCC $\overline{\text{FCCD}}$ $\overline{\text{DAMR}}$ QCPB

$\overline{\text{FMDA}}$ = $\overline{\text{FCCA}}$ FCCB FCCC $\overline{\text{FCCD}}$ QCPB + DRES



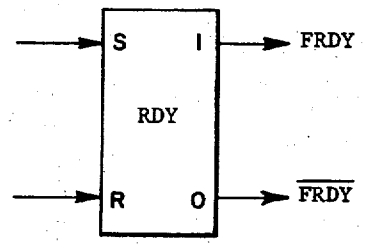
FRDX = DT00 QCPA

$\overline{\text{FRDX}}$ = DT07 $\overline{\text{FCMB}}$ + DT00 DRES



FRDY = DT01 QCPB

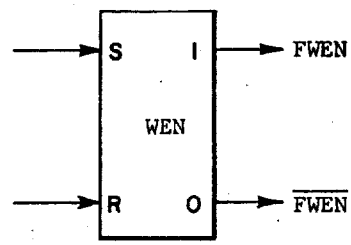
$\overline{\text{FRDY}}$ = (DT07 + DRES) $\overline{\text{FCMB}}$



MEMORY
FIG. 34.

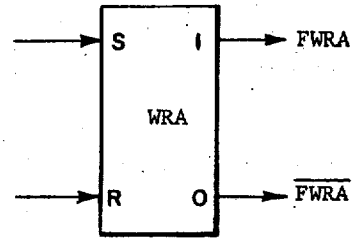
FWEN = FMWR $\overline{DT07}$ $\overline{FCCD}$ QCPA

$\overline{FWEN}$ = FMBD + DRES



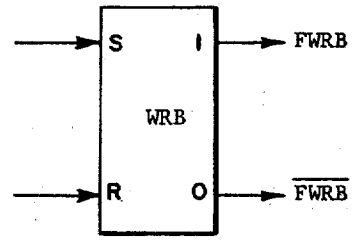
FWRA = (FCCA FCCB FCCC FWEN + DRES) QCPB

$\overline{FWRA}$ = DT00 QCPB + DRES



FWRB = (FCCA FCCB FCCC FWEN + DRES) QCPA

$\overline{FWRB}$ = DT13 QCPB + DRES



MEMORY
FIG. 35.

LOGICAL COMBINATION SIGNALS:

FARC DAMR FRDX	= CSTR
$\overline{\text{FMBZ}}$ $\overline{\text{FIAR}}$	= DAMC
$\overline{\text{FIAR}}$ FMBD + DRES	= DBRP
FCCA $\overline{\text{FCCB}}$ $\overline{\text{FCCC}}$ $\overline{\text{FCCD}}$ $\overline{\text{FRDY}}$ + FCCA FCCB FCCC $\overline{\text{FINH}}$	= DBRR
DBRR $\overline{\text{FARC}}$ $\overline{\text{FIAR}}$	= DBRS
$\overline{\text{FCCA}}$ FCCB $\overline{\text{FCCC}}$ $\overline{\text{FCCD}}$ FRDY	= DCLM
DB12 + DB13 + DB14 (4K SYSTEM)	= DIAD
DB13 + DB14 (8K SYSTEM)	= DIAD
DB14 (16K SYSTEM)	= DIAD
NO INPUTS (DISABLED FOR 32K SYSTEM)	= DIAD
FB12 FB13 FINH	= DIN1
$\overline{\text{FB12}}$ FB13 FINH	= DIN2
FB12 $\overline{\text{FB13}}$ FINH	= DIN3
$\overline{\text{FB12}}$ $\overline{\text{FB13}}$ FINH	= DIN4
CSTR + FMDA	= DMDA
FMRD + FMWR	= DMRW
$\overline{\text{FMAI}}$ ($\overline{\text{DIAD}}$ + FIAR) (FMWR + FMRD) DAMC	= DMST
FMWR $\overline{\text{FMRD}}$ DT00	= DMWT
FCCA FCCB FCCC FWEN + DRES	= DP07
DICR	= DRES
$\overline{\text{FCCA}}$ $\overline{\text{FCCB}}$ $\overline{\text{FCCC}}$ $\overline{\text{FCCD}}$	= DT00
FCCA $\overline{\text{FCCB}}$ $\overline{\text{FCCC}}$ $\overline{\text{FCCD}}$	= DT01
FCCA FCCB FCCC	= DT07
$\overline{\text{FCCA}}$ $\overline{\text{FCCB}}$ $\overline{\text{FCCC}}$ FCCD	= DT08
FCCA $\overline{\text{FCCB}}$ FCCC FCCD	= DT13

MEMORY
FIG.36.

LOGICAL COMBINATION SIGNALS (Cont.):

DB14 $\overline{\text{FIAR}}$ + $\overline{\text{FB14}}$ FIAR (Upper 16K)	= DULS
$\overline{\text{DB14}}$ $\overline{\text{FIAR}}$ + FB14 FIAR (Lower 16K)	= DULS
($\overline{\text{FCMA}}$ $\overline{\text{FCMB}}$ + FCMA FCMB) $\overline{\text{FMBD}}$	= QCGA
(FCMA $\overline{\text{FCMB}}$ + $\overline{\text{FCMA}}$ FCMB)	= QCGB

ADDRESS DECODE AND DRIVER INPUT SIGNALS:

Select X-Line Signals - Read:

FB00 FB01 FB02 FRDX	= $\overline{\text{XX01}}$
$\overline{\text{FB00}}$ FB01 FB02 FRDX	= $\overline{\text{XX02}}$
FB00 $\overline{\text{FB01}}$ FB02 FRDX	= $\overline{\text{XX03}}$
$\overline{\text{FB00}}$ $\overline{\text{FB01}}$ FB02 FRDX	= $\overline{\text{XX04}}$
FB00 FB01 $\overline{\text{FB02}}$ FRDX	= $\overline{\text{XX05}}$
$\overline{\text{FB00}}$ FB01 $\overline{\text{FB02}}$ FRDX	= $\overline{\text{XX06}}$
FB00 $\overline{\text{FB01}}$ $\overline{\text{FB02}}$ FRDX	= $\overline{\text{XX07}}$
$\overline{\text{FB00}}$ $\overline{\text{FB01}}$ $\overline{\text{FB02}}$ FRDX	= $\overline{\text{XX08}}$

Select X-Line Signals - Write:

FB00 FB01 FB02 FINH	= XX01
$\overline{\text{FB00}}$ FB01 FB02 FINH	= XX02
FB00 $\overline{\text{FB01}}$ FB02 FINH	= XX03
$\overline{\text{FB00}}$ $\overline{\text{FB01}}$ FB02 FINH	= XX04
FB00 FB01 $\overline{\text{FB02}}$ FINH	= XX05
$\overline{\text{FB00}}$ FB01 $\overline{\text{FB02}}$ FINH	= XX06
FB00 $\overline{\text{FB01}}$ $\overline{\text{FB02}}$ FINH	= XX07
$\overline{\text{FB00}}$ $\overline{\text{FB01}}$ $\overline{\text{FB02}}$ FINH	= XX08

MEMORY
FIG.37

ADDRESS DECODE AND DRIVER INPUT SIGNALS (Cont.):

Select X-Group Signals - Stacks #1 and #3:

FB03 FB04 FB05 FB12 (FRDX + FWRA) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$ = $\overline{\text{XXB1}}$

$\overline{\text{FB03}}$ FB04 FB05 FB12 (FRDX + FWRA) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$ = $\overline{\text{XXB2}}$

FB03 $\overline{\text{FB04}}$ FB05 FB12 (FRDX + FWRA) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$ = $\overline{\text{XXB3}}$

$\overline{\text{FB03}}$ $\overline{\text{FB04}}$ FB05 FB12 (FRDX + FWRA) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$ = $\overline{\text{XXB4}}$

FB03 FB04 $\overline{\text{FB05}}$ FB12 (FRDX + FWRA) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$ = $\overline{\text{XXB5}}$

$\overline{\text{FB03}}$ FB04 $\overline{\text{FB05}}$ FB12 (FRDX + FWRA) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$ = $\overline{\text{XXB6}}$

FB03 $\overline{\text{FB04}}$ $\overline{\text{FB05}}$ FB12 (FRDX + FWRA) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$ = $\overline{\text{XXB7}}$

$\overline{\text{FB03}}$ $\overline{\text{FB04}}$ $\overline{\text{FB05}}$ FB12 (FRDX + FWRA) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$ = $\overline{\text{XXB8}}$

Select X-Group Signals - Stacks #2 and #4:

FB03 FB04 FB05 $\overline{\text{FB12}}$ (FRDX + FWRA) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$ = XXB1

$\overline{\text{FB03}}$ FB04 FB05 $\overline{\text{FB12}}$ (FRDX + FWRA) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$ = XXB2

FB03 $\overline{\text{FB04}}$ FB05 $\overline{\text{FB12}}$ (FRDX + FWRA) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$ = XXB3

$\overline{\text{FB03}}$ $\overline{\text{FB04}}$ FB05 $\overline{\text{FB12}}$ (FRDX + FWRA) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$ = XXB4

FB03 FB04 $\overline{\text{FB05}}$ $\overline{\text{FB12}}$ (FRDX + FWRA) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$ = XXB5

$\overline{\text{FB03}}$ FB04 $\overline{\text{FB05}}$ $\overline{\text{FB12}}$ (FRDX + FWRA) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$ = XXB6

FB03 $\overline{\text{FB04}}$ $\overline{\text{FB05}}$ $\overline{\text{FB12}}$ (FRDX + FWRA) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$ = XXB7

$\overline{\text{FB03}}$ $\overline{\text{FB04}}$ $\overline{\text{FB05}}$ $\overline{\text{FB12}}$ (FRDX + FWRA) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$ = XXB8

MEMORY
FIG. 38.

ADDRESS DECODE AND DRIVER INPUT SIGNALS (Cont.):

Select Y-Line Signals - Read:

FB06 FB07 FB08 FRDY	= XY01
$\overline{\text{FB06}}$ FB07 FB08 FRDY	= XY02
FB06 $\overline{\text{FB07}}$ FB08 FRDY	= XY03
$\overline{\text{FB06}}$ $\overline{\text{FB07}}$ FB08 FRDY	= XY04
FB06 FB07 $\overline{\text{FB08}}$ FRDY	= XY05
$\overline{\text{FB06}}$ FB07 $\overline{\text{FB08}}$ FRDY	= XY06
FB06 $\overline{\text{FB07}}$ $\overline{\text{FB08}}$ FRDY	= XY07
$\overline{\text{FB06}}$ $\overline{\text{FB07}}$ $\overline{\text{FB08}}$ FRDY	= XY08

Select Y-Line Signals - Write:

FB06 FB07 FB08 FWRA	= $\overline{\text{XY01}}$
$\overline{\text{FB06}}$ FB07 FB08 FWRA	= $\overline{\text{XY02}}$
FB06 $\overline{\text{FB07}}$ FB08 FWRA	= $\overline{\text{XY03}}$
$\overline{\text{FB06}}$ $\overline{\text{FB07}}$ FB08 FWRA	= $\overline{\text{XY04}}$
FB06 FB07 $\overline{\text{FB08}}$ FWRA	= $\overline{\text{XY05}}$
$\overline{\text{FB06}}$ FB07 $\overline{\text{FB08}}$ FWRA	= $\overline{\text{XY06}}$
FB06 $\overline{\text{FB07}}$ $\overline{\text{FB08}}$ FWRA	= $\overline{\text{XY07}}$
$\overline{\text{FB06}}$ $\overline{\text{FB07}}$ $\overline{\text{FB08}}$ FWRA	= $\overline{\text{XY08}}$

MEMORY
FIG. 39.

ADDRESS DECODE AND DRIVER INPUT SIGNALS (Cont.):

Select Y-Group Signals - Stacks #1 and #2:

FB09 FB10 FB11 FB13 (FRDY + FWRB) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$	= $\overline{\text{XYB1}}$
$\overline{\text{FB09}}$ FB10 FB11 FB13 (FRDY + FWRB) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$	= $\overline{\text{XYB2}}$
FB09 $\overline{\text{FB10}}$ FB11 FB13 (FRDY + FWRB) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$	= $\overline{\text{XYB3}}$
$\overline{\text{FB09}}$ $\overline{\text{FB10}}$ FB11 FB13 (FRDY + FWRB) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$	= $\overline{\text{XYB4}}$
FB09 FB10 $\overline{\text{FB11}}$ FB13 (FRDY + FWRB) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$	= $\overline{\text{XYB5}}$
$\overline{\text{FB09}}$ FB10 $\overline{\text{FB11}}$ FB13 (FRDY + FWRB) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$	= $\overline{\text{XYB6}}$
FB09 $\overline{\text{FB10}}$ $\overline{\text{FB11}}$ FB13 (FRDY + FWRB) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$	= $\overline{\text{XYB7}}$
$\overline{\text{FB09}}$ $\overline{\text{FB10}}$ $\overline{\text{FB11}}$ FB13 (FRDY + FWRB) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$	= $\overline{\text{XYB8}}$

Select Y-Group Signals - Stacks #3 and #4:

FB09 FB10 FB11 $\overline{\text{FB13}}$ (FRDY + FWRB) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$	= XYB1
$\overline{\text{FB09}}$ FB10 FB11 $\overline{\text{FB13}}$ (FRDY + FWRB) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$	= XYB2
FB09 $\overline{\text{FB10}}$ FB11 $\overline{\text{FB13}}$ (FRDY + FWRB) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$	= XYB3
$\overline{\text{FB09}}$ $\overline{\text{FB10}}$ FB11 $\overline{\text{FB13}}$ (FRDY + FWRB) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$	= XYB4
FB09 FB10 $\overline{\text{FB11}}$ $\overline{\text{FB13}}$ (FRDY + FWRB) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$	= XYB5
$\overline{\text{FB09}}$ FB10 $\overline{\text{FB11}}$ $\overline{\text{FB13}}$ (FRDY + FWRB) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$	= XYB6
FB09 $\overline{\text{FB10}}$ $\overline{\text{FB11}}$ $\overline{\text{FB13}}$ (FRDY + FWRB) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$	= XYB7
$\overline{\text{FB09}}$ $\overline{\text{FB10}}$ $\overline{\text{FB11}}$ $\overline{\text{FB13}}$ (FRDY + FWRB) + $\overline{\text{FRDX}}$ $\overline{\text{FINH}}$	= XYB8

CLOCK GENERATOR INPUT SIGNALS:

QCPG $\overline{\text{DRES}}$ $\overline{\text{FMBD}}$	= QCPD
DMST $\overline{\text{FMBD}}$ $\overline{\text{FMBZ}}$ DULS + QCPD	= QCPG

GATED CLOCK AMPLIFIER INPUT SIGNALS:

QCPG QCGA	= QCPA
QCPG QCGB	= QCPB
QCPG	= QCPC

MEMORY
FIG. 40.

INHIBIT DRIVER INPUT SIGNALS:

Stack #1:		Stack #2:	
$\overline{FM00}$ DIN1	= Z100	$\overline{FM00}$ DIN2	= Z200
$\overline{FM01}$ DIN1	= Z101	$\overline{FM01}$ DIN2	= Z201
$\overline{FM02}$ DIN1	= Z102	$\overline{FM02}$ DIN2	= Z202
$\overline{FM03}$ DIN1	= Z103	$\overline{FM03}$ DIN2	= Z203
$\overline{FM04}$ DIN1	= Z104	$\overline{FM04}$ DIN2	= Z204
$\overline{FM05}$ DIN1	= Z105	$\overline{FM05}$ DIN2	= Z205
$\overline{FM06}$ DIN1	= Z106	$\overline{FM06}$ DIN2	= Z206
$\overline{FM07}$ DIN1	= Z107	$\overline{FM07}$ DIN2	= Z207
$\overline{FM08}$ DIN1	= Z108	$\overline{FM08}$ DIN2	= Z208
$\overline{FM09}$ DIN1	= Z109	$\overline{FM09}$ DIN2	= Z209
$\overline{FM10}$ DIN1	= Z110	$\overline{FM10}$ DIN2	= Z210
$\overline{FM11}$ DIN1	= Z111	$\overline{FM11}$ DIN2	= Z211
$\overline{FM12}$ DIN1	= Z112	$\overline{FM12}$ DIN2	= Z212
$\overline{FM13}$ DIN1	= Z113	$\overline{FM13}$ DIN2	= Z213
$\overline{FM14}$ DIN1	= Z114	$\overline{FM14}$ DIN2	= Z214
$\overline{FM15}$ DIN1	= Z115	$\overline{FM15}$ DIN2	= Z215
$\overline{FM16}$ DIN1	= Z116	$\overline{FM16}$ DIN2	= Z216
$\overline{FM17}$ DIN1	= Z117	$\overline{FM17}$ DIN2	= Z217
$\overline{FM18}$ DIN1	= Z118	$\overline{FM18}$ DIN2	= Z218
$\overline{FM19}$ DIN1	= Z119	$\overline{FM19}$ DIN2	= Z219
$\overline{FM20}$ DIN1	= Z120	$\overline{FM20}$ DIN2	= Z220
$\overline{FM21}$ DIN1	= Z121	$\overline{FM21}$ DIN2	= Z221
$\overline{FM22}$ DIN1	= Z122	$\overline{FM22}$ DIN2	= Z222
$\overline{FM23}$ DIN1	= Z123	$\overline{FM23}$ DIN2	= Z223
$\overline{DM24}$ DIN1	= Z124	$\overline{DM24}$ DIN2	= Z224

MEMORY

FIG. 41.

INHIBIT DRIVER INPUT SIGNALS (Cont.):

Stack #3:		Stack #4:	
$\overline{FM00}$ DIN3	= Z300	$\overline{FM00}$ DIN4	= Z400
$\overline{FM01}$ DIN3	= Z301	$\overline{FM01}$ DIN4	= Z401
$\overline{FM02}$ DIN3	= Z302	$\overline{FM02}$ DIN4	= Z402
$\overline{FM03}$ DIN3	= Z303	$\overline{FM03}$ DIN4	= Z403
$\overline{FM04}$ DIN3	= Z304	$\overline{FM04}$ DIN4	= Z404
$\overline{FM05}$ DIN3	= Z305	$\overline{FM05}$ DIN4	= Z405
$\overline{FM06}$ DIN3	= Z306	$\overline{FM06}$ DIN4	= Z406
$\overline{FM07}$ DIN3	= Z307	$\overline{FM07}$ DIN4	= Z407
$\overline{FM08}$ DIN3	= Z308	$\overline{FM08}$ DIN4	= Z408
$\overline{FM09}$ DIN3	= Z309	$\overline{FM09}$ DIN4	= Z409
$\overline{FM10}$ DIN3	= Z310	$\overline{FM10}$ DIN4	= Z410
$\overline{FM11}$ DIN3	= Z311	$\overline{FM11}$ DIN4	= Z411
$\overline{FM12}$ DIN3	= Z312	$\overline{FM12}$ DIN4	= Z412
$\overline{FM13}$ DIN3	= Z313	$\overline{FM13}$ DIN4	= Z413
$\overline{FM14}$ DIN3	= Z314	$\overline{FM14}$ DIN4	= Z414
$\overline{FM15}$ DIN3	= Z315	$\overline{FM15}$ DIN4	= Z415
$\overline{FM16}$ DIN3	= Z316	$\overline{FM16}$ DIN4	= Z416
$\overline{FM17}$ DIN3	= Z317	$\overline{FM17}$ DIN4	= Z417
$\overline{FM18}$ DIN3	= Z318	$\overline{FM18}$ DIN4	= Z418
$\overline{FM19}$ DIN3	= Z319	$\overline{FM19}$ DIN4	= Z419
$\overline{FM20}$ DIN3	= Z320	$\overline{FM20}$ DIN4	= Z420
$\overline{FM21}$ DIN3	= Z321	$\overline{FM21}$ DIN4	= Z421
$\overline{FM22}$ DIN3	= Z322	$\overline{FM22}$ DIN4	= Z422
$\overline{FM23}$ DIN3	= Z323	$\overline{FM23}$ DIN4	= Z423
$\overline{DM24}$ DIN3	= Z324	$\overline{DM24}$ DIN4	= Z424

MEMORY
FIG. 42.

DATA DRIVER INPUT SIGNALS:

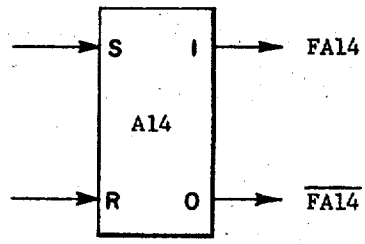
(W100 + W200 + W300 + W400) CSTR	= WD00
(W101 + W201 + W301 + W401) CSTR	= WD01
(W102 + W202 + W302 + W402) CSTR	= WD02
(W103 + W203 + W303 + W403) CSTR	= WD03
(W104 + W204 + W304 + W404) CSTR	= WD04
(W105 + W205 + W305 + W405) CSTR	= WD05
(W106 + W206 + W306 + W406) CSTR	= WD06
(W107 + W207 + W307 + W407) CSTR	= WD07
(W108 + W208 + W308 + W408) CSTR	= WD08
(W109 + W209 + W309 + W409) CSTR	= WD09
(W110 + W210 + W310 + W410) CSTR	= WD10
(W111 + W211 + W311 + W411) CSTR	= WD11
(W112 + W212 + W312 + W412) CSTR	= WD12
(W113 + W213 + W313 + W413) CSTR	= WD13
(W114 + W214 + W314 + W414) CSTR	= WD14
(W115 + W215 + W315 + W415) CSTR	= WD15
(W116 + W216 + W316 + W416) CSTR	= WD16
(W117 + W217 + W317 + W417) CSTR	= WD17
(W118 + W218 + W318 + W418) CSTR	= WD18
(W119 + W219 + W319 + W419) CSTR	= WD19
(W120 + W220 + W320 + W420) CSTR	= WD20
(W121 + W221 + W321 + W421) CSTR	= WD21
(W122 + W222 + W322 + W422) CSTR	= WD22
(W123 + W223 + W323 + W423) CSTR	= WD23
(W124 + W224 + W324 + W424) CSTR	= WD24

MEMORY

FIG. 43.

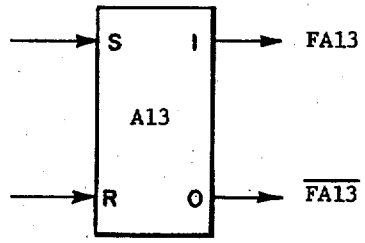
FA14 = FE14 QEAX

$\overline{FA14}$ = DCLA



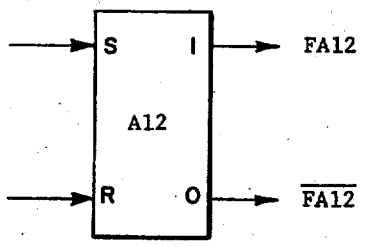
FA13 = FE13 QEAX

$\overline{FA13}$ = DCLA



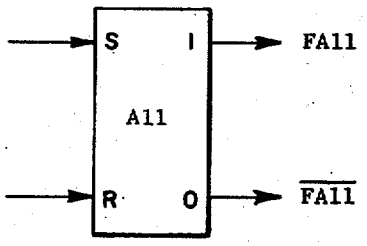
FA12 = FE12 QEAX

$\overline{FA12}$ = DCLA



FA11 = FE11 QEAX

$\overline{FA11}$ = DCLA

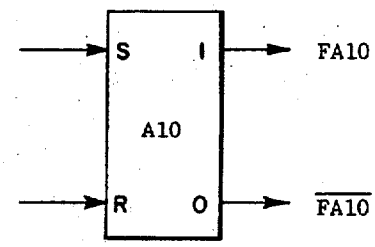


PROGRAM PROCESSOR

FIG. 44.

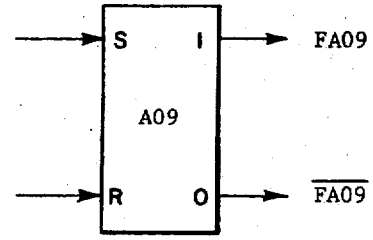
FA10 = FE10 QEAX

$\overline{FA10}$ = DCLA



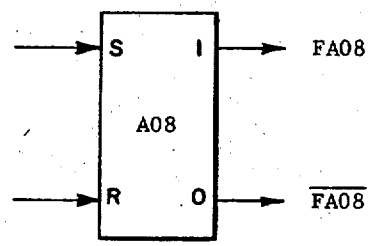
FA09 = FE09 QEAX

$\overline{FA09}$ = DCLA



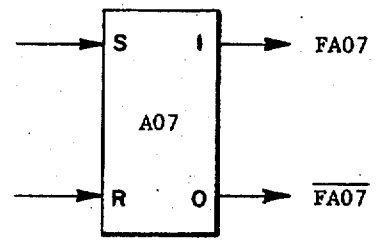
FA08 = FE08 QEAX

$\overline{FA08}$ = DCLA



FA07 = FE07 QEAX

$\overline{FA07}$ = DCLA

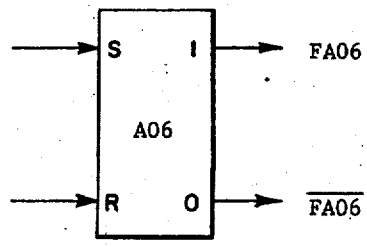


PROGRAM PROCESSOR

FIG. 45.

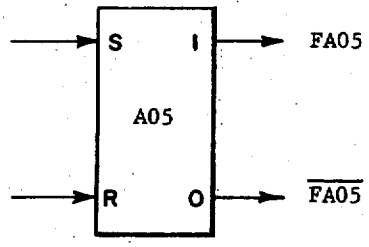
FA06 = FE06 QEAX

$\overline{FA06}$ = DCLA



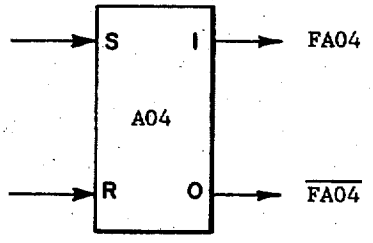
FA05 = FE05 QEAX

$\overline{FA05}$ = DCLA



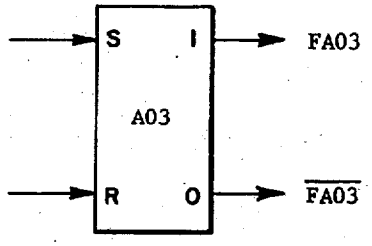
FA04 = FE04 QEAX

$\overline{FA04}$ = DCLA



FA03 = $\overline{FE03}$ QEAX

$\overline{FA03}$ = DCLA

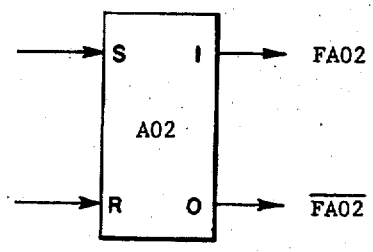


PROGRAM PROCESSOR

FIG.46.

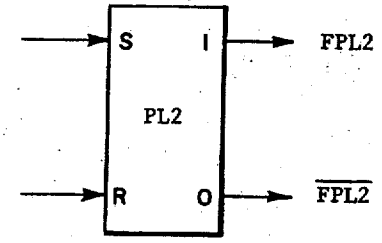
$FA02 = \overline{FE02}\ QEAX$

$\overline{FA02} = DCLA$



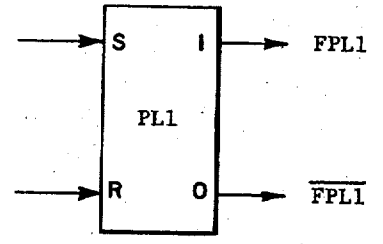
$FPL2 = (\overline{FIR1}\ DILX + DDLX\ DD10)\ QTCK + DEAP\ FE01$

$\overline{FPL2} = (FIR1\ DILX + DDLX\ \overline{FD10})\ QTCK + DEAP\ \overline{FE01}$



$FPL1 = \overline{FIR0}\ DILX\ QTCK + DEAP\ FE00 + DDLX\ DD09$

$\overline{FPL1} = FIR0\ DILX\ QTCK + DEAP\ \overline{FE00} + DDLX\ \overline{FD09}$

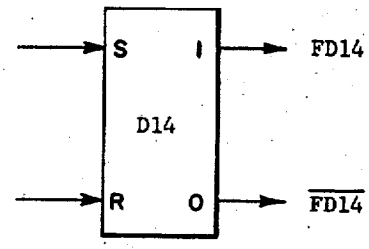


PROGRAM PROCESSOR

FIG. 47

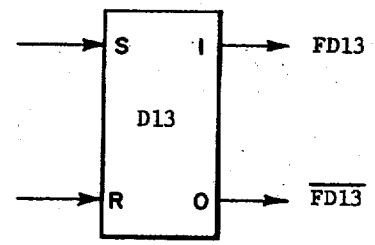
$FD14 = (FD13)\ \overline{DSDT}\ \overline{FD14} + QSDX\ \overline{MS14}$

$\overline{FD14} = (FD13)\ \overline{DSDT}\ FD14 + QSDX\ MS14$



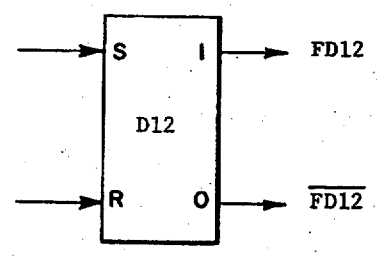
$FD13 = (FD12)\ \overline{DSDT}\ \overline{FD13} + QSDX\ \overline{MS13}$

$\overline{FD13} = (FD12)\ \overline{DSDT}\ FD13 + QSDX\ MS13$



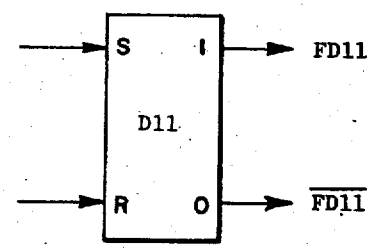
$FD12 = (FD11)\ \overline{DSDT}\ \overline{FD12} + QSDX\ \overline{MS12}$

$\overline{FD12} = (FD11)\ \overline{DSDT}\ FD12 + QSDX\ MS12$



$FD11 = (FD10)\ \overline{DSDT}\ \overline{FD11} + QSDX\ \overline{MS11}$

$\overline{FD11} = (FD10)\ \overline{DSDT}\ FD11 + QSDX\ MS11$

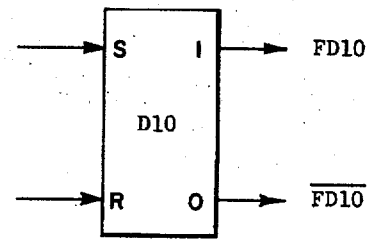


PROGRAM PROCESSOR

FIG.48.

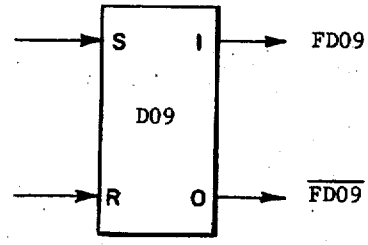
$FD10 = (FD09)\ \overline{DSDT}\ \overline{FD10} + QSDX\ \overline{MS10}$

$\overline{FD10} = (FD09)\ \overline{DSDT}\ FD10 + QSDX\ MS10$



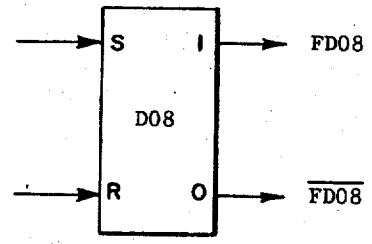
$FD09 = (FD08)\ \overline{DSDT}\ \overline{FD09} + QSDX\ \overline{MS09}$

$\overline{FD09} = (FD08)\ \overline{DSDT}\ FD09 + QSDX\ MS09$



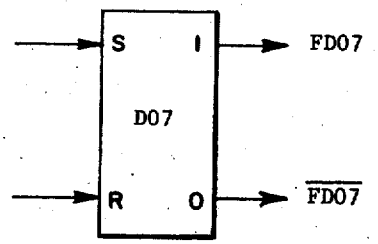
$FD08 = (FD07)\ \overline{DSDT}\ \overline{FD08} + QSDX\ \overline{MS08}$

$\overline{FD08} = (FD07)\ \overline{DSDT}\ FD08 + QSDX\ MS08$



$FD07 = (FD06)\ \overline{DSDT}\ \overline{FD07} + QSDX\ \overline{MS07}$

$\overline{FD07} = (FD06)\ \overline{DSDT}\ FD07 + QSDX\ MS07$

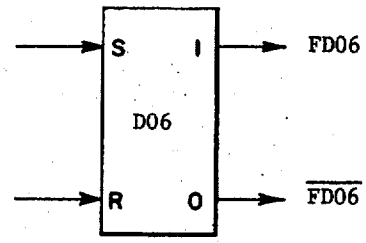


PROGRAM PROCESSOR

FIG. 49.

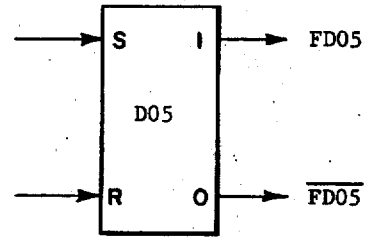
$FD06 = (FD05)\ \overline{DSDT}\ \overline{FD06} + QSDX\ \overline{MS06}$

$\overline{FD06} = (FD05)\ \overline{DSDT}\ FD06 + QSDX\ MS06$



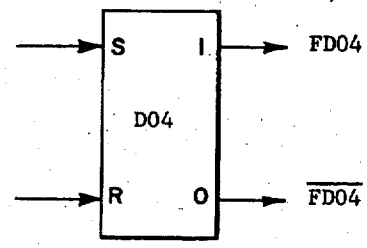
$FD05 = (FD04)\ \overline{DSDT}\ \overline{FD05} + QSDX\ \overline{MS05}$

$\overline{FD05} = (FD04)\ \overline{DSDT}\ FD05 + QSDX\ MS05$



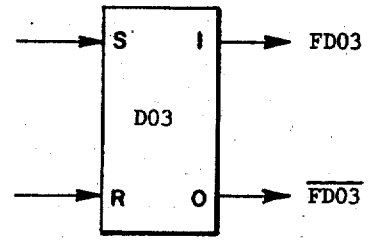
$FD04 = (FD03)\ \overline{DSDT}\ \overline{FD04} + QSDX\ \overline{MS04}$

$\overline{FD04} = (FD03)\ \overline{DSDT}\ FD04 + QSDX\ MS04$



$FD03 = (FD02)\ \overline{DSDT}\ \overline{FD03} + QSDX\ \overline{MS03}$

$\overline{FD03} = (FD02)\ \overline{DSDT}\ FD03 + QSDX\ MS03$

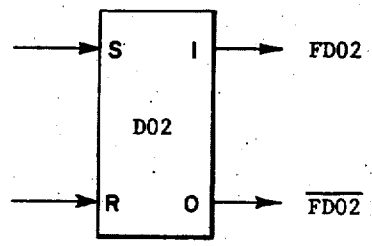


PROGRAM PROCESSOR

FIG.50.

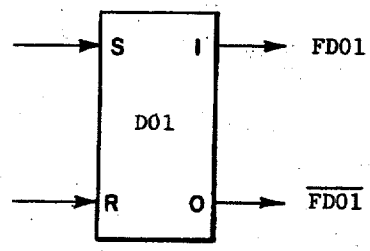
$FD02 = (FD01)\ \overline{DSDT}\ \overline{FD02} + QSDX\ \overline{MS02}$

$\overline{FD02} = (FD01)\ \overline{DSDT}\ FD02 + QSDX\ MS02$



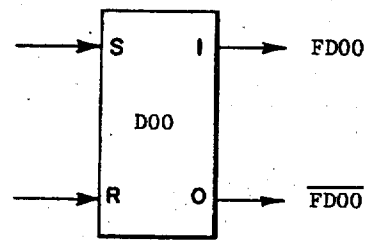
$FD01 = (FD00)\ \overline{DSDT}\ \overline{FD01} + QSDX\ \overline{MS01}$

$\overline{FD01} = (FD00)\ \overline{DSDT}\ FD01 + QSDX\ MS01$



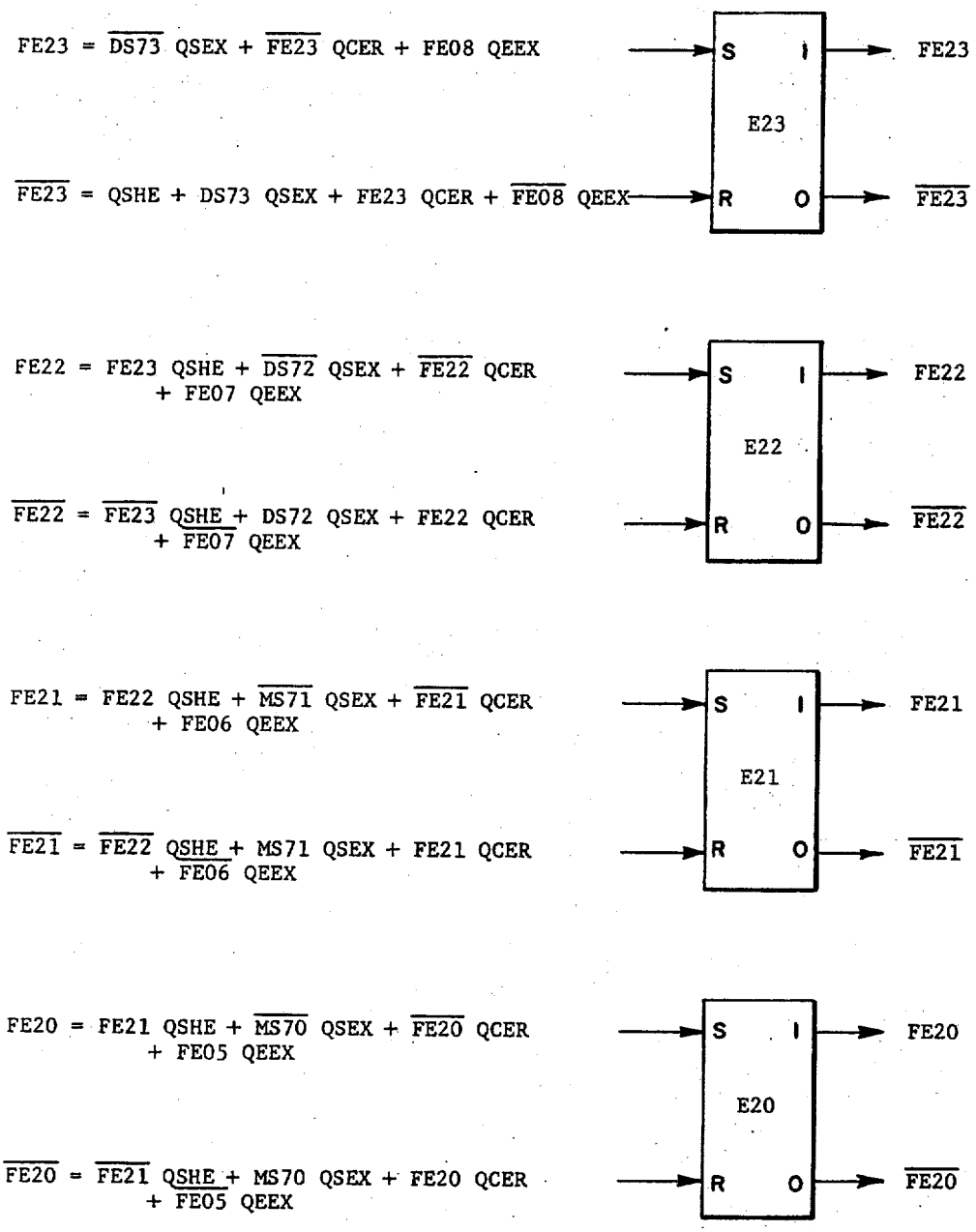
$FD00 = (DDCD)\ \overline{DSDT}\ \overline{FD00} + QSDX\ \overline{MS00}$

$\overline{FD00} = (DDCD)\ \overline{DSDT}\ FD00 + QSDX\ MS00$



PROGRAM PROCESSOR

FIG. 51.



PROGRAM PROCESSOR

FIG.52.

$FE19 = FE20\ QSHE + \overline{MS69}\ QSEX + \overline{FE19}\ QCER + FE04\ QEEX$

$\overline{FE19} = \overline{FE20}\ \underline{QSHE} + MS69\ QSEX + FE19\ QCER + FE04\ QEEX$

$FE18 = FE19\ QSHE + \overline{MS18}\ QSEX + \overline{FE18}\ QCER + FE03\ QEEX$

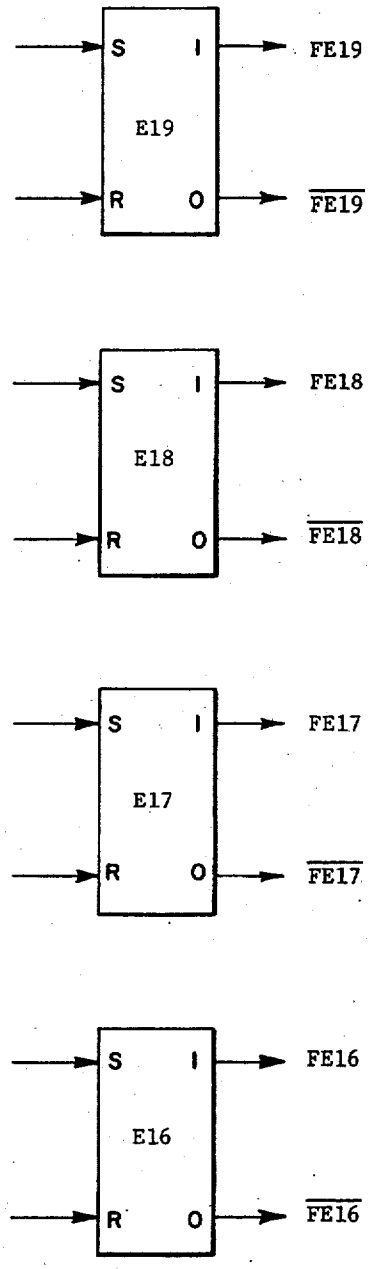
$\overline{FE18} = \overline{FE19}\ \underline{QSHE} + MS18\ QSEX + FE18\ QCER + FE03\ QEEX$

$FE17 = FE18\ QSHE + \overline{DS67}\ QSEX + \overline{FE17}\ QCER + FE02\ QEEX$

$\overline{FE17} = \overline{FE18}\ \underline{QSHE} + DS67\ QSEX + FE17\ QCER + FE02\ QEEX$

$FE16 = FE17\ QSHE + \overline{DS66}\ QSEX + \overline{FE16}\ QCER + FE01\ QEEX$

$\overline{FE16} = \overline{FE17}\ \underline{QSHE} + DS66\ QSEX + FE16\ QCER + \overline{FE01}\ QEEX$



PROGRAM PROCESSOR

FIG.53.

$$FE15 = FE16\ QSHE + \overline{MS65}\ QSEX + \overline{FE15}\ QCER + FE00\ QEEX$$

$$\overline{FE15} = \overline{FE16}\ QSHE + MS65\ QSEX + FE15\ QCER + FE00\ QEEX$$

$$FE14 = FE15\ QSHE + \overline{MS64}\ QSEX + \overline{FE14}\ QCER + FA14\ DAEX$$

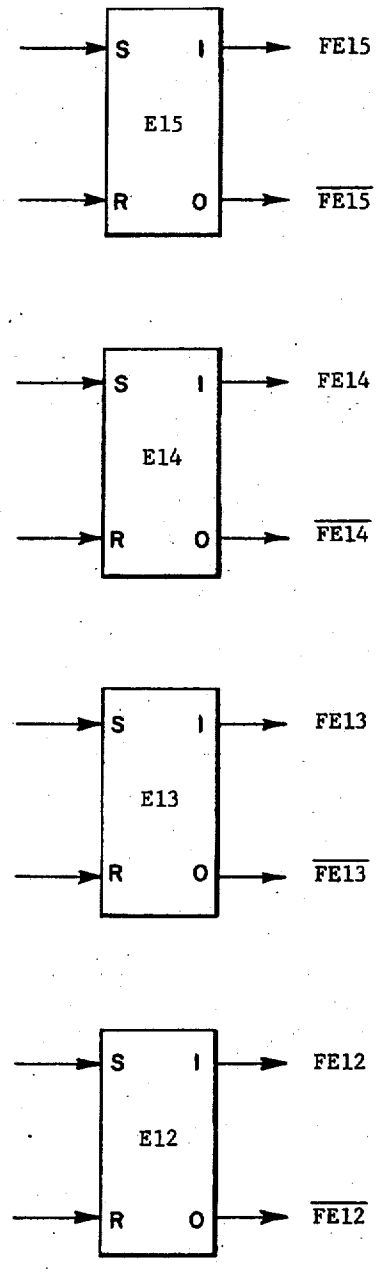
$$\overline{FE14} = \overline{FE15}\ QSHE + MS64\ QSEX + FE14\ QCER + FA14\ DAEX$$

$$FE13 = FE14\ QSHE + \overline{MS63}\ QSEX + \overline{FE13}\ QCER + FA13\ DAEX$$

$$\overline{FE13} = \overline{FE14}\ QSHE + MS63\ QSEX + FE13\ QCER + FA13\ DAEX$$

$$FE12 = FE13\ QSHE + \overline{MS12}\ QSEX + \overline{FE12}\ QCER + FA12\ DAEX$$

$$\overline{FE12} = \overline{FE13}\ QSHE + MS12\ QSEX + FE12\ QCER + FA12\ DAEX$$

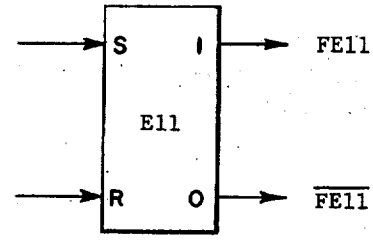


PROGRAM PROCESSOR

FIG. 54.

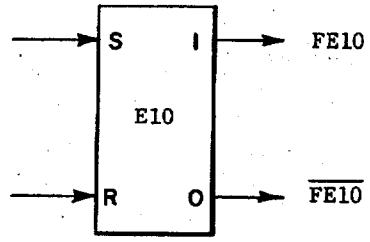
$$FE11 = FE12\ QSHE + \overline{DS61}\ QSEX + \overline{FE11}\ QCER + FA11\ DAEX$$

$$\overline{FE11} = \overline{FE12}\ \overline{QSHE} + DS61\ QSEX + FE11\ QCER + \overline{FA11}\ DAEX$$



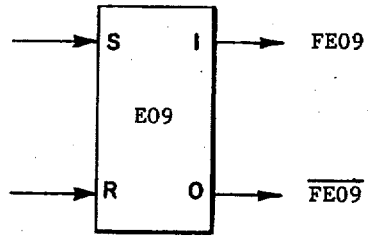
$$FE10 = FE11\ QSHE + \overline{DS60}\ QSEX + \overline{FE10}\ QCER + FA10\ DAEX$$

$$\overline{FE10} = \overline{FE11}\ \overline{QSHE} + DS60\ QSEX + FE10\ QCER + \overline{FA10}\ DAEX$$



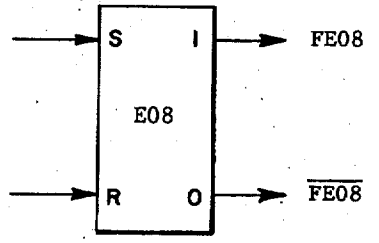
$$FE09 = FE10\ QSHE + \overline{MS59}\ QSEX + \overline{FE09}\ QCER + FA09\ DAEX$$

$$\overline{FE09} = \overline{FE10}\ \overline{QSHE} + MS59\ QSEX + FE09\ QCER + \overline{FA09}\ DAEX$$



$$FE08 = FE09\ QSHE + \overline{MS58}\ QSEX + \overline{FE08}\ QCER + FA08\ DAEX$$

$$\overline{FE08} = \overline{FE09}\ \overline{QSHE} + MS58\ QSEX + FE08\ QCER + \overline{FA08}\ DAEX$$

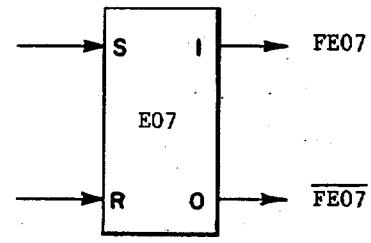


PROGRAM PROCESSOR

FIG. 55.

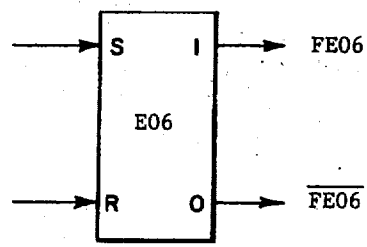
$$FE07 = FE08\ QSHE + \overline{MS57}\ QSEX + \overline{FE07}\ QCER + FA07\ DAEX$$

$$\overline{FE07} = \overline{FE08}\ \overline{QSHE} + MS57\ QSEX + FE07\ QCER + \overline{FA07}\ DAEX$$



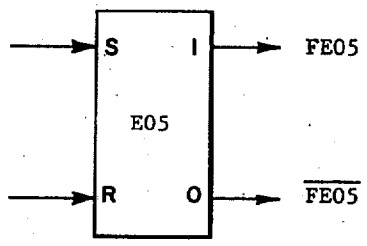
$$FE06 = FE07\ QSHE + \overline{MS06}\ QSEX + \overline{FE06}\ QCER + FA06\ DAEX$$

$$\overline{FE06} = \overline{FE07}\ \overline{QSHE} + MS06\ QSEX + FE06\ QCER + \overline{FA06}\ DAEX$$



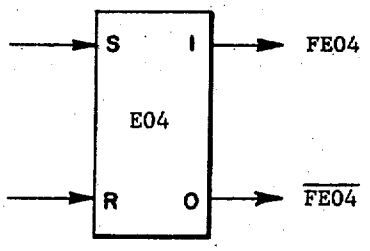
$$FE05 = FE06\ QSHE + \overline{DS55}\ QSEX + \overline{FE05}\ QCER + (DW11\ FTL3\ DMVL\ FMPL\ DBAC\ \overline{FLSN} + FA05\ DAEX)$$

$$\overline{FE05} = \overline{FE06}\ QSHE + DS55\ QSEX + FE05\ QCER + (DRE5 + \overline{FA05}\ DAEX)$$



$$FE04 = FE05\ QSHE + \overline{DS54}\ QSEX + \overline{FE04}\ QCER + FA04\ DAEX$$

$$\overline{FE04} = \overline{FE05}\ QSHE + DS54\ QSEX + FE04\ QCER + (DRE5 + \overline{FA04}\ DAEX)$$

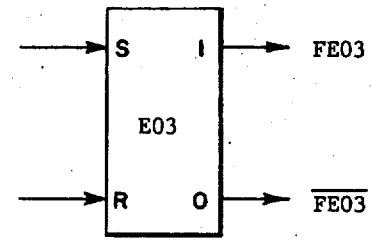


PROGRAM PROCESSOR

FIG. 56.

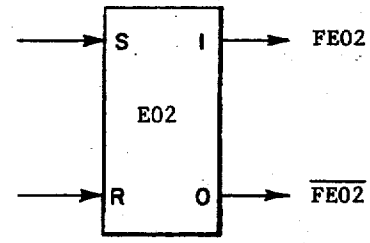
$$FE03 = FE04\ \underline{QSHE} + \overline{MS53}\ QSEX + \overline{FE03}\ QCER + \underline{FA03}\ DAEX$$

$$\overline{FE03} = \overline{FE04}\ QSHE + MS53\ QSEX + FE03\ QCER + FA03\ DAEX$$



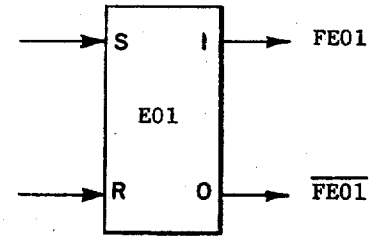
$$FE02 = FE03\ \underline{QSHE} + \overline{MS52}\ QSEX + \overline{FE02}\ QCER + \underline{FA02}\ DAEX$$

$$\overline{FE02} = \overline{FE03}\ QSHE + MS52\ QSEX + FE02\ QCER + FA02\ DAEX$$



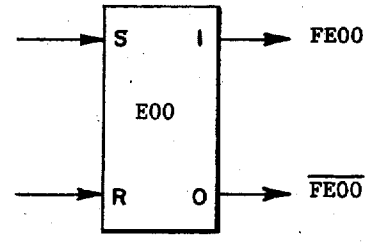
$$FE01 = FE02\ QSHE + \overline{MS51}\ QSEX + \overline{FE01}\ QCER + FPL2\ DAEX$$

$$\overline{FE01} = \overline{FE02}\ \underline{QSHE} + MS51\ QSEX + FE01\ QCER + \underline{FPL2}\ DAEX$$



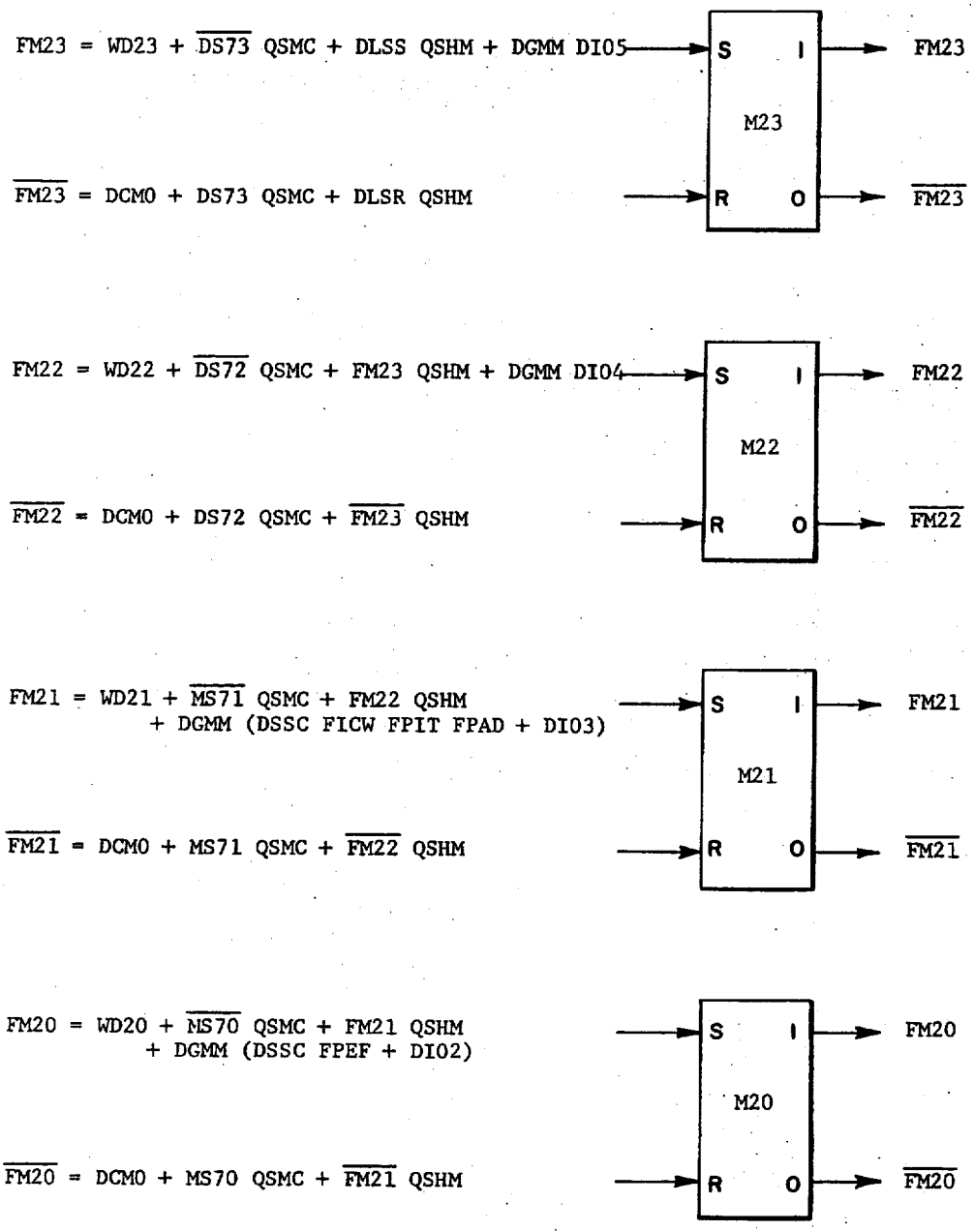
$$FE00 = FE01\ QSHE + \overline{MS00}\ QSEX + \overline{FE00}\ QCER + FPL1\ DAEX$$

$$\overline{FE00} = \overline{FE01}\ \underline{QSHE} + MS00\ QSEX + FE00\ QCER + \underline{FPL1}\ DAEX$$



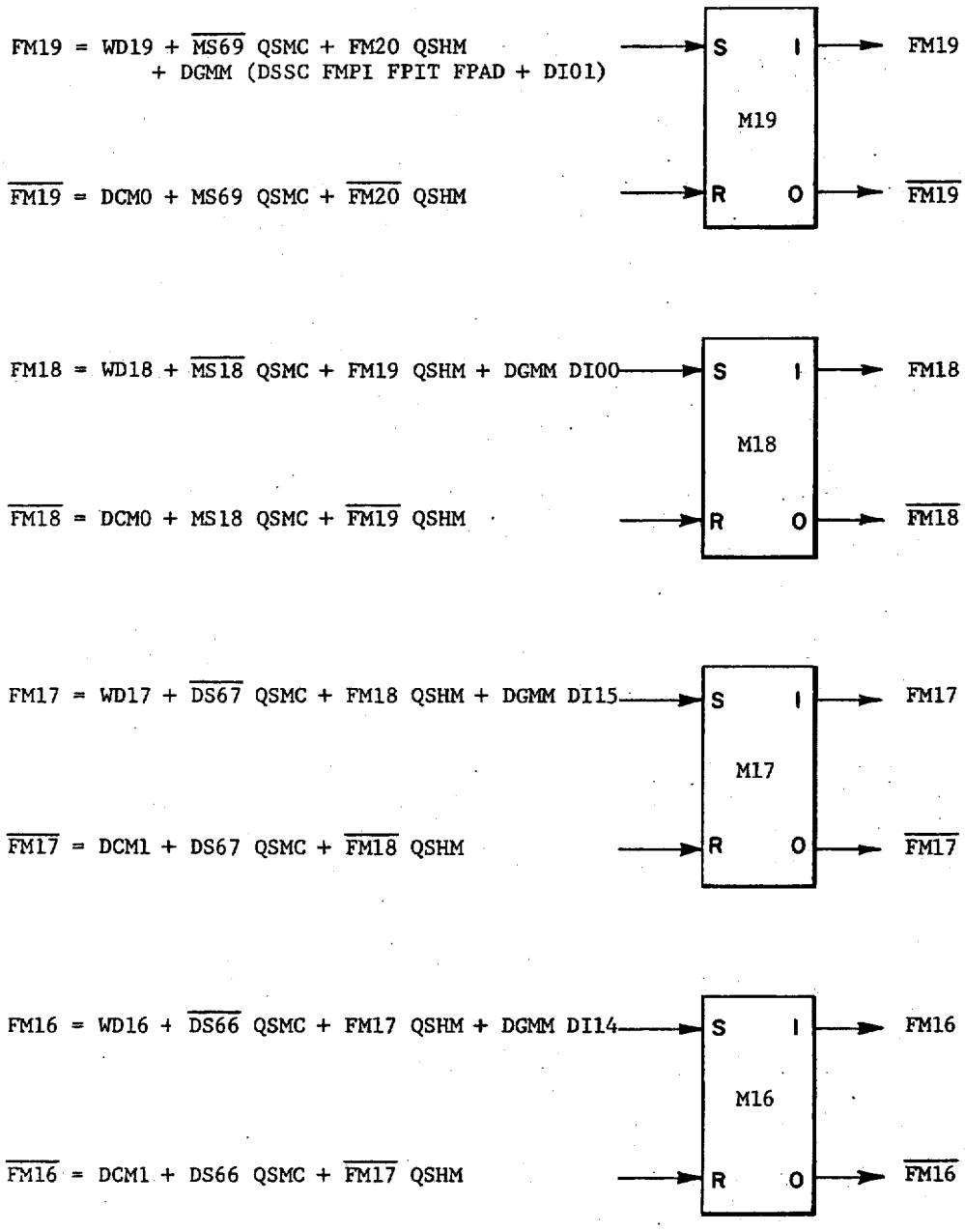
PROGRAM PROCESSOR

FIG. 57



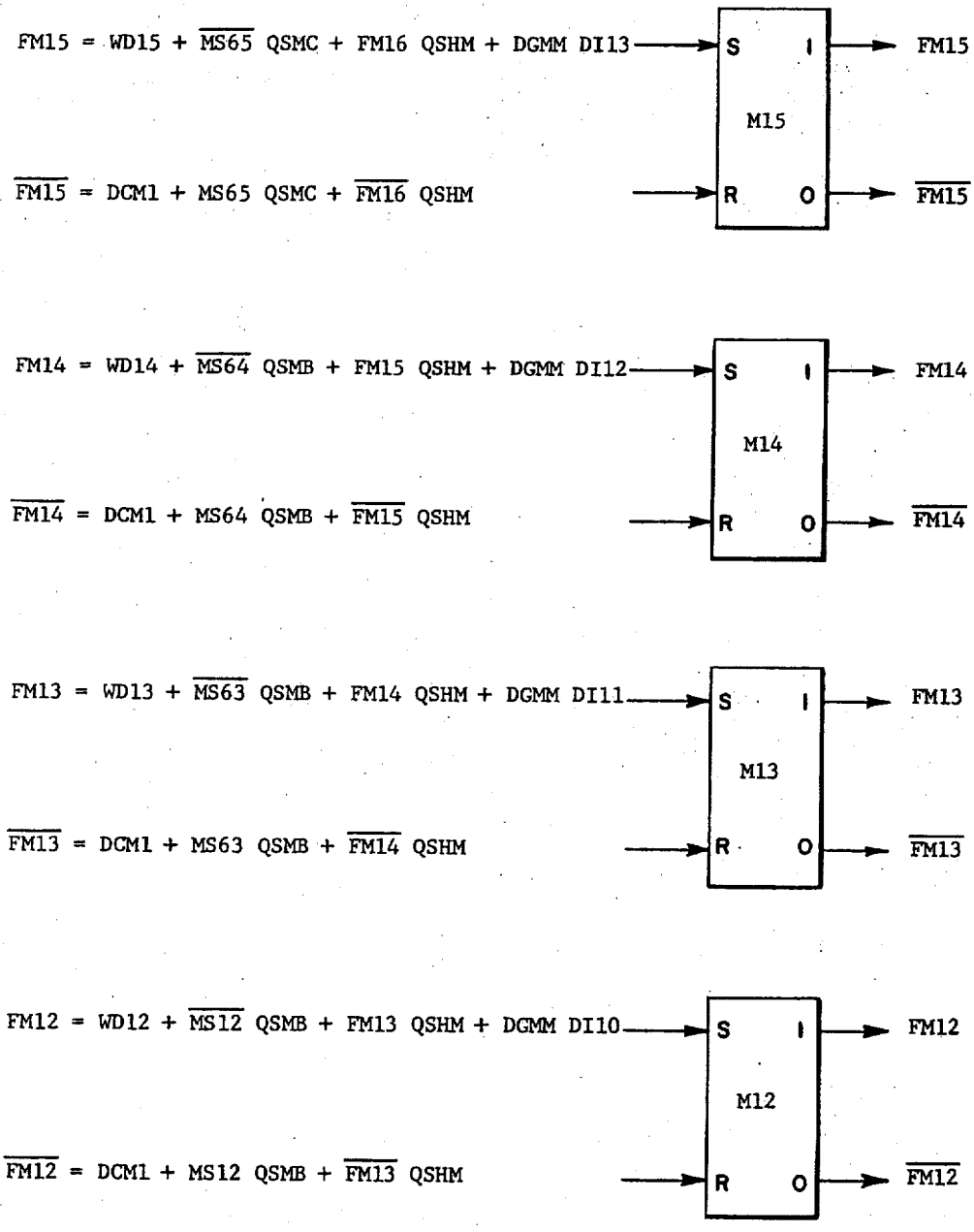
PROGRAM PROCESSOR

FIG.58.



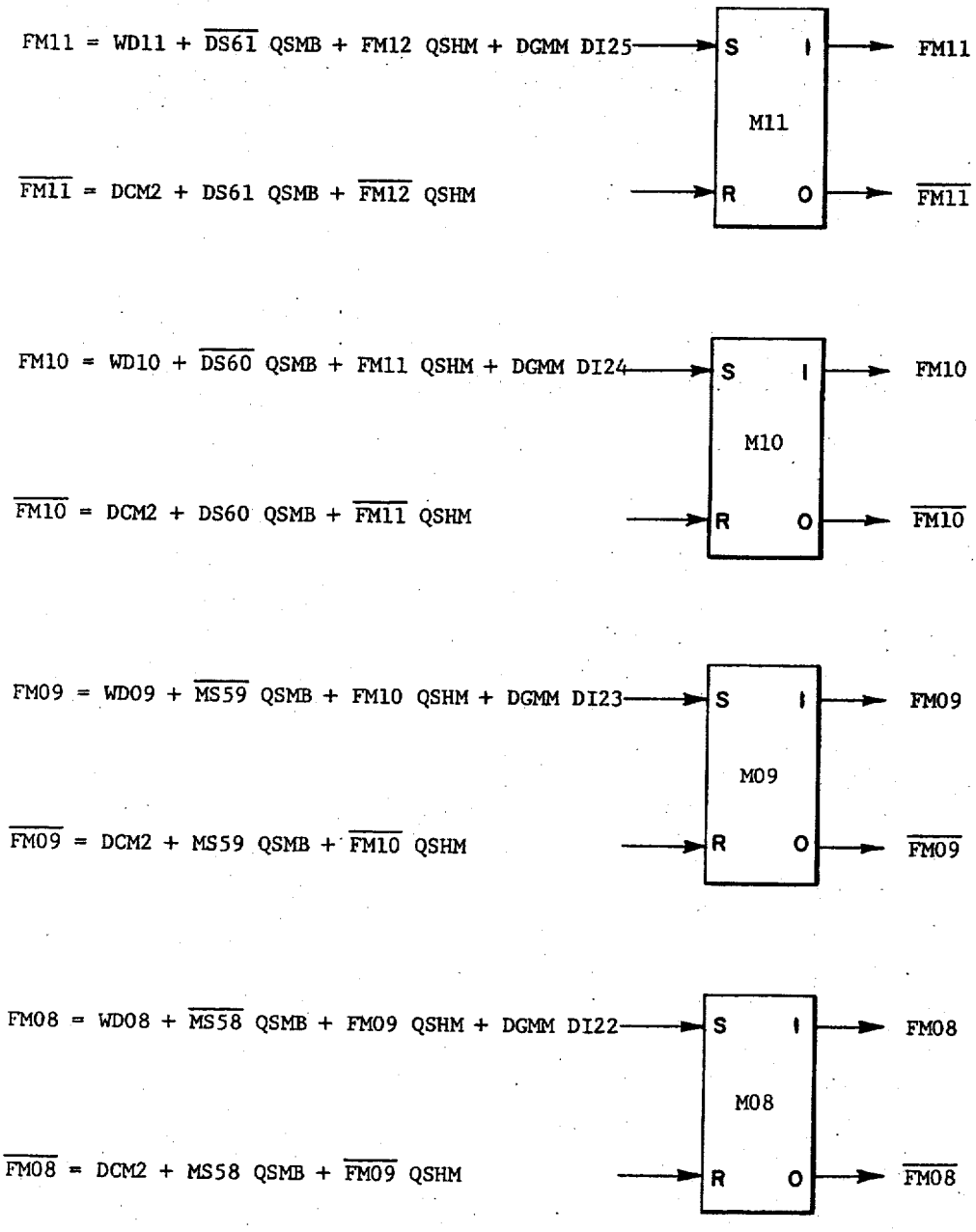
PROGRAM PROCESSOR

FIG.59.



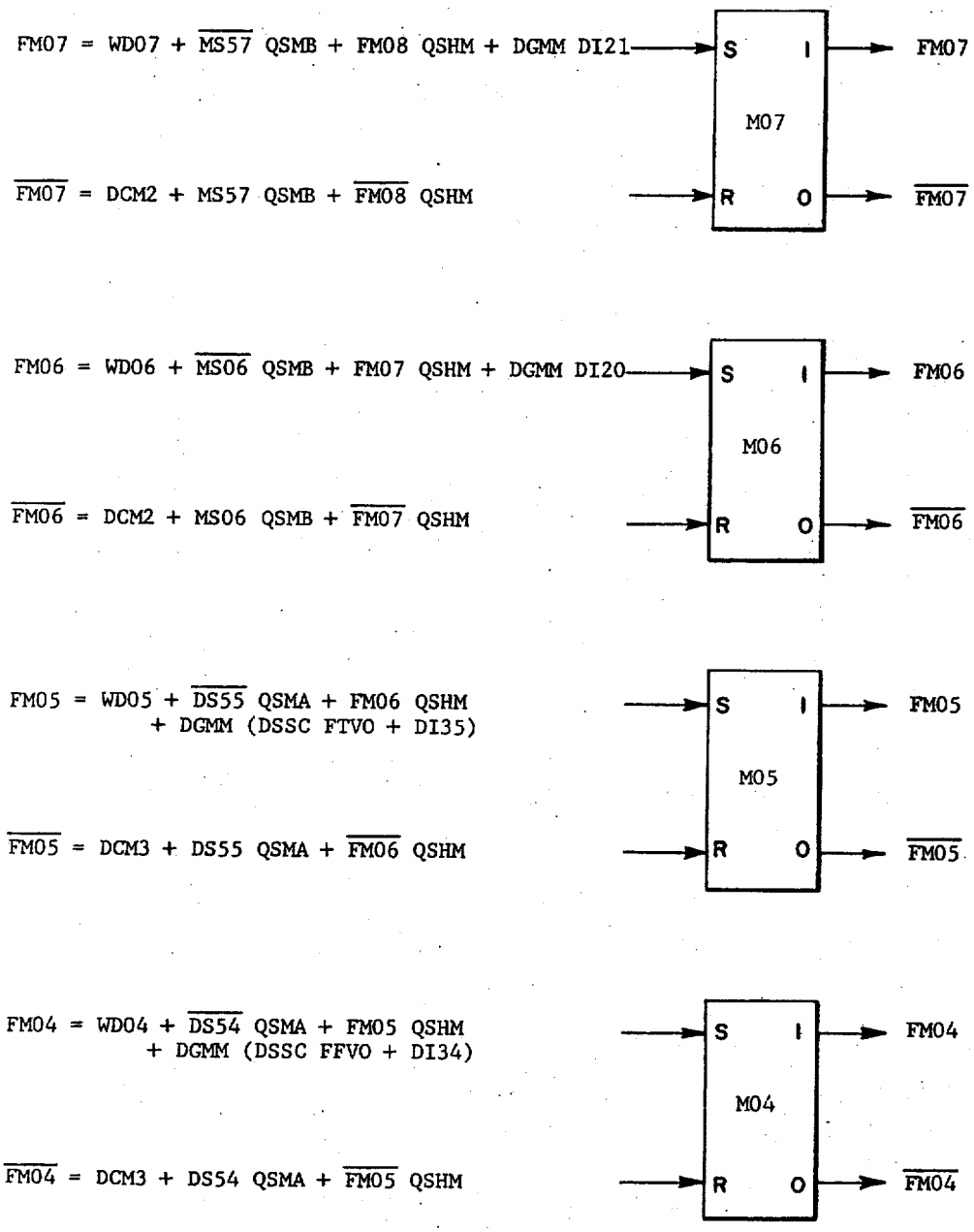
PROGRAM PROCESSOR

FIG. 60.



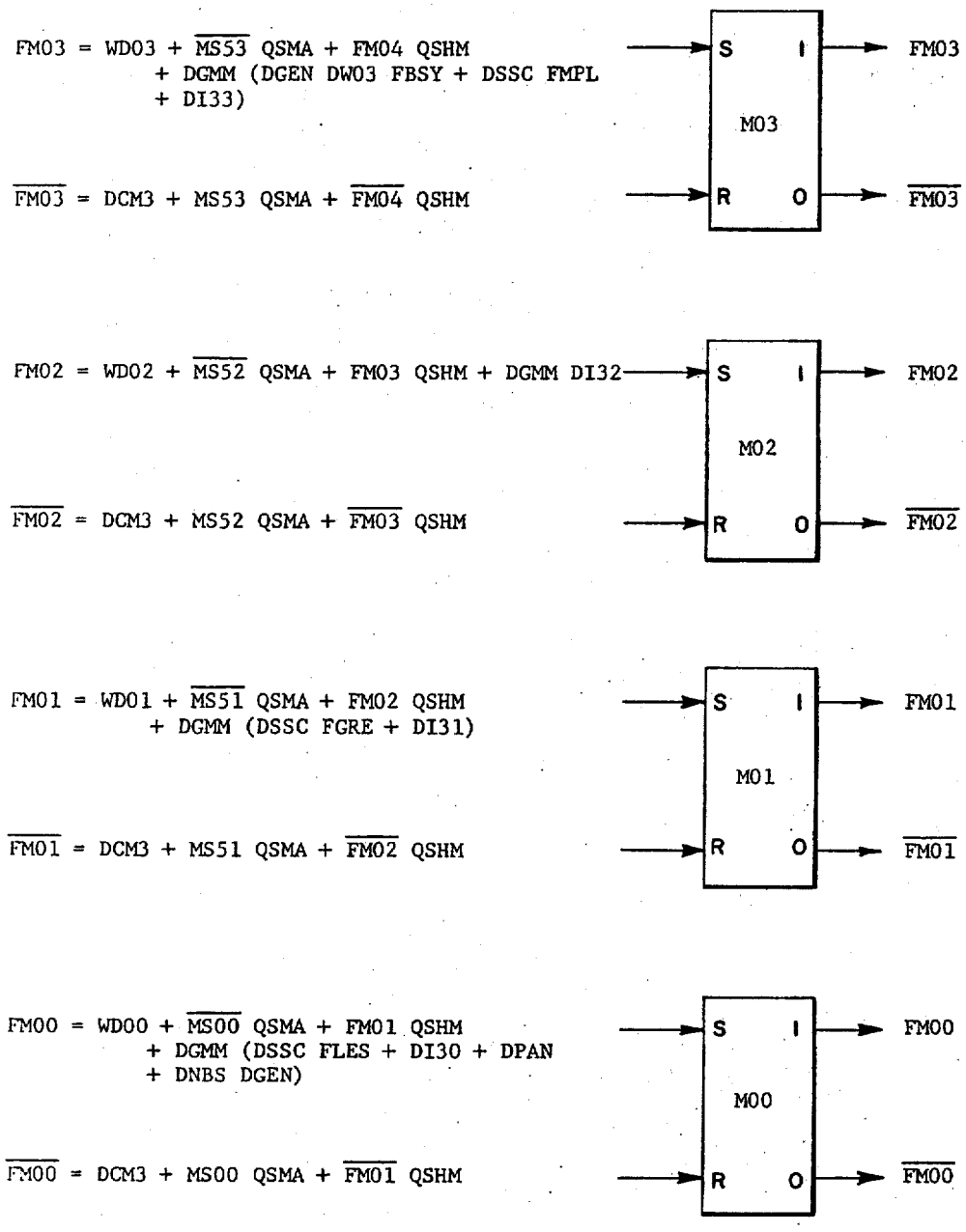
PROGRAM PROCESSOR

FIG. 61.



PROGRAM PROCESSOR

FIG. 62.

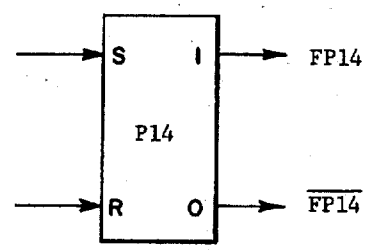


PROGRAM PROCESSOR

FIG. 63.

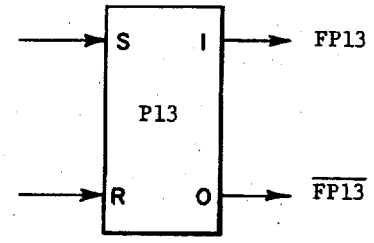
$FP14 = (\overline{FP13})\ \overline{DSPT}\ \overline{FP14} + QSPX\ \overline{MS64}$

$\overline{FP14} = (\overline{FP13})\ \overline{DSPT}\ FP14 + QSPX\ MS64$



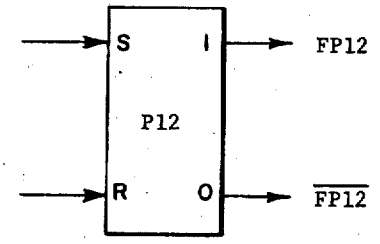
$FP13 = (\overline{FP12})\ \overline{DSPT}\ \overline{FP13} + QSPX\ \overline{MS63}$

$\overline{FP13} = (\overline{FP12})\ \overline{DSPT}\ FP13 + QSPX\ MS63$



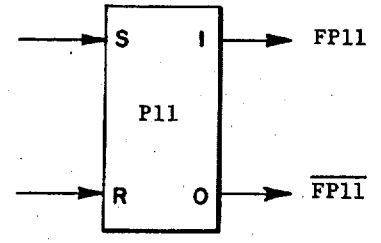
$FP12 = (\overline{FP11})\ \overline{DSPT}\ \overline{FP12} + QSPX\ \overline{MS12}$

$\overline{FP12} = (\overline{FP11})\ \overline{DSPT}\ FP12 + QSPX\ MS12$



$FP11 = (\overline{FP10})\ \overline{DSPT}\ \overline{FP11} + QSPX\ \overline{MS11}$

$\overline{FP11} = (\overline{FP10})\ \overline{DSPT}\ FP11 + QSPX\ MS11$

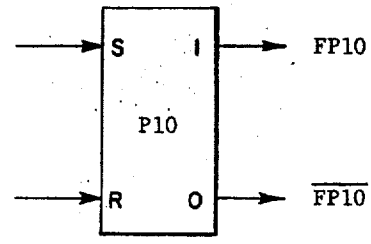


PROGRAM PROCESSOR

FIG. 64

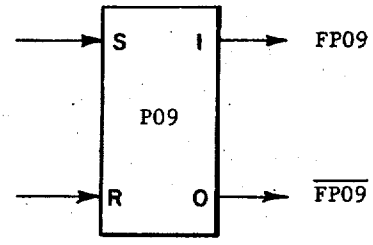
$FP10 = (\overline{FP09})\ \overline{DSPT}\ \overline{FP10} + QSPX\ \overline{MS10}$

$\overline{FP10} = (\overline{FP09})\ \overline{DSPT}\ FP10 + QSPX\ MS10$



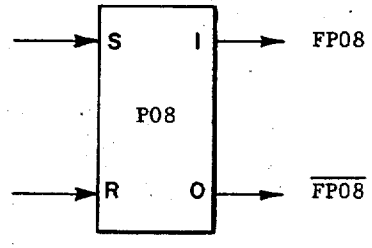
$FP09 = (\overline{FP08})\ \overline{DSPT}\ \overline{FP09} + QSPX\ \overline{MS59}$

$\overline{FP09} = (\overline{FP08})\ \overline{DSPT}\ FP09 + QSPX\ MS59$



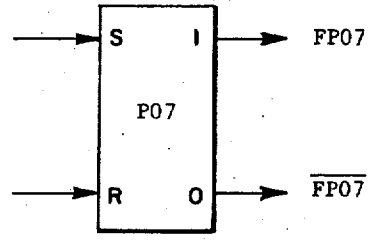
$FP08 = (\overline{FP07})\ \overline{DSPT}\ \overline{FP08} + QSPX\ \overline{MS58}$

$\overline{FP08} = (\overline{FP07})\ \overline{DSPT}\ FP08 + QSPX\ MS58$



$FP07 = (\overline{FP06})\ \overline{DSPT}\ \overline{FP07} + QSPX\ \overline{MS57}$

$\overline{FP07} = (\overline{FP06})\ \overline{DSPT}\ FP07 + QSPX\ MS57$

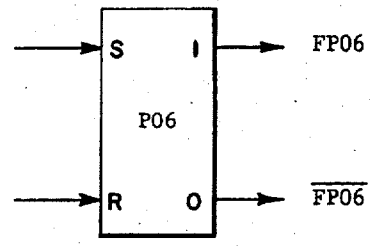


PROGRAM PROCESSOR

FIG. 65.

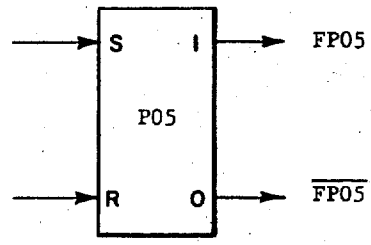
$FP06 = (\overline{FP05})\ \overline{DSPT}\ \overline{FP06} + QSPX\ \overline{MS06}$

$\overline{FP06} = (\overline{FP05})\ \overline{DSPT}\ FP06 + QSPX\ MS06$



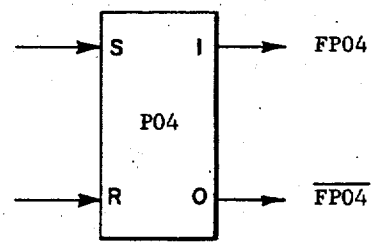
$FP05 = (\overline{FP04})\ \overline{DSPT}\ \overline{FP05} + QSPX\ \overline{MS05}$

$\overline{FP05} = (\overline{FP04})\ \overline{DSPT}\ FP05 + QSPX\ MS05$



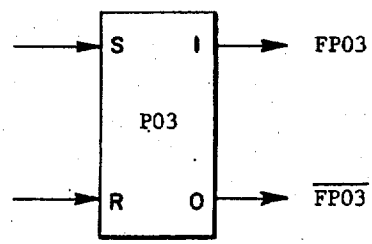
$FP04 = (\overline{FP03})\ \overline{DSPT}\ \overline{FP04} + QSPX\ \overline{MS04}$

$\overline{FP04} = (\overline{FP03})\ \overline{DSPT}\ FP04 + QSPX\ MS04$



$FP03 = (\overline{FP02})\ \overline{DSPT}\ \overline{FP03} + QSPX\ \overline{MS53}$

$\overline{FP03} = (\overline{FP02})\ \overline{DSPT}\ FP03 + QSPX\ MS53$

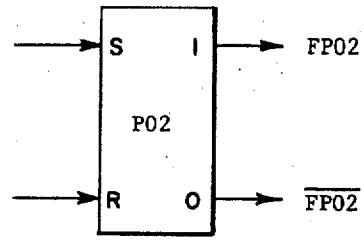


PROGRAM PROCESSOR

FIG. 66.

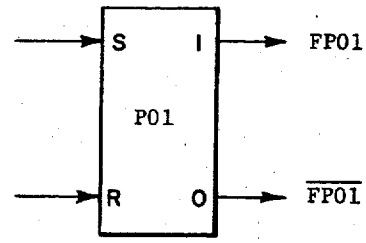
$FP02 = (\overline{FP01})\ \overline{DSPT}\ \overline{FP02} + QSPX\ \overline{MS52}$

$\overline{FP02} = (\overline{FP01})\ \overline{DSPT}\ FP02 + QSPX\ MS52$



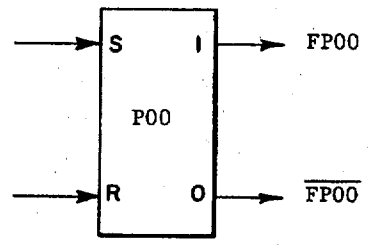
$FP01 = (\overline{FP00})\ \overline{DSPT}\ \overline{FP01} + QSPX\ \overline{MS51}$

$\overline{FP01} = (\overline{FP00})\ \overline{DSPT}\ FP01 + QSPX\ MS51$



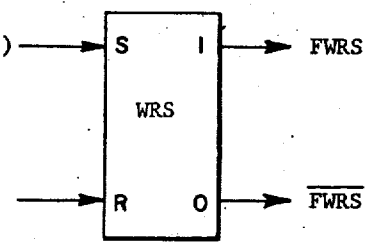
$FP00 = (DPCU)\ \overline{DSPT}\ \overline{FP00} + QSPX\ \overline{MS00}$

$\overline{FP00} = (DPCU)\ \overline{DSPT}\ FP00 + QSPX\ MS00$



$FWRS = DRIT\ (DW11\ FTL0 + DW12\ FTL0 + DW12\ FTL3) \cdot QTCK$

$\overline{FWRS} = QTRK$

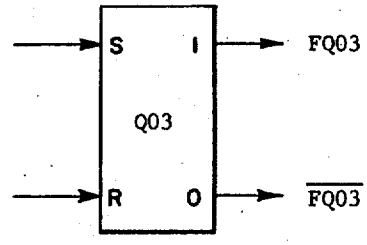


PROGRAM PROCESSOR

FIG. 67.

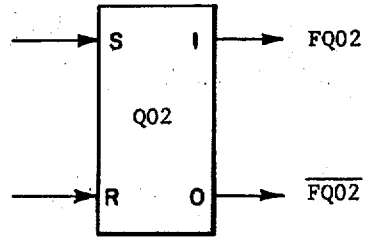
$FQ03 = FN03\ DWT9$

$\overline{FQ03} = DCLQ + \overline{FN03}\ DWT9$



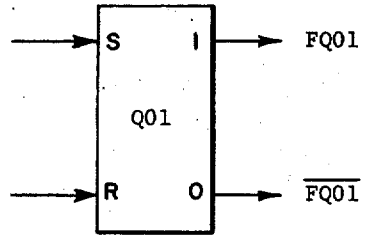
$FQ02 = FN02\ DWT9 + \overline{DAOX}\ FSRI\ FQ00\ FQ01\ \overline{FQ02} \cdot QTCK + FM17\ DMQX$

$\overline{FQ02} = DCLQ + \overline{FN02}\ DWT9 + \overline{DAOX}\ FSRI\ FQ00\ FQ02 \cdot QTCK$



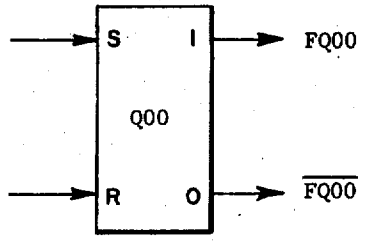
$FQ01 = FN01\ DWT9 + \overline{DAOX}\ FSRI\ FQ00\ \overline{FQ01}\ \overline{FQ02} \cdot QTCK + FM16\ DMQX$

$\overline{FQ01} = DCLQ + \overline{FN01}\ DWT9 + \overline{DAOX}\ FSRI\ FQ00\ FQ01 \cdot QTCK$



$FQ00 = FN00\ DWT9 + \overline{DAOX}\ FSRI\ \overline{FQ00}\ QTCK + FM15\ DMQX$

$\overline{FQ00} = DCLQ + \overline{FN00}\ DWT9 + \overline{DAOX}\ FSRI\ FQ00\ QTCK$

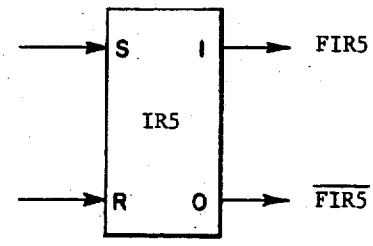


PROGRAM PROCESSOR

FIG. 68.

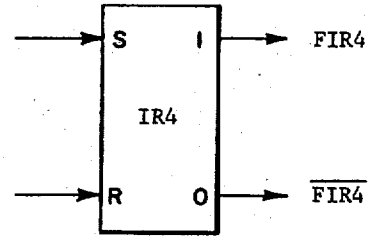
FIR5 = QMIX FM23

$\overline{FIR5}$ = QMIX $\overline{FM23}$



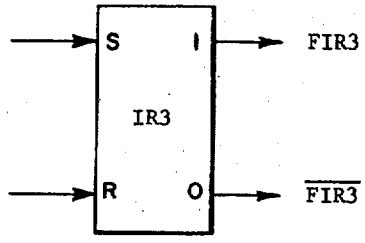
FIR4 = QMIX FM22

$\overline{FIR4}$ = QMIX $\overline{FM22}$



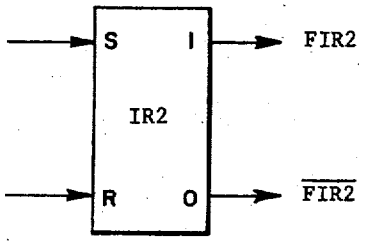
FIR3 = QMIX FM21

$\overline{FIR3}$ = QMIX $\overline{FM21}$



FIR2 = QMIX FM20

$\overline{FIR2}$ = QMIX $\overline{FM20}$

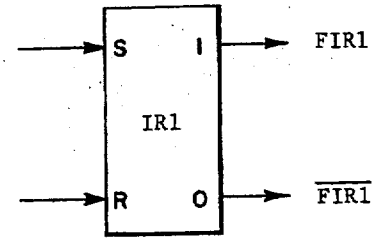


PROGRAM PROCESSOR

FIG. 69.

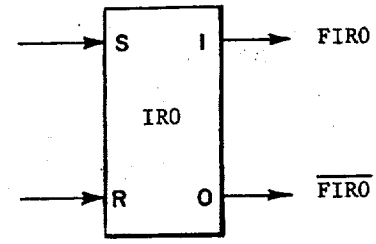
$FIR1 = QMIX\ FM19$

$\overline{FIR1} = QMIX\ \overline{FM19}$



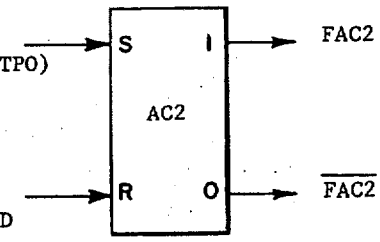
$FIR0 = QMIX\ FM18$

$\overline{FIR0} = QMIX\ \overline{FM18}$



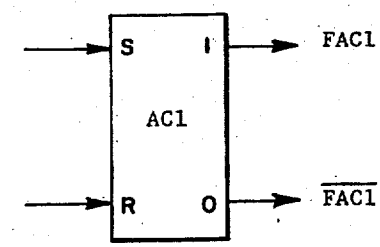
$$FAC2 = QTC0\ \overline{FAC2}\ \overline{DSGO}\ \overline{DDBO}\ [\underline{DAC1}\ (DACU + DBYD\ DTPO) + \underline{DAC1}\ (\underline{DACD} + DBYU\ DTPO) + DBUD] + QTC2\ \underline{FAC2}\ DSGO\ \overline{DDBO} \cdot (DAC1\ DCAU + \overline{DAC1}\ DCAD)$$

$$\overline{FAC2} = QTC0\ DAC2\ (DW00 + DBUD + DAUQ + DTPO\ DADU) + QTC2\ DAC2\ (DTPO\ \underline{DAUD} + DCAU\ DQDO\ DAC1 + DCAD\ DQDO\ \overline{DAC1})$$



$$FAC1 = QTC0\ \overline{DAC1}\ \overline{DSGO}\ (DACU\ \overline{DTAT} + DACD\ \underline{\overline{DTAF}} + DBYU\ \underline{DTPO}\ \underline{FAC2} + DBYD\ \underline{DTPO}\ \overline{FAC2}) + QTC2\ \underline{DAC1}\ \overline{DSGO}\ (DCAD\ \overline{DTAF} + DCAU\ \overline{DTAT})$$

$$\overline{FAC1} = QTC0\ DAC1\ DADU + QTC2\ DAC1\ DAUD$$

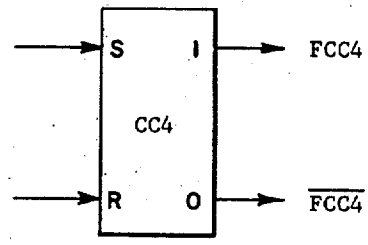


PROGRAM PROCESSOR

FIG. 70.

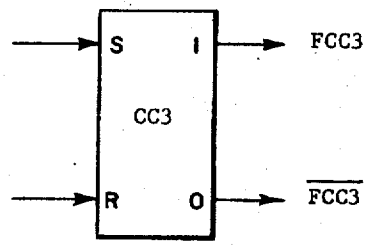
$FCC4 = \overline{DS54}\ QSCX + FN04\ QNCX + FCC4\ DCTV\ DCAB\ QTCK\ DCCU$

$\overline{FCC4} = DCLC + DS54\ QSCX + \overline{FN04}\ QNCX + FCC4\ DCTV\ DCAB\ QTCK\ DCCU$



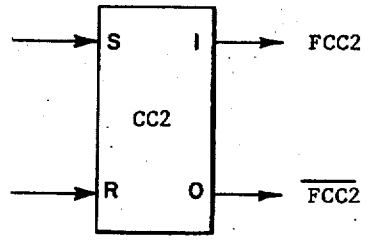
$FCC3 = \overline{MS53}\ QSCX + FN03\ QNCX + DCCU\ FCC3\ FCC2\ DCAB\ QTCK$

$\overline{FCC3} = DCLC + MS53\ QSCX + \overline{FN03}\ QNCX + DCCU\ DCTV\ DCAB\ QTCK$



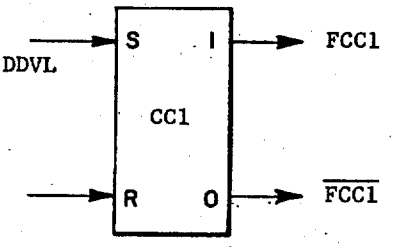
$FCC2 = \overline{MS52}\ QSCX + FN02\ QNCX + DCCU\ \overline{FCC2}\ DCAB\ QTCK$

$\overline{FCC2} = DCLC + MS52\ QSCX + \overline{FN02}\ QNCX + DCCU\ FCC2\ DCAB\ QTCK$



$FCC1 = DCL4\ QSCX + FN01\ QNCX + DCCU\ \overline{FCC1}\ FCC0\ QTCK + DW15\ FTL1\ DDVL$

$\overline{FCC1} = DCLC + \overline{DCL4}\ QSCX + \overline{FN01}\ QNCX + DCCU\ DCAB\ QTCK$

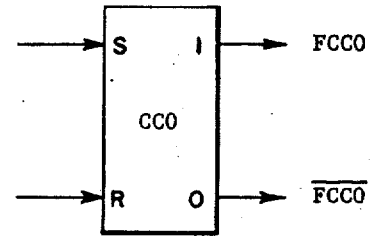


PROGRAM PROCESSOR

FIG. 71.

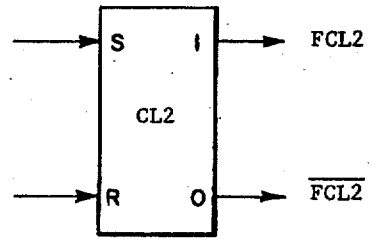
$FCCO = \overline{DS50}\ QSCX + FN00\ QNCX + DCCU\ \overline{FCCO}\ QTCK + DW15\ FTL1\ DDVL$

$\overline{FCCO} = DCLC + DS50\ QSCX + \overline{FN00}\ QNCX + DCCU\ FCCO\ QTCK$



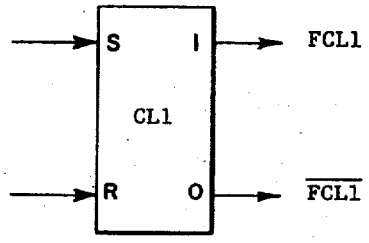
$FCL2 = \overline{DVRF}\ DW01\ FTL3\ FM21 + DW14\ FTL2\ DVRB$

$\overline{FCL2} = \overline{DVRF}\ DW01\ \overline{FM21}\ \overline{FTL3} + DWT1 + DW14\ FTL2\ \overline{DVRB}$



$FCL1 = \overline{FXS1}\ \overline{FXS2}\ DW01\ \overline{FTL3}\ DNWX + DW14\ FTL2\ \overline{DVRA}$

$\overline{FCL1} = \overline{FXS1}\ \overline{FXS2}\ DW01\ \overline{DNWX}\ FTL3 + DWT1 + DW14\ FTL2\ DVRA$

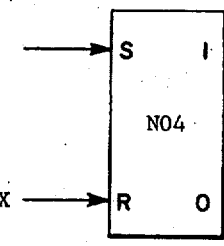


PROGRAM PROCESSOR

FIG.72.

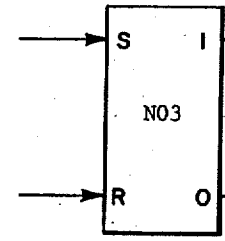
$$FN04 = DW14\ FTL1\ DQDO\ DAAC + FCC4\ QCNX + QNCD\ DW02\ FN04\ DNMF\ DNTS + DW11\ DAOX\ FTL3$$

$$\overline{FN04} = (DNS4 + DNS2 + DNS8 + DNS0) + \overline{FCC4}\ QCNX + QNCD\ (DNMF\ DNTS\ FN04\ \overline{DW02} + FN04\ DNTS\ DWDN + DWDB)$$



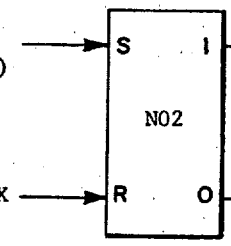
$$FN03 = (DNS2 + DNS8 + DNSL) + FCC3\ QCNX + FQ03\ QQNX + QNCD\ (DNMF\ DNTS\ \overline{DW02} + DNTS\ DWDN)$$

$$\overline{FN03} = (DNS4 + DNS6 + DNS0) + \overline{FCC3}\ QCNX + \overline{FQ03}\ QQNX + QNCD\ (DNMF\ DNAT\ \overline{DW02} + DNAT\ DWDN + DWDB)$$



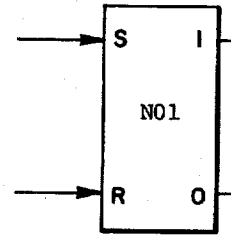
$$FN02 = (DNS4 + DNS2) + FCC2\ QCNX + FQ02\ QQNX + QNCD\ (DNMF\ FN02\ DW02 + FN02\ DWDN)$$

$$\overline{FN02} = (DNS8 + DNS6 + DNS0 + DNSL) + \overline{FCC2}\ QCNX + \overline{FQ02}\ QQNX + QNCD\ (DNMF\ FN02\ \overline{DW02} + FN02\ DWDN + DWNL)$$



$$FN01 = FCC1\ QCNX + FQ01\ QQNX + QNCD\ \overline{DW02}\ DNMF + DIMP\ DW14\ FTL3\ DAAC\ \overline{DDAC}$$

$$\overline{FN01} = DNR0 + \overline{FCC1}\ QCNX + \overline{FQ01}\ QQNX + QNCD\ (FN00\ FN01\ DW02 + DWNL)$$

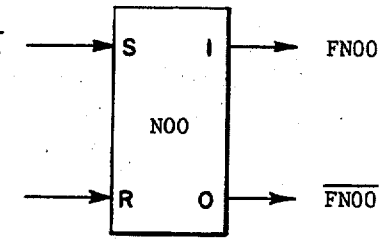


PROGRAM PROCESSOR

FIG. 73.

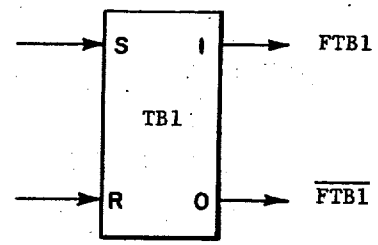
$FN00 = FCC0\ QCNX + FQ00\ QQNX + QNCD\ \overline{FN00}\ \overline{DW02} + DW14\ FTL3\ DAC1\ DIMP$

$\overline{FN00} = DNR0 + \overline{FCC0}\ QCNX + \overline{FQ00}\ QQNX + QNCD\ (FN00\ \overline{DW02} + DWNL)$



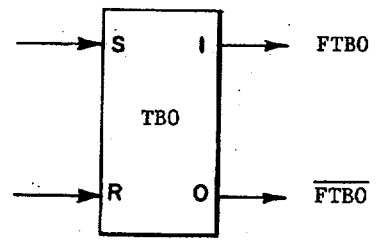
$FTB1 = DBCU\ QTCK\ FTB0\ \overline{FTB1}$

$\overline{FTB1} = DBCU\ FMOD\ FTB1\ \overline{QTCK} + QTCK\ DRBG + QTCK\ DBCU\ \overline{FMOD}\ FTB0\ FTB1$



$FTB0 = DBCU\ \overline{FTB0}\ \overline{FMOD\ QTCK} + DBCU\ \overline{FTB0}\ FTB1\ QTCK$

$\overline{FTB0} = DBCU\ QTCK\ FTB0 + QTCK\ DRBG$

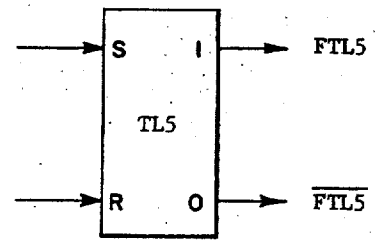


PROGRAM PROCESSOR

FIG. 74.

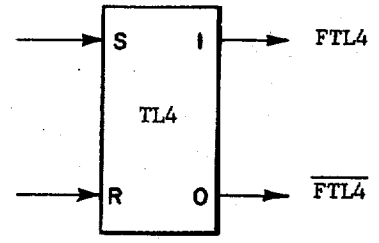
FTL5 = QTLP FTL4

$\overline{FTL5}$ = QTLP $\overline{FTL4}$ + DMLR



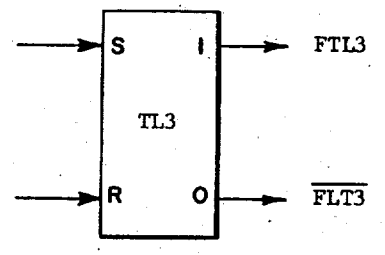
FTL4 = QTLP FTL3

$\overline{FTL4}$ = QTLP $\overline{FTL3}$ + DMLR



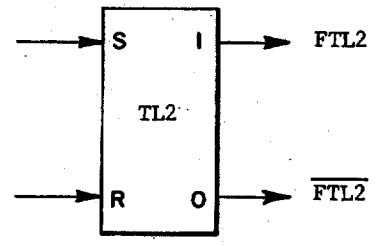
FTL3 = QTLP FTL2

$\overline{FTL3}$ = QTLP $\overline{FTL2}$ + DMLR



FTL2 = QTLP FTL1 + QTC0 DVGD + DMLR

$\overline{FTL2}$ = QTLP $\overline{FTL1}$ + QTC2 DVGC

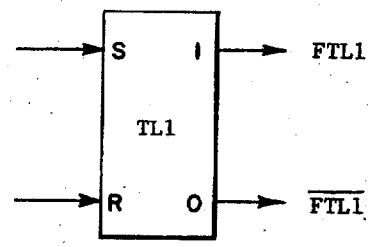


PROGRAM PROCESSOR

FIG.75.

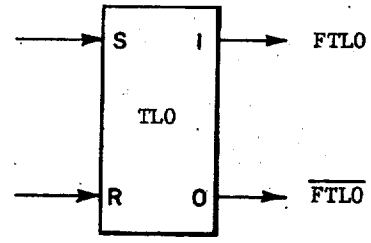
$FTL1 = QTLP\ FTL0$

$\overline{FTL1} = QTLP\ \overline{FTL0} + DMLR$



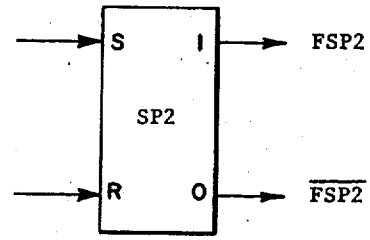
$FTL0 = QTLP\ FTL5 + QTC2\ DVGC$

$\overline{FTL0} = QTLP\ \overline{FTL5} + QTC0\ DVGD + DMLR$



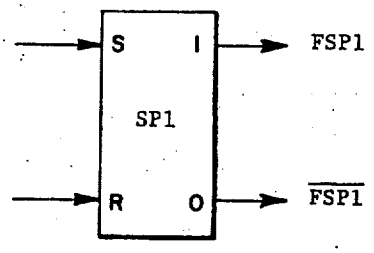
$FSP2 = FTL4\ MJE0\ (DGPP + DW02\ \overline{DLA0}\ DIMP\ \overline{FSP2})$
$\cdot\ QTCK + (DVRC + DVRD + DW02\ DSRD$
$\cdot\ DLFT\ FSP1\ FTL1 + DW11\ DLFT\ DSRD$
$\cdot\ FSP1\ FTL4)\ QTCK$

$\overline{FSP2} = FTL3\ (DGPP + DW11\ DLFT\ DSRD)\ QTCK$
$+ (DVRE + DWT1 + DGE4)\ QTCK$



$FSP1 = (DVRC + DVRE)\ QTCK$
$+ DW03\ DLFT\ DSRD\ FTL4\ MJE0\ QTCK$

$\overline{FSP1} = (DVRD + DWT1)\ QTCK$
$+ DW03\ DLFT\ DSRD\ FTL3\ QTCK$

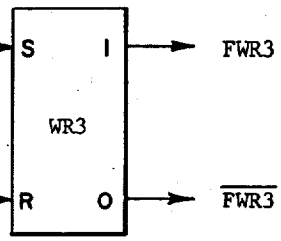


PROGRAM PROCESSOR

FIG. 76.

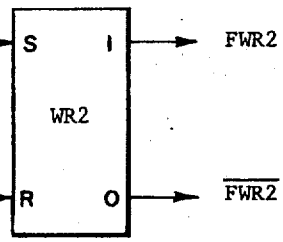
$$FWR3 = QTC0\ \overline{DINP}\ (\underline{DSW3} + DHXM + DVSP + DWSM) + QTC2\ \overline{DINP}\ (DVSA + DVSK + DVSH) + FCSF\ FTL2$$

$$\overline{FWR3} = QTC0\ (DVSM + DVTE + DINP + DFDE + DVTJ + DRW3) + QTC2\ (DINP + DSRD\ DW15)$$



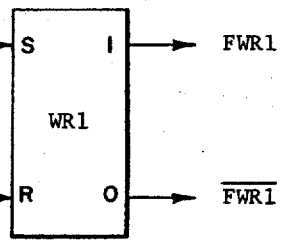
$$FWR2 = DSW2\ \overline{DINP}\ \underline{QTC0} + QTC2\ \overline{DINP}\ (DVSC + DVSF + DVVW)$$

$$\overline{FWR2} = QTC0\ (DVSM + DVTE + DFDE + DVTJ + DINP + DRW2) + QTC2\ (DVSK + DINP + DW06) + FCSF\ FTL2$$



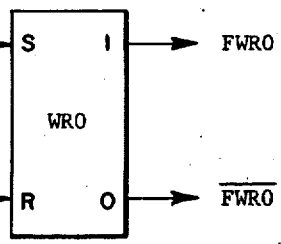
$$FWR1 = QTC0\ \overline{DINP}\ (\underline{DVSM} + DSW1 + DHXM + DVSP) + QTC2\ \overline{DINP}\ [DVSH + DW04\ (DVSB + DMOV + DMVL + DASF\ FCOR)] + FCSF\ FTL2$$

$$\overline{FWR1} = (DVTA + DVTB + DRW1 + DINP + DVTJ)\ QTC0 + QTC2\ (DW06 + DVSF + DVZG + DINP + DW02\ DSRD\ DNEZ)$$



$$FWR0 = (DW08 + \underline{DSW0} + DFDE + DDNA + DVSM) \cdot \overline{DINP}\ QTC0 + [DVTL + DW06 + DVSA + DW04\ (\underline{DSAS} + DVTM + DVLR + DVSB + DMND)]\ \overline{DINP}\ QTC2$$

$$\overline{FWR0} = (DVTA + DW09 + DW03 + DHXM + DVTB + DINP + DRW0)\ QTC0 + QTC2\ (DVSC + DVSF + DINP + DVSG) + FCSF\ FTL2$$

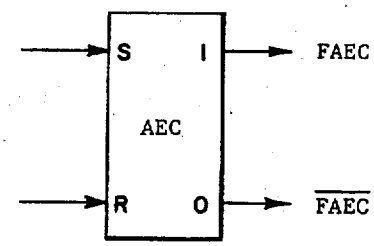


PROGRAM PROCESSOR

FIG. 77.

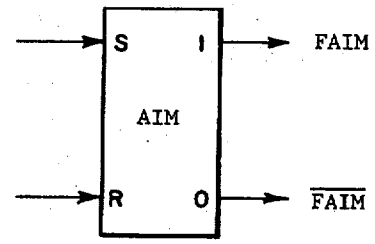
FAEC = (DR05 + DRRV + DR00) FTL5

$\overline{FAEC}$ = DIMR



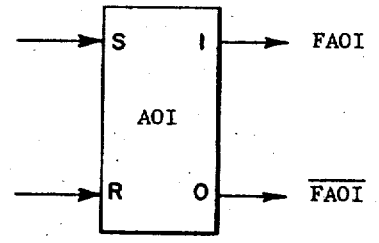
FAIM = FMAI ($\overline{DW00}$ + $\overline{FMAN}$) + DIAM

$\overline{FAIM}$ = DW00 FTL3 + DMLR



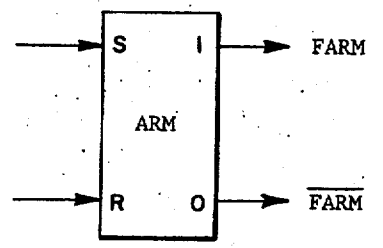
FAOI = QTC2 DW04 DSRD
+ QTC0 DVJS $\overline{FM19}$ FM18 $\overline{FAOI}$

$\overline{FAOI}$ = QTC0 (DW04 DSRD + DW00 + DW01 DIMM)



FARM = SACS

$\overline{FARM}$ = $\overline{DACS}$ $\overline{FMNX}$

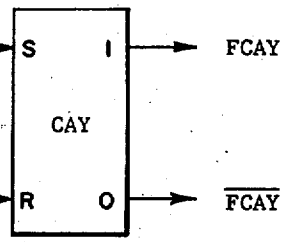


PROGRAM PROCESSOR

FIG. 78.

$$FCAY = [DNAI + DW01\ FTL3\ DSBM\ DVCA + DMVF + DW05\ FTL1\ FCRE + DCID + DWT7 + \underline{DW06}\ FTL3\ \underline{(DASG\ DAAC + DDVL + \overline{DMVL}\ FCRE\ \overline{DMOV})}]\ QTCK + DVMA\ QCER$$

$$\overline{FCAY} = (DWT1 + DWT5 + DWL3 + DWT3 + DMRC + DWT4 + DW03\ FTL5)\ QTCK$$

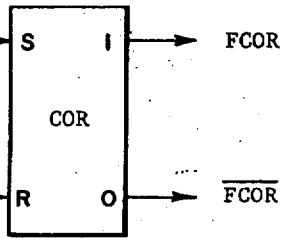


FCAY

$\overline{FCAY}$

$$FCOR = QTCK\ (DW15\ FTL3\ \underline{DNEZ}\ \overline{DCCZ}\ \overline{FCOR}\ DEDT + DW05\ FTL5\ \overline{FLSN}\ \overline{DDC0}\ DASG\ DLAO + DW00\ DDVL\ FTL4) + QTCK\ DVJK\ DSTZ$$

$$\overline{FCOR} = QTCK\ [DW06\ FTL3\ DASG\ DLAO\ \overline{DSGO} + DW15\ FTL3\ DNEZ\ FCOR\ DEDT + DWT1 + DW04\ \underline{FTL3}\ (DASG\ DSGO + DDVL\ \overline{FAOI}\ DAAC)]$$

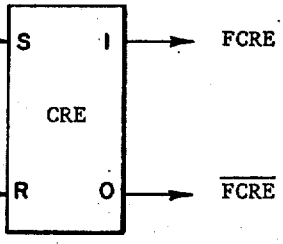


FCOR

$\overline{FCOR}$

$$FCRE = (DVEV + DW02\ \overline{MC23}\ DMOV + DMDC)\ QSEX + [DW04\ DDC0\ DASG + \underline{DMDC} + DW15\ (DMEC\ DIGE\ \overline{MC23} + DSBM\ MC23 + DVLR\ DDC0)]\ QSMC$$

$$\overline{FCRE} = (DWT1 + DW06\ FTL4\ \overline{DMOV} + DWT8 + DW05\ FTL3 + FMOD\ \overline{FTB1}\ FTB0)$$

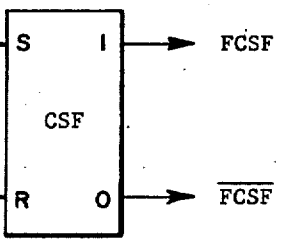


FCRE

$\overline{FCRE}$

$$FCSF = DMLR$$

$$\overline{FCSF} = DMRC + FSRE$$



FCSF

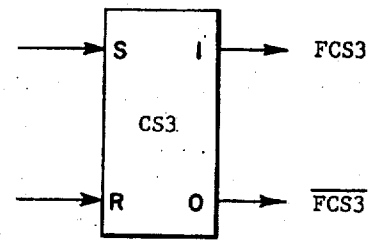
$\overline{FCSF}$

PROGRAM PROCESSOR

FIG. 79.

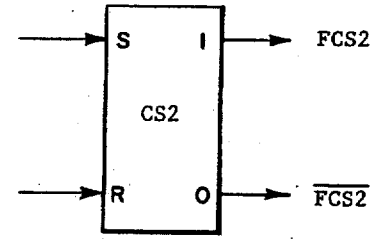
FCS3 = DW11 FTL4 DMVL DCAC

$\overline{FCS3}$ = DWT3 + DMRC + DINP



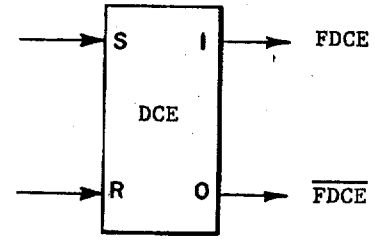
FCS2 = DW06 FTL3 DMVL + DVLR DW03 FTL1

$\overline{FCS2}$ = DWT9 + DMRC + DINP
+ DDVL FTL3 (DW03 + DW06)



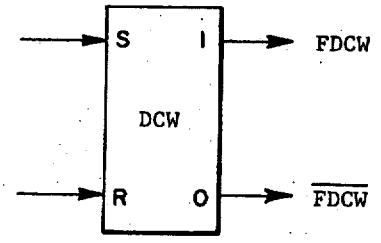
FDCE = DW05 FTL1 (DASG + DVLR)
+ DW06 FTL3 (DASG + DVLR + DMVL)
+ DWT9 + DW09 FTL1 DCDA

$\overline{FDCE}$ = DWT3 + DWT8 + DWL3 + DWT4 + DMRC
+ DINP + DW03 DVLR FTL3



FDCW = FLPM + SDMS

$\overline{FDCW}$ = $\overline{DDMS}$ $\overline{FLPM}$ $\overline{FMNX}$

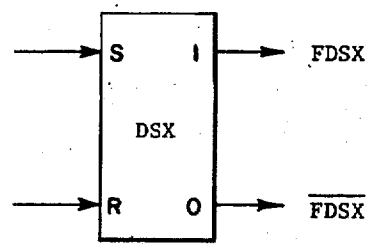


PROGRAM PROCESSOR

FIG. 80.

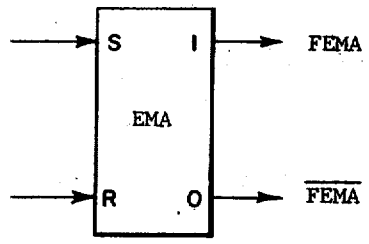
$$FDSX = DNAI + DW02\ \underline{FTL5}\ (DMRM\ DBCL\ DIGB + DBRZ\ \overline{FRNZ}\ \underline{DLAO)} + DW03\ FTL3 \cdot (DVLR\ DDAC\ \overline{FFVO} + DMOV) + DW15\ FTL3\ DMOV + DW04\ FTL4\ DAMG\ FCOR\ DLAO\ \overline{DSGO}$$

$$\overline{FDSX} = DWL4 + DWL1 + DWT5 + DWT1 + DW02\ FTL1 + DMRC$$



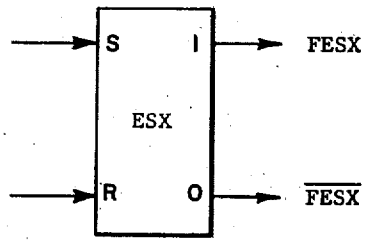
$$FEMA = SERS$$

$$\overline{FEMA} = \overline{SERS}\ \overline{FMNX}$$



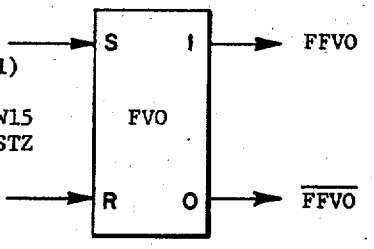
$$FESX = QTCK\ DSES + QTCK\ (DWT7 + DMVF + DESS + DMVD)$$

$$\overline{FESX} = QTCK\ (DVTS + DWT1 + DWL1 + DWT5 + DMVE + DVTY + DWL3 + DMRC + DWT4 + DWT3 + DVHZ + DW03\ FTL4\ DESA)$$



$$FFVO = DW04\ FTL4\ FCRE\ FLSN\ DASG\ DLAO + \underline{DW07}\ DDVL\ DCE1\ DSGP + (\overline{FTVO} + FE01) \cdot \overline{FM05}\ \underline{DSCA}\ FM04 + DVHF + DW15\ FTL5 \cdot DVLR\ \overline{FN00}\ FN01\ DNAT + FCRE\ DASB\ DW15 \cdot FTL4 + DW02\ FTL4\ DVLR\ DDAC\ DSE9\ DSTZ$$

$$\overline{FFVO} = DSCZ\ \overline{FM04} + DRCF + DMLR + DDVL\ FMPL\ DW15\ FTL1$$

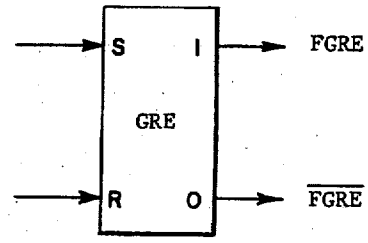


PROGRAM PROCESSOR

FIG.8I.

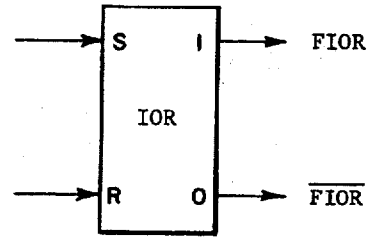
$FGRE = DVMR + FM01\ DSCA$

$\overline{FGRE} = DVMC + DVMS + DSCZ\ \overline{FM01} + DTSB$



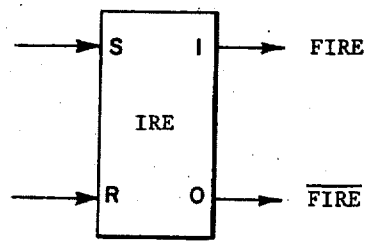
$FIOR = DW03\ \overline{FBSY}\ DBYC\ FTL3$

$\overline{FIOR} = DR03\ FTL3\ \overline{DDIB} + DMLR$



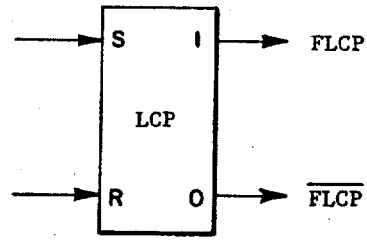
$FIRE = DW01\ FTL3\ DBRC\ DVCA + DMVF$

$\overline{FIRE} = DVEV\ FTL5\ FCRE + DMRC + DWT4 + DINP + DWL1 + DIMR$



$FLCP = QTC0\ DBL2\ \overline{DINP}$

$\overline{FLCP} = QTC0\ (DR00\ \overline{DBL2} + DINP)$

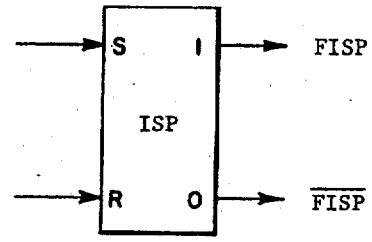


PROGRAM PROCESSOR

FIG. 82.

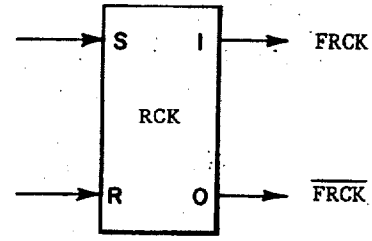
$FISP = QTCK$

$\overline{FISP} = QTSP\ FISP + DICR$



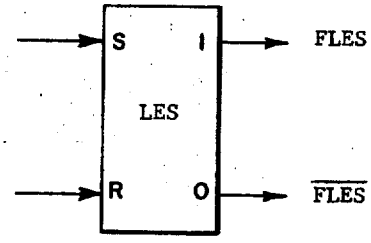
$FRCK = QTRK$

$\overline{FRCK} = QTLP\ (FTL1 + FTL5) + DICR$



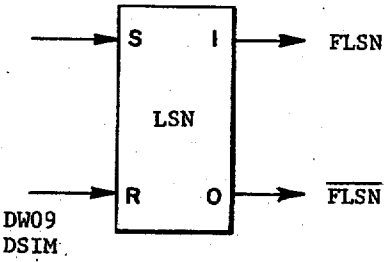
$FLES = DVMS + DSCA\ FM00 + DTSB\ \overline{FM00}$

$\overline{FLES} = DVMC + DVMR + DSCZ\ \overline{FM00} + DTSB\ FM00$



$FLSN = (DWT1 + DW05\ FTL1\ DDVL)\ QTCK$

$\overline{FLSN} = DVMA\ QCER + (DW07\ FTL2\ FMPL\ FSIN\ \overline{DMRM} + DW07\ FTL2\ FMPL\ DMRM\ FSIN + FTL2\ DW09 \cdot DCDA\ DAAC\ DVTR + DW04\ FTL3\ DSMG\ DSIM + DW11\ FTL3\ DSRD\ DSIG\ DSIL + DW07\ FTL2 \cdot DDVL)\ QTCK$

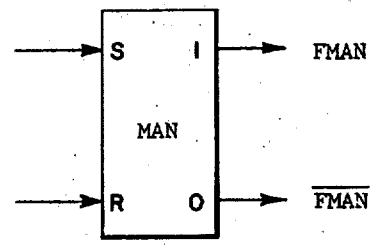


PROGRAM PROCESSOR

FIG.83.

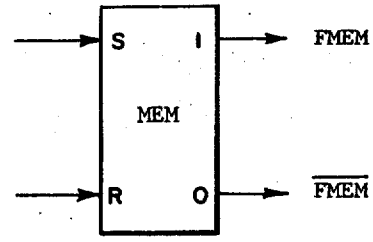
$$FMAN = FSTP\ DEIN\ \overline{DW10}\ QTCO + (FWJE + DMSE + DHSM + DIIP)\ QTCK + DICR$$

$$\overline{FMAN} = FSTT\ \overline{FSSS}\ FTL3\ QTCK + FSTE\ \overline{FSSS}\ FTL3\ QTCK + FSSS\ \overline{FSIE}\ FTL3\ DW10$$



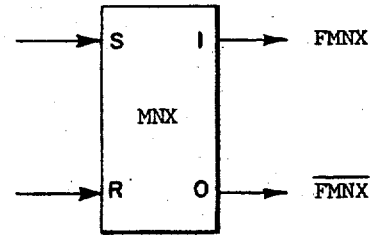
$$FMEM = SMSW\ FNBY$$

$$\overline{FMEM} = DMVR + DMLR + DREM$$



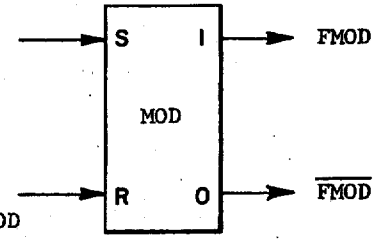
$$FMNX = (FDCW + FPRM + FEMA + FARM) \cdot FNBY\ \overline{FMXS}\ \overline{DMLR}$$

$$\overline{FMNX} = (DMLR + DREM) + DR01\ DCH0\ QTCO$$



$$FMOD = (DW07\ FTL2\ \overline{DNEZ}\ DMVL + DDVL\ FFVO\ FMPL\ DW07\ FTL2)\ QTCK$$

$$\overline{FMOD} = [DMRC + DMVL\ DNE1\ \overline{FTB0}\ FTB1 + DW07\ \overline{DDVL}\ \overline{FTB1}\ \overline{FTB0}\ DDAC\ DCE1\ FMOD + DDVL\ FMPL\ \overline{FTB1}\ \overline{FTB0}\ (DCAC\ DNE1 + DDAC\ DNEZ\ \overline{FCRE} + DCAC\ DNEZ)]\ QTCK + DINP$$

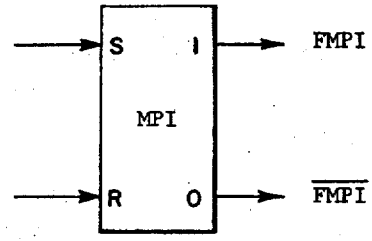


PROGRAM PROCESSOR

FIG. 84.

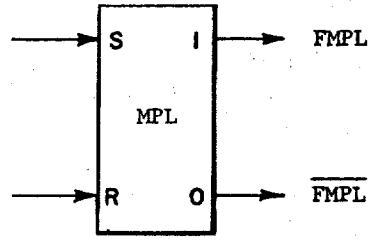
$FMPI = FPIM\ \overline{FPMI}$

$\overline{FMPI} = (DMIR + DMLR)$



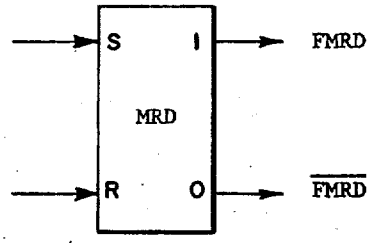
$FMPL = DW15\ DBRC\ FCRE\ FTL3 + DCID + DSCA\ FM03 + DMLR$

$\overline{FMPL} = DW07\ FTL1\ DCAC\ \underline{DMVL} + DSCZ\ \overline{FM03} + DW07\ DDVL\ \overline{FTB1}\ FTB0\ FMOD + DW02\ DVLR\ FTL5\ FFVO$



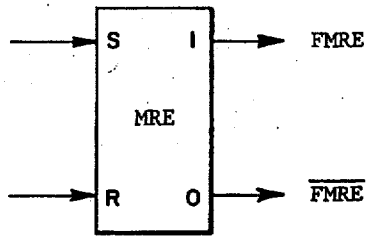
$FMRD = (DW00 + DW01 + DW02 + DW03 + DW04 + \underline{DW05} + \underline{DW07} + \underline{DW08} + DCW9 + DW14 + DR00\ DVEQ)\ \overline{FMRD}\ FTL1 + DW10\ FCSF\ FTL2\ QTCK$

$\overline{FMRD} = (DMBU + DMAI)\ (FTL2 + FSRE) + DMLR + DW10\ FTL4$



$FMRE = SMRS$

$\overline{FMRE} = \overline{SMRS} + \overline{FNBY}$

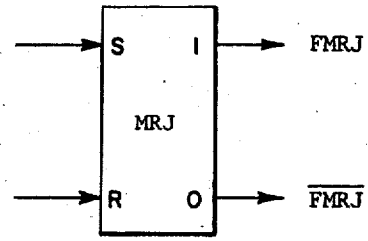


PROGRAM PROCESSOR

FIG.85.

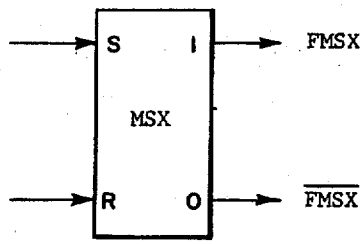
$FMRJ = FMRD\ DAMR\ FTL1\ QTCK$

$\overline{FMRJ} = DMLR + \overline{FTL1}\ QTCK$



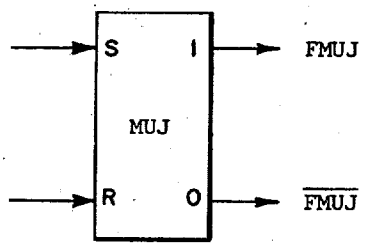
$FMSX = DSMS\ QTCK$

$\overline{FMSX} = (DMXR + DVTY + DR13\ FTL5 + DWT6 + DWT5 + DWL5 + DWL3 + DMRC + DW00\ FTL5 + DWT8 + DWT4)\ QTCK$



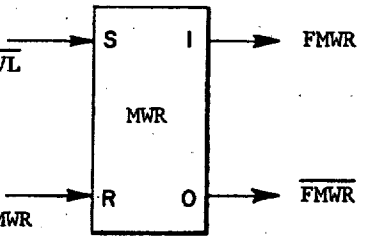
$FMUJ = WD24$

$\overline{FMUJ} = DCM4 + DGEP\ QTCK\ FTL2$



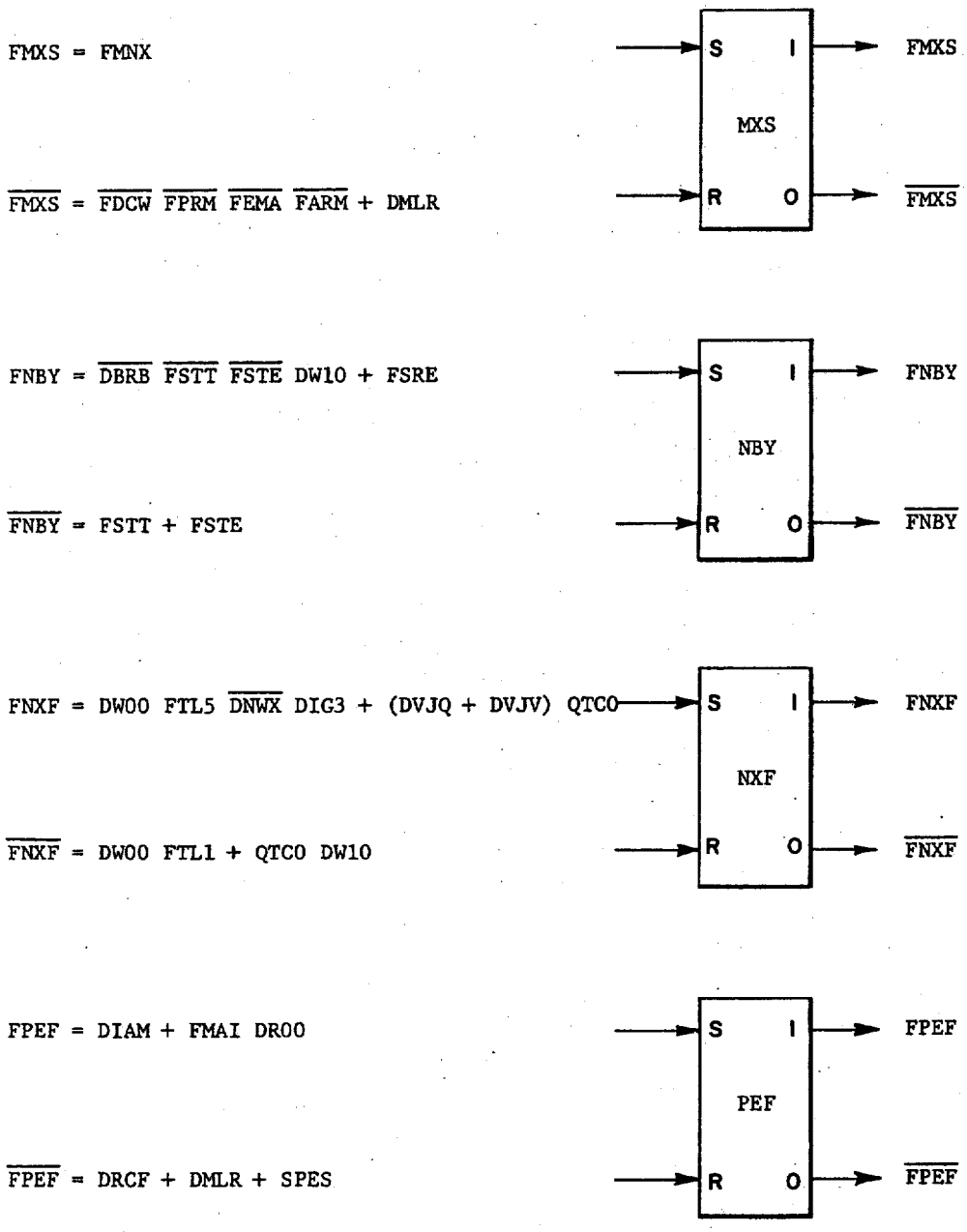
$FMWR = [DW00 + DW01\ (\overline{DIG4}\ \overline{DGCP} + \overline{DVCA}) + DW02 + DW03 + DW04 + DW05\ \overline{DAAE} + DW07\ \overline{DDVL} + DW08 + DCW9 + \overline{DW15} + \overline{DR05} + DR04] \cdot QTCK\ \overline{FTL2} + (DVEQ\ DR01\ \overline{FRVO}\ FTL4 + DR03\ \overline{FRVO}\ FTL4 + DR02\ FTL3)\ QTCK$

$\overline{FMWR} = DWEN\ DMBU\ (FSRE + FTL0) + DMLR + QTCK\ FMWR\ FMAI + QTCK\ DWEN\ DMBU\ FMWR$



PROGRAM PROCESSOR

FIG.86.

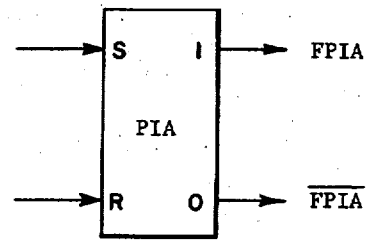


PROGRAM PROCESSOR

FIG. 87.

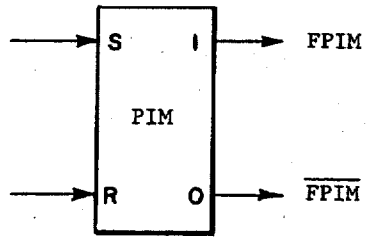
FPIA = (DEI0 DEI1 DEI2 $\underline{DEI3}$ $\underline{DPDC}$ + $\underline{DINP}$
+ FAIM $\underline{DR05}$ $\underline{DDIB}$) FPSA FPIT QTC0
+ DINP FPSA FPIT QTC2

$\overline{FPIA}$ = FCSF QTC2 + [DW01 (FXS1 + $\underline{FXS2}$ + $\overline{DIMM}$)
+ DW03 + DW15] FPIA DINP DIPR QTC0



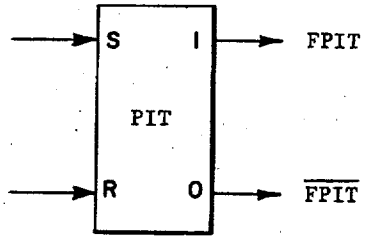
FPIM = SPIS

$\overline{FPIM}$ = $\overline{SPIS}$



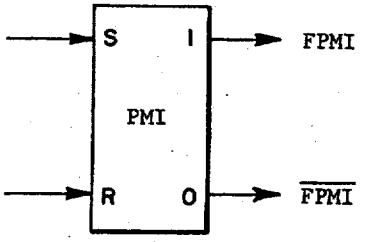
FPIT = DPSU + DSCA FM02

$\overline{FPIT}$ = DMLR + DSCZ $\overline{FM02}$ + SPIR FNBY



FPMI = FPIM FTL0

$\overline{FPMI}$ = $\overline{FPIM}$ FTL3

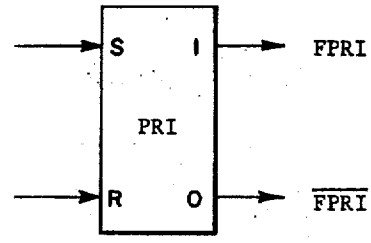


PROGRAM PROCESSOR

FIG.88.

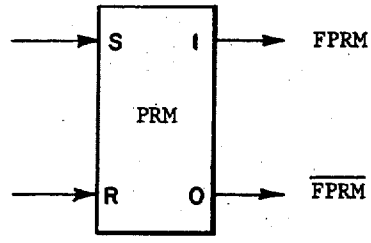
$FPRI = DW01\ \overline{FPAD}\ FPIA$

$\overline{FPRI} = DGEN\ DW03\ FTL5$



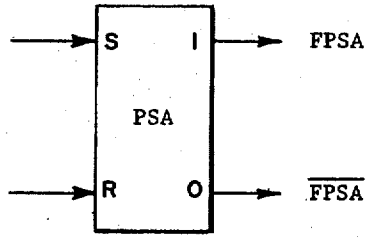
$FPRM = FLPM + SPMS$

$\overline{FPRM} = \overline{DPMS}\ \overline{FLPM}\ \overline{FMNX}$



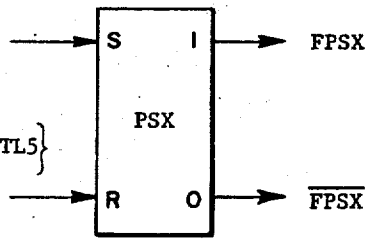
$FPSA = DPSU$

$\overline{FPSA} = DMLR + \overline{FPIT}\ \overline{FTL5}\ DBCL\ DIGA \cdot (DW00\ \overline{DAMS} + DW01\ DXIM)$



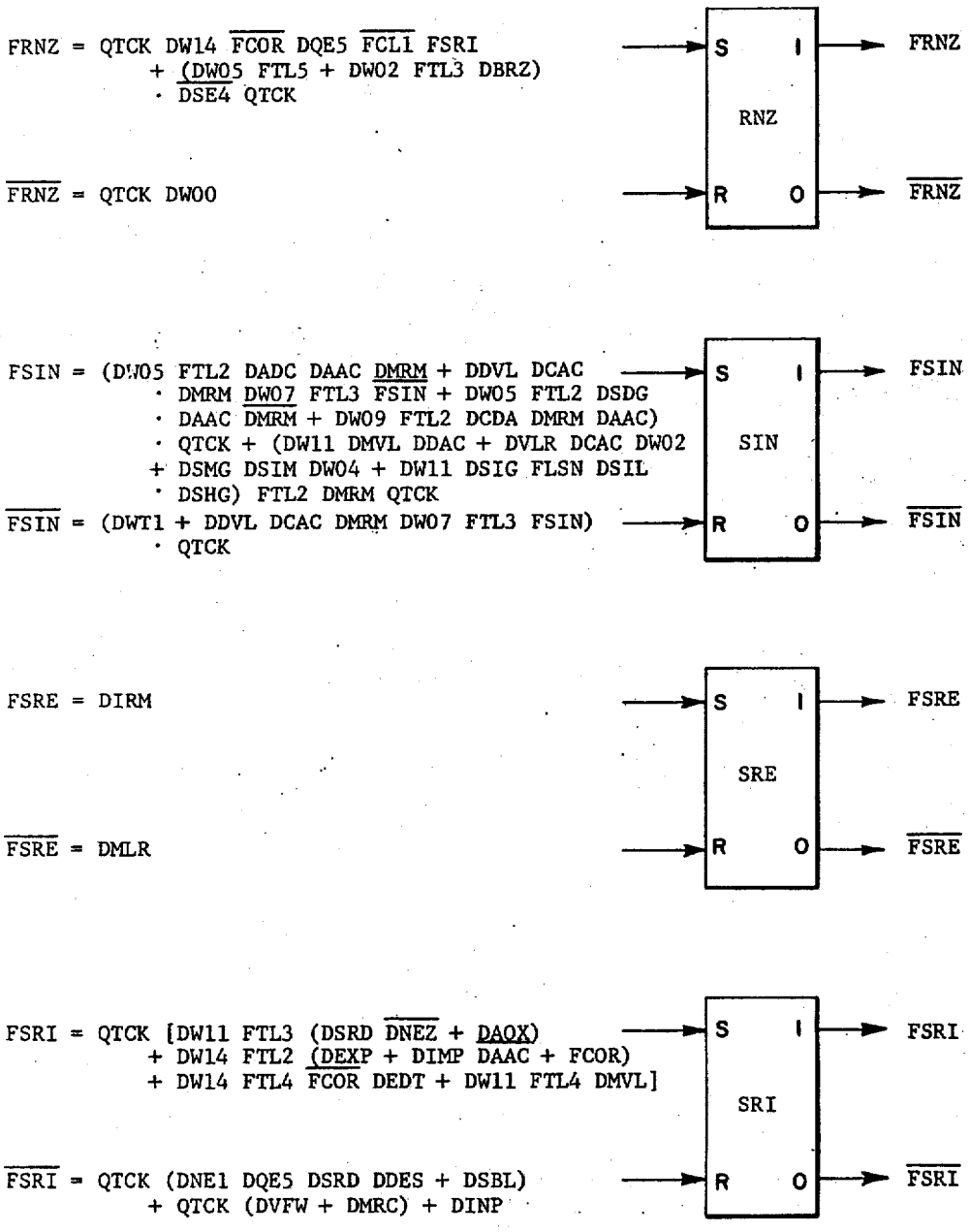
$FPSX = \{DWL4 + DMRC + DVHW + DVHF + FTL5\ DW03\ DVLR\ FFVO + DW06\ DMOV\ FCRE\ FTL5 + DW00\ FTL5 \cdot [\overline{DNWX} + DIG5 + DBCB + \overline{DAMS}\ (DLAL + DBCT)] + DW06\ DSHG\ FTL5\} \cdot QTCK$

$\overline{FPSX} = (DWT1 + DWT5 + DWT4)\ QTCK$



PROGRAM PROCESSOR

FIG.89.

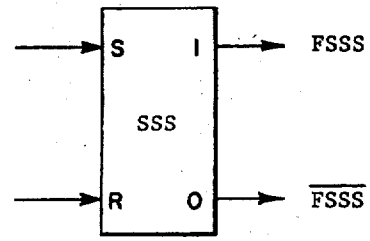


PROGRAM PROCESSOR

FIG. 90.

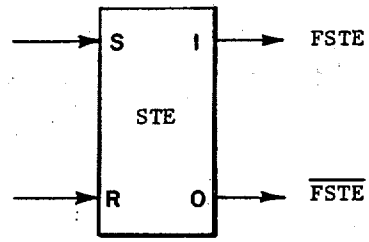
$FSSS = FSTE\ FTL3\ \overline{QTCK} + \overline{FSTT}\ FTL3\ QTCK + FSTE\ FSIE\ \overline{FSSS}$

$\overline{FSSS} = \overline{FSTE}\ \overline{FSTT}\ FTL5\ QTCK + \overline{FSTE}\ \overline{FSIE}\ FSSS$



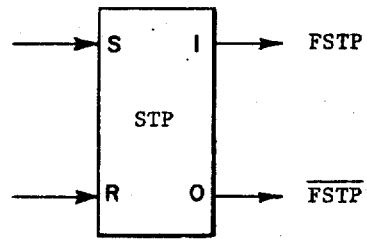
$FSTE = SSTS$

$\overline{FSTE} = \overline{SSTS} + DBRB\ FSIE$



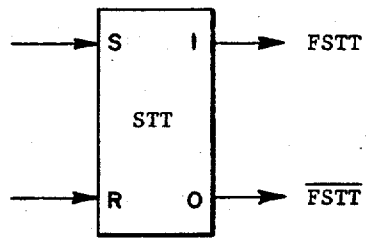
$FSTP = SSPS + FSTE$

$\overline{FSTP} = DMLR + FSTT\ \overline{FSSS}$



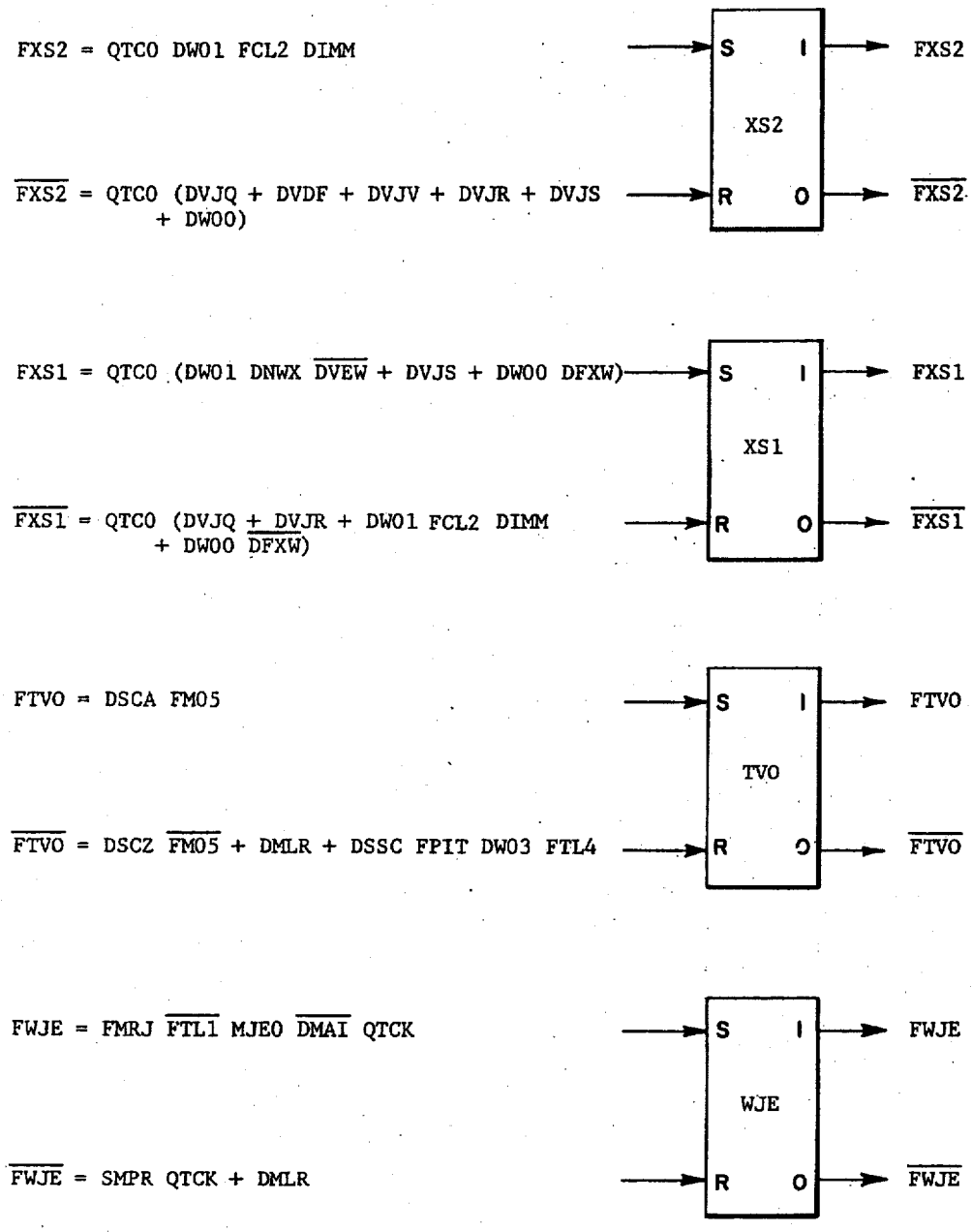
$FSTT = SSSW$

$\overline{FSTT} = \overline{SSSW} + DBRB$



PROGRAM PROCESSOR

FIG. 91.



PROGRAM PROCESSOR

FIG. 92.

LOGICAL COMBINATION SIGNALS:

$\overline{FAC1}\ \overline{FAC2}$	= DAAC
$\overline{DF00}\ \overline{DF01}\ \overline{DF02}\ \overline{DF03}\ \overline{DF04}\ \overline{DF05}\ \overline{DF06}\ \overline{DF07}\ \overline{DF08}\ \overline{DF09}\ \overline{DF10}$ $\cdot\overline{DF11}\ \overline{DF12}\ \overline{DF13}\ \overline{DF14}$	= DAAE
DMEC DIGE	= DABM
DW08 + DW14 + DVJD	= DABX
FAC1	= DAC1
FAC2	= DAC2
DW03 DVLR + DW04 DVLR DAAC + DW15 (DVLR DDAC $\overline{FCRE}$ + DSLF)	= DACD
SACS	= DACS
DW02 (DBRZ + DIMP) + DW03 (DNE4 + DLSC + DSRD + DMVL) + DW04 (DASG + DDNA + DMVL + DSHG $\overline{DLFT}$ DNL4) + DW09 + DW11 DMVL DDAC + DW14 DEXP + DW15 [DNE4 + DNZE + DSHG $\overline{DLFT}$ + DVLR ($\overline{DDAC}$ + FCRE)]	= DACU
DDLS $\overline{MC21}$ + DDLS $\overline{MS21}\ \overline{MS20}$ + DDLS $\overline{MS21}\ \overline{MS19}$	= DAD0
DDLS DDC1	= DAD1
DDLS DDC2	= DAD2
DDLS DDC3	= DAD3
FIR5 $\overline{FIR4}$ FIR3	= DADC
DACU + DACD + DBYU DTPO + DBYD DTPO + DW00	= DADU
(DW03 DSAL + DMVD + DR01 FARM FMNX) FTL1	= DAEX
DSGI $\overline{DAAC}$ + DDBI FAC2 + DTPI DDAC	= DAGI
FPL2 FPL1 $\overline{DAAC}$ + FPL2 $\overline{FPL1}$ FAC2 + $\overline{FPL2}$ FPL1 DDAC	= DAGL
DLFT DNEZ + FA0I $\overline{DLFT}$ · (DNL4 + $\overline{FN04}\ \overline{FN03}$ FN02 DNMF DDES + $\overline{DDES}$ FN04 DNAT DNMF)	= DALS
DADC FIR2	= DAMG
DMEC DIGD	= DAMI
$\overline{DW10}\ \overline{DIFP}\ \overline{DVTL}$	= DAMR
(FM15 + FM16 + FM17) $\overline{FPIA}$	= DAMS

PROGRAM PROCESSOR
FIG. 93.

LOGICAL COMBINATION SIGNALS (Cont.):

DSHC DICE = DANM

DASG + DDVL = DAOD

DANM + DRIM + DRXM = DAOX

DASG $\overline{\text{DAMG}}$ = DARS

DABM + DSBM = DASB

DASG $\overline{\text{DSGO}}$ = DASF

DSDG + DADC = DASG

DCAU + DCAD = DAUD

DACU DQDO FAC1 + DACD DQDO $\overline{\text{FAC1}}$ = DAUQ

FD00 $\underline{\text{DDBX}}$ + FM00 $\underline{\text{DMBX}}$ + $\underline{\text{DABX}}$ FPL1 $\underline{\text{DMOV}}$ + $\underline{\text{DQBX}}$ FQ00
+ $\overline{\text{FAC1}}$ DABX $\overline{\text{DMOV}}$ + $\overline{\text{DW00}}$ DPIB + $\overline{\text{DR03}}$ ($\overline{\text{DR05}}$ + $\overline{\text{FRV0}}$) DMRR = DB00

FD01 $\underline{\text{DDBX}}$ + DMOV $\underline{\text{DABX}}$ FPL2 + FM01 DMBX + DPIB + DQBX FQ01
+ $\overline{\text{FAC2}}$ DABX $\overline{\text{DMOV}}$ = DB01

FD02 DDBX + DQBX FQ02 + FM02 DMBX + DABX $\overline{\text{FA02}}$ + DPIB FPS0 $\overline{\text{FPAD}}$
+ (ZDPA + ZDPC + ZDPE + ZDPG) DMRR DDIB + FE11 DMRR $\overline{\text{DDIB}}$ = DB02

FD03 DDBX + DABX $\overline{\text{FA03}}$ + FM03 DMBX + DFBS + DPIB (FPS1 + $\underline{\text{FPAD}}$)
+ (ZDPB + ZDPC + ZDPF + ZDPG) DMRR DDIB + FE12 DMRR DDIB = DB03

FD04 $\underline{\text{DDBX}}$ + DABX FA04 + $\underline{\text{FM04}}$ DMBX + $\underline{\text{DPIB}}$ $\overline{\text{FPS2}}$ $\overline{\text{FPAD}}$
+ $\overline{\text{DB45}}$ DMRR DDIB + $\overline{\text{FE13}}$ DMRR $\overline{\text{DDIB}}$ = DB04

FD05 DDBX + DABX FA05 + FM05 DMBX + $\underline{\text{FPS2}}$ DPIB $\overline{\text{FPAD}}$
+ DB45 DMRR DDIB + FE13 DMRR $\overline{\text{DDIB}}$ = DB05

FD06 DDBX + DABX FA06 + FM06 DMBX = DB06

FD07 DDBX + DABX FA07 + FM07 DMBX = DB07

FD08 DDBX + DABX FA08 + FM08 DMBX = DB08

FD09 DDBX + DABX FA09 + FM09 DMBX = DB09

FD10 DDBX + DABX FA10 + FM10 DMBX = DB10

FD11 DDBX + DABX FA11 + FM11 DMBX = DB11

FD12 DDBX + DABX FA12 + FM12 DMBX = DB12

FD13 DDBX + DABX FA13 + FM13 DMBX = DB13

LOGICAL COMBINATION SIGNALS (Cont.):

FD14 DDBX + DABX FA14 + FM14 DMBX	= DB14
ZDPD + ZDPE + ZDPF + ZDPG	= DB45
FAC1 $\overline{\text{FAC2}}$	= DBAC
(FGRE + FLES) DBCL DIGD + DBCL DIGE $\overline{\text{FLES}}$ + $\overline{\text{FGRE}}$ DBCL DIGC	= DBCB
$\overline{\text{FIR5}}$ $\overline{\text{FIR4}}$ FIR3	= DBCL
DBRU + DBRF $\overline{\text{DBCB}}$	= DBCT
FMOD + DW14 $\overline{\text{FCOR}}$ DQE5 $\overline{\text{FCL1}}$ FSRI	= DBCU
DVTL + DW06 + DW11 + DVZG + DW12 $\overline{\text{FWRS}}$ + DW04 (DSAS + DVTM + DVLR + DMVL DCAC) + FMAI DINP DR00	= DBLO
DW11 + DW12 + DVVP + DW10 FMAN	= DBL2
DBCB $\overline{\text{FCL1}}$	= DBNX
DBCL DIGG	= DBRC
DBCL DIGD	= DBRE
DBRG + DBRL + DBRE	= DBRF
DBCL DIGC	= DBRG
DBCL DIGE	= DBRL
DBCL DIGB	= DBRM
DW01 FNXF DGEN FTL1	= DBRS
DBCL DIGA	= DBRU
DBCL DIGF	= DBRZ
DBCL FIR2 FIR1	= DBSA
$\overline{\text{FPCA}}$ $\overline{\text{FPCB}}$	= DBTO
(DBYU + DBYD) DQD0	= DBUD
DBCT DXIM	= DBXM
DW04 DSRD DLFT	= DBYD
DW04 DSHG $\overline{\text{DLFT}}$ $\overline{\text{DNL4}}$	= DBYU

PROGRAM PROCESSOR

FIG. 95.

LOGICAL COMBINATION SIGNALS (Cont.):

DMIS DIGD = DCAA

FCC0 FCC1 = DCAB

$\overline{FAC1}$ FAC2 = DCAC

DW02 DSRD $\overline{DLFT}$ + DW04 DSHG $\overline{DLFT}$ + DW15 DSRD DLFT + DW04 DMVL = DCAD

DSRD DLFT DW02 + DW04 DSRD DLFT $\overline{DNL4}$ = DCAU

DW11 FTL2 DMVL FCRE + DW14 FTL3 $\overline{FCOR}$ DVFB (FCL2 + FCL1 + DEEZ)
+ DW14 FCOR DQE5 FSRI DCEL + FMOD FTB1 FTB0
· (DMVL FCRE + DVLR $\overline{FMPL}$ DCAC $\overline{DNEZ}$ FCRE
+ DVLR FMPL DDAC $\overline{DNEZ}$ + DVLR $\overline{FMPL}$ DDAC FCRE) = DCCU

$\overline{FCC0}$ $\overline{FCC1}$ $\overline{FCC2}$ $\overline{FCC3}$ $\overline{FCC4}$ = DCCZ

DMIS DIGC = DCDA

FCC3 $\overline{FCC2}$ FCC1 FCC0 = DCE1

DCDA + DCAA + DEDT + DBRZ + DEXP + DIMP = DCEE

DSC0 $\overline{FCC0}$ $\overline{FCC1}$ FCC2 + DDB0 $\overline{FCC0}$ $\overline{FCC1}$ $\overline{FCC2}$ FCC3
+ DTP0 $\overline{FCC0}$ $\overline{FCC1}$ DCTV + DQD0 $\overline{FCC0}$ $\overline{FCC1}$ $\overline{FCC2}$ $\overline{FCC3}$ FCC4 = DCEL

DVMA + DCES + DW07 FTL0 DDVL = DCER

DW06 FTL3 DAOD + DW08 FTL4 + DW05 FTL2 $\overline{FLSN}$ $\overline{DAAC}$
+ DW09 FTL1 DVTV + DW02 FTL5 DSBM = DCES

DW07 DDVL FTL1 $\overline{FFV0}$ = DCID

MS51 $\overline{DS50}$ DCSL + $\overline{MS51}$ ($\overline{DCSL}$ + DS50) = DCL4

(DTMV + DSMV) FTL3 + DICR + DW00 DLAL $\overline{FMAN}$ FTL4 = DCLA

DWT9 + DW14 FTL3 $\overline{FCOR}$ DUSE $\overline{DEEZ}$ DVFB + DWT1 = DCLC

$\overline{DAOX}$ DW11 FTL3 + DWT1 + DW14 FTL1 + DW02 FTL1 $\overline{FIR5}$ FIR4 $\overline{FIR3}$
· $\overline{FIR2}$ $\overline{FIR1}$ = DCLQ

DVJH + DVRM + DCMC + DW14 DSGP DIMP $\overline{DDAC}$ = DCM0

DVJH + DVRM + DCMC + DW14 DSGP DIMP $\overline{DCAC}$ = DCM1

DVJH + DVRM + DCMC + DW14 DSGP DIMP $\overline{DBAC}$ = DCM2

DVJH + DCMC + DVRM + DW14 DSGP DIMP $\overline{DAAC}$ = DCM3

DVJH + DVRM + DCMC + DW02 FTL3 DIMP = DCM4

PROGRAM PROCESSOR

FIG.96.

LOGICAL COMBINATION SIGNALS (Cont.):

DR01 FMNX FTL1 ($\overline{\text{FDCW}}$ DRIT + FPRM $\overline{\text{DRIT}}$ DCH0) + DW01 FTL3 DVCA (DGEN + $\overline{\text{FE00}}$ $\overline{\text{FE01}}$ DCP0) + DMBX FMAI $\overline{\text{FMRD}}$	= DCMC
DMIS DIGB	= DCMI
DMIS DIGE	= DCMM
DCAA + DCDA	= DCPG
FIR5 FIR4 $\overline{\text{FIR3}}$ DIGH	= DCPO
DSHG $\overline{\text{MS57}}$	= DCSL
FCC2 FCC3	= DCTV
DW09 DCPG	= DCW9
FD00	= DD00
FD01	= DD01
FD02	= DD02
FD03	= DD03
FD04	= DD04
FD05	= DD05
FD06	= DD06
FD07	= DD07
FD08	= DD08
FD09	= DD09
FD10	= DD10
FD11	= DD11
FD12	= DD12
FD13	= DD13
DVLR DAAC	= DDAA
FAC1 FAC2	= DDAC
$\overline{\text{FIR1}}$ FIR0	= DDBI

PROGRAM PROCESSOR
FIG. 97

LOGICAL COMBINATION SIGNALS (Cont.):

DDBI DLSM + FPL2 $\overline{\text{FPL1}}$ DVWA + DDBI FPL2 DARS
+ DD10 $\overline{\text{FD09}}$ DSHG + $\overline{\text{FD09}}$ DDES DSHG = DDBO

DDXB + DW09 = DDBX

DAD0 + DDCS $\overline{\text{MC21}}$ = DDC0

DDCS $\overline{\text{MC15}}$ + DDLS $\overline{\text{MS15}}$ $\overline{\text{MS14}}$ + DDLS $\overline{\text{MS15}}$ $\overline{\text{MS13}}$ = DDC1

DDCS $\overline{\text{MC09}}$ + DDLS $\overline{\text{MS09}}$ $\overline{\text{MS08}}$ + DDLS $\overline{\text{MS09}}$ $\overline{\text{MS07}}$ = DDC2

DDCS $\overline{\text{MC03}}$ + DDLS $\overline{\text{MS03}}$ $\overline{\text{MS02}}$ + DDLS $\overline{\text{MS03}}$ $\overline{\text{MS01}}$ = DDC3

DVLR $\overline{\text{DCAC}}$ = DDCA

DW03 FTL5 DELS + DW04 FTL5 DAMG $\overline{\text{DLAO}}$ + FTL1 DW08 $\overline{\text{DAAC}}$
+ FTL1 DW04 (DMAD + DDDA + DARS $\overline{\text{DAAC}}$) + FTL1 DW15 DDVL = DDCD

FDCE FAEC = DDCS

DVLR DDAC = DDDA

$\overline{\text{FD13}}$ DSHG + DDVS = DDES

DW11 FTL2 FWRS = DDEV

DDES DD07 DSHG = DDFS

DDCS FLSN $\overline{\text{DCDA}}$ + DDCS DMVL = DDLS

DW15 FTL1 DSHG DD11 $\overline{\text{FD13}}$ = DDLX

DW04 DVLR $\overline{\text{DAAC}}$ = DDNA

DMVL + DVLR = DDOM

FDSX FAEC = DDST

DDCS $\overline{\text{FLSN}}$ $\overline{\text{DMVL}}$ + DDCS DCDA = DDUS

DSHC DIGG = DDVL

DDVL FCOR = DDVS

DW07 DMVL + DW05 $\overline{\text{DAMG}}$ + DW04 DAMG
+ DW15 (DMTA + DIG4 $\overline{\text{DFXW}}$ $\overline{\text{FPIA}}$)
+ DW03 (DSTG + DMOV + DSAL + DGCP $\overline{\text{DFXW}}$ $\overline{\text{FPIA}}$)
+ DW02 (DLDG + DVLR + DVSL + DEDT + DEXP + DIMP)
+ DW00 $\overline{\text{FPIA}}$ + DW01 $\overline{\text{FPIA}}$ $\overline{\text{DVTX}}$ = DDXB

(DSMV + DTMV) FTL5 + DW00 DLAL FTL0 = DEAP

PROGRAM PROCESSOR
FIG. 98.

LOGICAL COMBINATION SIGNALS (Cont.):

(DSEG + $\overline{\text{DR00}}$ $\overline{\text{DDIB}}$ $\overline{\text{FLPM}}$) $\overline{\text{FE14}}$	= DECH
DSEG $\overline{\text{FBSY}}$	= DEDF
DMIS DIGF	= DEDT
DEDT + DEXP + DIMP	= DEEI
$\overline{\text{FE00}}$ $\overline{\text{FE01}}$ $\overline{\text{FE02}}$ $\overline{\text{FE03}}$ $\overline{\text{FE04}}$ $\overline{\text{FE05}}$	= DEEZ
DW09 + DW03 + DRW0 + $\overline{\text{FWR0}}$ DR00 $\overline{\text{DSW0}}$	= DEI0
DRW1 + DVTJ + $\overline{\text{FWR1}}$ $\overline{\text{DSW1}}$ DR00	= DEI1
DVTJ + DRW2 + $\overline{\text{FWR2}}$ DR00 $\overline{\text{DSW2}}$	= DEI2
DVTJ + DRW3 + $\overline{\text{FWR3}}$ DR00 $\overline{\text{DSW3}}$	= DEI3
DEI0 DEI1 DEI2 DEI3 + DINP	= DEIN
DLDG + DSTG + DEDT	= DELS
$\overline{\text{FE04}}$ FE05	= DERM
SERS	= DERS
DSAL + DEXP	= DESA
DR01 FMNX (FEMA + FARM)	= DESC
DW03 FPRI	= DESP
DW03 FTL1 DESA + DW03 $\underline{\text{FTL3}}$ DSRD + DW01 FTL2 DIMM ($\overline{\text{FNXF}}$ + $\overline{\text{DSPB}}$ $\overline{\text{DBRC}}$ $\overline{\text{DGCP}}$)	= DESS
DESC + FESX FAEC	= DEST
DEXP DLAO	= DEXL
DSHC DIGA	= DEXP
FAC1 DD00 + $\overline{\text{FAC1}}$ $\overline{\text{FD00}}$	= DF00
FAC2 DD01 + $\overline{\text{FAC2}}$ $\overline{\text{FD01}}$	= DF01
FA02 DD02 + $\overline{\text{FA02}}$ $\overline{\text{FD02}}$	= DF02
FA03 DD03 + $\overline{\text{FA03}}$ $\overline{\text{FD03}}$	= DF03
$\overline{\text{FA04}}$ DD04 + FA04 $\overline{\text{FD04}}$	= DF04

PROGRAM PROCESSOR

FIG. 99.

LOGICAL COMBINATION SIGNALS (Cont.):

$\overline{FA05}$ DD05 + FA05 $\overline{FD05}$ = DF05

$\overline{FA06}$ DD06 + FA06 $\overline{FD06}$ = DF06

$\overline{FA07}$ DD07 + FA07 $\overline{FD07}$ = DF07

$\overline{FA08}$ DD08 + FA08 $\overline{FD08}$ = DF08

$\overline{FA09}$ DD09 + FA09 $\overline{FD09}$ = DF09

$\overline{FA10}$ DD10 + FA10 $\overline{FD10}$ = DF10

$\overline{FA11}$ DD11 + FA11 $\overline{FD11}$ = DF11

$\overline{FA12}$ DD12 + FA12 $\overline{FD12}$ = DF12

$\overline{FA13}$ DD13 + FA13 $\overline{FD13}$ = DF13

$\overline{FA14}$ FD14 + FA14 $\overline{FD14}$ = DF14

DSRD DW02 + DW03 DSHG $\overline{DLFT}$ DLAO + DW04 $\overline{DMOV}$ + DW10 + DW15 DMOV + DW04 DSRD FA0I (DLAA + $\overline{DLFT}$ DLAO $\overline{DNL4}$) = DFBS

$\overline{FTB1}$ $\overline{FTB0}$ $\overline{DNZE}$ $\overline{FCOR}$ DEDT $\overline{FCL1}$ DW14 = DFDE

$\overline{FQ00}$ FQ02 + FQ01 $\overline{FQ02}$ + FQ00 $\overline{FQ01}$ = DFXW

DCP0 + DGEN = DGCP

DW04 DSRD (DNEZ + DLFT + $\overline{DNL4}$) = DGE1

DW04 DDVL DAAC = DGE2

DW04 $\overline{DSRD}$ DDAC + DW04 $\overline{DMVL}$ $\overline{DSRD}$ + DW03 $\overline{DIMP}$ $\overline{DSLF}$ = DGE3

FSP1 FCOR DCEL DW14 FTL2 DEDT DZCP + MJE0 DW02 $\overline{DLAO}$ DIMP FTL4 FSP2 + DW02 DSRD DLFT FTL0 = DGE4

DMIS DIGH = DGEN

DW15 + DGE1 + DIML + DGE2 + DGE3 + DR01 + DR02 + DR03 + DR05 = DGEP

DDTE $\overline{DRIT}$ FTL3 $\overline{DMAI}$ + DW03 DGCP FTL2 = DGMM

DW04 DSRD DNL4 $\overline{DLFT}$ $\overline{DNEZ}$ + DW04 DMVL = DGPP

DMIS DIGA = DHLT

DW01 FTL4 DXIM DHLT = DHSM

PROGRAM PROCESSOR

FIG. 100

LOGICAL COMBINATION SIGNALS (Cont.):

DW01 DXIM DHLT	= DHXM
DW00 FTL4 DINV $\overline{\text{FMAN}}$	= DIAM
FSRE $\overline{\text{DIRM}}$	= DICR
DW03 $\overline{\text{DIMP}}$ $\overline{\text{DSAL}}$ $\overline{\text{DVLR}}$ + DR01 DCH0 $\overline{\text{FMNX}}$ FDCW $\overline{\text{DRIT}}$ + DR02 DFTC $\overline{\text{DRIT}}$ + DR02 DCH0 $\overline{\text{DRIT}}$	= DIFP
DMFM + DBSA + DCMI + DMTA + DCMM + DAOX + DASB + DMFI + DMOV + DGCP + DAMI	= DIG3
DMFM + DAMI + DSPB + DBRC + DMFI + DAOX + DASB	= DIG4
DBSA + DCMI + DMTA + DMFI + DGCP + DAMI	= DIG5
DSGI $\overline{\text{FIR2}}$	= DIGA
DDBI $\overline{\text{FIR2}}$	= DIGB
DTPI $\overline{\text{FIR2}}$	= DIGC
DQDI $\overline{\text{FIR2}}$	= DIGD
DSGI FIR2	= DIGE
DDBI FIR2	= DIGF
DTPI FIR2	= DIGG
DQDI FIR2	= DIGH
FPIA $\overline{\text{DGEN}}$ $\overline{\text{DSPB}}$ FTL4 DW00	= DIIP
DW00 FTL5 DLDG + FTL3 DLAO DARS DW05 · (DDBI FPL1 FPL2 + DTPI DPNQ + DQDI)	= DILX
DW02 DIMP DLAO	= DIML
DVEW FAOI + DVEW $\overline{\text{FM18}}$ $\overline{\text{FM19}}$ + DVTX	= DIMM
DSHC DIGB	= DIMP
DW02 DIMP $\overline{\text{DLAO}}$	= DINL
FAIM FPEF	= DINP
FIR5 FIR4 (FIR3 + FIR2 $\overline{\text{DCPO}}$)	= DINV
FEMA + FARM	= DIOA

PROGRAM PROCESSOR
FIG. 101.

LOGICAL COMBINATION SIGNALS (Cont.):

DW02 $\overline{\text{DLAO}}$ DIMP + DW04 DMVL $\overline{\text{DDAC}}$ + DW03 DSRD DLFT + DW04 DSRD DNL4 $\overline{\text{DNEZ}}$ $\overline{\text{DLFT}}$	= DIPT
DW04 + DVTL + DW03 DVLR + DRSI + DCW9 + DVVW + DVSH	= DIRK
DPSD	= DIRM
(DRRV + DR05 + DW00 + DW01 $\overline{\text{DVCA}}$ + DW02 $\overline{\text{DLSC}}$ $\overline{\text{DMOV}}$ + DW03 $\overline{\text{DIMP}}$ + DW04 + DWSM + DW08 + DW09 + DW10 + DW14 DEXP + DW15) FTL4	= DIRS
DLFT DAAC	= DLAA
DMEC DIGG	= DLAL
DSGO + DDBO DBAC + DTPO DCAC + DQDO DDAC	= DLAO
DW01 DLAL DXIM	= DLAX
DLSC $\overline{\text{FIR2}}$	= DLDG
DDFS + DDVS	= DLFT
$\overline{\text{DLFT}}$ DLAO	= DLLA
DHLI	= DLLI
DLFT DSBC	= DLSB
FIR5 $\overline{\text{FIR4}}$ $\overline{\text{FIR3}}$	= DLSC
DLSC DLAO	= DLSL
DLSC + DAMG	= DLSM
DVGJ + ($\overline{\text{FM00}}$ + $\overline{\text{FE00}}$) DANM + DRIM $\overline{\text{FM00}}$ $\overline{\text{FE00}}$ + DRXM DVVV	= DLSR
DVGH + FM00 FE00 DANM + (FM00 + FE00) DRIM + DRXM DVVQ	= DLSS
FMUJ $\overline{\text{DIPT}}$ + DGEP MJE0 $\overline{\text{DIPT}}$	= DM24
DMVL DDAC	= DMAD
FMAI	= DMAI
FMBU	= DMBU
DW07 DDC0	= DMDC
$\overline{\text{FIR5}}$ FIR4 FIR3	= DMEC
DMEC DIGB	= DMFI

PROGRAM PROCESSOR

FIG. 102.

LOGICAL COMBINATION SIGNALS (Cont.):

DMEC DIGA = DMFM

DMOV DVCA = DMIC

DSSC DW03 FPAD FPIT FTL4 = DMIR

$\overline{\text{FIR5}}$ $\overline{\text{FIR4}}$ $\overline{\text{FIR3}}$ = DMIS

$\overline{\text{FISP}}$ $\overline{\text{FRCK}}$ FMAN FNBY FMRE = DMLR

DMVL $\overline{\text{DDAC}}$ = DMND

(FMEM + FMNX) $\overline{\text{FLPM}}$ = DMNS

SMPR = DMPR

DW00 FTL3 $\overline{\text{FPIA}}$ = DMQX

DW10 FTL5 = DMRC

$\overline{\text{FM04}}$ FM05 = DMRM

FAIM FPSA + DW00 FTL4 DHLT $\overline{\text{DAMS}}$ = DMSE

DW08 FTL1 + DW06 FTL3 $\underline{\text{DDVL}}$ + DW06 FTL2 DMOV + DW04 $\overline{\text{DVHN}}$ FTL1
+ DW02 FTL1 DMOV ($\overline{\text{DSHG}}$ + FD12) + DW01 FTL2 ($\overline{\text{DVCA}}$ + $\overline{\text{DMTA}}$)
+ DW05 FTL1 DARS DAGL = DMSS

DMEC DIGC = DMTA

DW11 FTL2 DMVL DDAC = DMVC

DW15 DMOV FTL1 = DMVD

DW06 DMOV FTL2 = DMVE

DW02 DMOV FTL1 = DMVF

DSHC DIGH = DMVL

DCLM (DR02 + DR04 + DTMV + DW00 + DW01 + DW03 + DW08) = DMWS

DW01 FTL5 [$\overline{\text{DVEW}}$ DNWX + DXNW + DVDF + DBXM + DLAX + DVCD + DBNX
+ DVCA (DMFM + DGCP + DSPB + DMFI + DMOV)] + DW06 FTL3 DASG
+ DW05 FTL1 (DMTA + DAGL DAMG + DAGI $\underline{\text{DARS}}$) + DW06 FTL5 DMOV
+ DW03 FTL1 DSAL + DW04 FTL3 (DDVL + DAAC) + DW15 FTL1
+ DW01 FTL5 DVCA DBRC FCRE = DMXR

DW01 FTL5 FXS2 $\overline{\text{FXS1}}$ DNWX = DNAI

DMIS DIGG = DMOV

PROGRAM PROCESSOR

FIG. 103.

LOGICAL COMBINATION SIGNALS (Cont.):

$\overline{FN02}$ FN03 = DNAT

DW15 FTL3 DVLR DDAC $\overline{FCRE}$ + FMOD $\overline{FMPL}$ DVLR $\overline{FTB1}$ $\overline{FTB0}$ (DCAC + $\overline{FCRE}$)
+ DW11 FSRI DQE5 DSRD + DW11 FSRI DSHG $\overline{DDES}$ + DW11 FSRI DAOX = DNCS

FN00 $\overline{FN01}$ $\overline{FN02}$ $\overline{FN03}$ $\overline{FN04}$ = DNE1

DNMF DEDT (FN02 + FN03) = DNE4

$\overline{FN00}$ $\overline{FN01}$ $\overline{FN02}$ $\overline{FN03}$ $\overline{FN04}$ = DNEZ

DDES DNTS $\overline{FN04}$ + $\overline{DDES}$ ($\overline{FN04}$ + $\overline{FN03}$) = DNL4

$\overline{FN00}$ $\overline{FN01}$ = DNMF

DNS4 + DNS8 + DNS2 + DNS6 + DNS0 + DNSL = DNR0

DW14 FTL2 DIMP = DNS0

DW14 FTL1 DTPO DAAC = DNS2

DW14 FTL1 DSGO = DNS4

DW14 FTL1 DQDO DAAC = DNS6

DW14 FTL1 DDBO DAAC = DNS8

DW11 DAOX FTL3 = DNSL

$\overline{FN02}$ $\overline{FN03}$ = DNTS

FM15 FM16 FM17 $\overline{FPIA}$ = DNWX

DNEZ DEDT = DNZE

DW00 FTL3 $\overline{FPIA}$ $\overline{FMAN}$ + DW01 FTL3 $\overline{FXS1}$ $\overline{FXS2}$ $\overline{FPIA}$ = DPCU

DPSC + FPAD = DPDC

DSGI ($\overline{FPL1}$ + $\overline{FPL2}$) DAMG + DDBI $\overline{FPL2}$ DAMG + DTPI $\overline{FPL2}$ $\overline{FPL1}$ DAMG = DPGI

(RERX + RPTS) $\overline{REBX}$ = DPIO

FPIA (DW00 + DW01 + DW03 + DW15) = DPIB

SPMS = DPMS

FPL2 + FPL1 = DPNQ

FPRI FTL4 DW03 + DSCZ FPSA $\overline{FM02}$ = DPSP

FPSX DPSR = DPST

PROGRAM PROCESSOR

FIG. 104.

LOGICAL COMBINATION SIGNALS (Cont.):

FAEC + DR01 FPRM DRIT FMNX = DPSR

DW01 FPIA DBCL DIGH = DPSU

DVTX DW01 + DW03 DGCP DFXW + DW15 DIG4 DFXW = DQBX

FIR1 FIR0 = DQDI

DQDI DARS + $\overline{\text{FPL2}}$ $\overline{\text{FPL1}}$ (DARS + DCEE) + DQDI DLSM
+ $\overline{\text{FD10}}$ $\overline{\text{FD09}}$ DDES DSHG + DDVL + DMVL = DQDO

FQ02 $\overline{\text{FQ01}}$ FQ00 = DQE5

DWT9 + DWT1 = DRBG

DSSC DW03 FTL4 = DRCF

DW04 DSRD DSIG DSIM FLSN $\overline{\text{DNL4}}$ + DW11 DSRD DSIG DSIL FLSN = DRE4

DW14 FTL5 $\overline{\text{FCOR}}$ $\overline{\text{FRNZ}}$ FCL1 FCL2 DEDT = DRE5

DSHC DIGD = DRIM

DW03 FTL4 $\overline{\text{FM04}}$ FPRI + FLPB = DRPI

DW03 FTL4 FM04 FPRI = DRSP

DW15 (DIG4 + DMTA + DVTC + DMOV + DSRD + DEDT)
+ DW05 (DSAS + DASF + DDCA) + DW01 [DMIC + DXIM $\overline{\text{FNXF}}$ DVSL
+ DXIM (DBRM + DLSC + DSAS + DASF + DEXP + DIMP + DLAL + DCPG
+ DEDT + DBRZ + DBRU + DMVL) + DVVH] = DRW0

DW06 (DSAS + DASF + DSHG + DVSY) + DW10 $\overline{\text{FMAN}}$
+ DW15 (DAOX + DSRD + DVTC + DVTD)
+ DW02 [DBRM + DIML + DAOX + DASB + DCMM + DMFM
+ DBRZ (FRNZ + DLAO)] + DW03 (DSAL + DLSL + DVLR FFVO
+ DMVL + DVSY + DGCP + DSRD + DEXL + DDAA) = DRW1

DW06 (DMOV + DSHG) + DW07 (DMVL + DDCA)
+ DW04 (DSAS + DMAD + DVLR + DASF $\overline{\text{FCOR}}$ DLAO + DVVL + DSNZ)
+ DW15 (DAOX + DVTD + DVLR DDAC FCRE) = DRW2

DW15 (DVTD + DSRD + DMOV + DVLR + DAOX) + DW11 DSMP + DW10 $\overline{\text{FMAN}}$
+ DW09 (DCMI + DCMM + DVBJ + $\overline{\text{FLSN}}$) = DRW3

DSHC DIGF = DRXM

MS00 = DS50

MS04 + DDCS + DRE4 + DSS4 + DVLC + DMVC = DS54

PROGRAM PROCESSOR
FIG. 105.

LOGICAL COMBINATION SIGNALS (Cont.):

$(\overline{DW03} + \overline{DVLR} + \overline{DAAC} + \overline{FSIN})\ (\overline{DW04} + \overline{FCOR} + \overline{FSIN} + \overline{DASG} + \overline{DAAC})$
$\cdot\ (\overline{DW04} + \overline{FCOR} + \overline{FSIN} + \overline{DASG} + \overline{DAAC})$
$\cdot\ (\overline{DW07} + \overline{DCAC} + \overline{DDVL} + \overline{FMOD} + \overline{DMRM})$
$\cdot\ (\overline{DW15} + \overline{DVLR} + \overline{DCAC} + \overline{FE04} + \overline{FE05})$
$\cdot\ (\overline{DW04} + \overline{DSRD} + \overline{DAAC} + \overline{DSIG} + \overline{FSIN} + \overline{DALS})$
$\cdot\ (DDCS + MS05 + DRE4 + DSS4) = DS55$

$MS10 + DDCS = DS60$

$MS11 + DDCS = DS61$

$MS16 + DDCS = DS66$

$MS17 + DDCS = DS67$

$MS22 + DDCS = DS72$

$MS23 + DDCS = DS73$

$DMEC\ DIGH = DSAL$

$DASG\ DSGO = DSAS$

$DDUS\ MC21 = DSB0$

$DDUS\ MC15 = DSB1$

$DDUS\ MC09 = DSB2$

$DDUS\ MC03 = DSB3$

$DBAC + DSGO = DSBC$

$DW11\ DNE1\ DSHG\ \overline{DDES}\ FSRI + DAOX\ DW11\ DNE1 = DSBL$

$DMEC\ DIGF = DSBM$

$DCPO\ DW01\ FTL5\ FE00\ DVCA = DSCA$

$DCPO\ DW01\ FTL5\ FE01\ DVCA = DSCZ$

$FIR5\ FIR4\ \overline{FIR3}\ \overline{FIR2} = DSDG$

$DW00\ FTL0 + DW01\ FTL0\ \overline{DVRF} + DW01\ FTL4\ DVCA\ DMTA$
$+ DW03\ FTL5\ (DMOV + DDDA) + DW04\ FTL0\ DAMG\ FDSX$
$+ DW06\ FTL2\ DMOV + DW00\ FTL4 + DW01\ FTL5\ \overline{DNAI}\ (\overline{DIG3} + \overline{DVCA}) = DSDT$

$MS69\ MS70\ MS71\ DS72\ DS73\ DSE9 = DSE4$

PROGRAM PROCESSOR
FIG. 106.

LOGICAL COMBINATION SIGNALS (Cont.):

(DVLR + DS50 MS51 MS52 MS53 DS54 DS55)
· MS06 MS57 MS58 MS59 DS60 DS61 MS12 MS63 MS64 = DSE5

MS65 DS66 DS67 MS18 DSE5 = DSE9

DR00 FNXF DGEN = DSEG

DVJC + DVJB + DWT9 + DWT6 = DSES

$\overline{\text{FIR1}}$ $\overline{\text{FIR0}}$ = DSGI

DSGI DLSM + FPL2 FPL1 DVWA + (FD10 + $\overline{\text{DDES}}$) DSHG DD09 = DSGO

FISP $\overline{\text{FRCK}}$ + FSRE = DSGP

DESH (DSHX + DRH0) = DSH0

DESH DWS0 $\overline{\text{DWS1}}$ DDIB $\overline{\text{DRH0}}$ = DSH1

DESH (DWS1 $\overline{\text{DWS0}}$ DDIB $\overline{\text{DRH0}}$ + WCCS $\overline{\text{FGST}}$ $\overline{\text{DDIB}}$ $\overline{\text{DW12}}$ + DSDF) = DSH2

DESH (DWCS $\overline{\text{FGST}}$ $\overline{\text{DDIB}}$ $\overline{\text{DSCS}}$ + DWS0 DWS1 DDIB $\overline{\text{DRH0}}$
+ DCCS FGST DDIB DW12) = DSH3

$\overline{\text{FIR5}}$ FIR4 $\overline{\text{FIR3}}$ = DSHC

DDES FD14 DSHG + DDVL FCOR $\overline{\text{FMPL}}$ = DSIG

DLAA + DLA0 $\overline{\text{DLFT}}$ = DSIL

DLSB + DLA0 $\overline{\text{DLFT}}$ = DSIM

DSRD DLFT = DSLF

DSRD DSIG $\overline{\text{DNL4}}$ FLSN = DSMG

DW03 FTL2 DSAL + DW15 FTL2 $\overline{\text{DEDT}}$ $\overline{\text{DAOX}}$ + DR01 FMNX FARM FTL2 = DSML

DSRD + DMVL = DSMP

DMSS + DWT7 + DWT9 + DVTS + DMVE + DIMR + DWL6 = DSMS

DW06 DMOV = DSMV

DSHG DNEZ = DSNZ

DBCL DIGH = DSPB

$\overline{\text{DMLR}}$ $\overline{\text{FISP}}$ $\overline{\text{FRCK}}$ DVRS $\overline{\text{FMWR}}$ $\overline{\text{DSGP}}$ = DSPG

DSHC DIGC = DSHG

PROGRAM PROCESSOR
FIG. 107.

LOGICAL COMBINATION SIGNALS (Cont.):

DW00 FTL0 $\overline{\text{FAIM}}$ (DBRM + DBRZ + DBRC + DSPB)
+ DR01 FPRM DRIT FMNX FTL4 FWRT + DW01 DXIM FTL5 DBCT
+ DW00 FTL4 DAMS DBCT FMAN = DSPT

DSHG + DDVS = DSRD

DW04 FCOR DASF + DW04 DMVL $\overline{\text{DCAC}}$ + DW04 DSRD DNL4 = DSRK

DW04 FTL2 DSRD DAAC DSIG DALS = DSS4

DCP0 $\overline{\text{FE00}}$ $\overline{\text{FE01}}$ DW03 = DSSC

DLSC FIR2 = DSTG

MS69 MS70 MS71 = DSTZ

DW00 $\overline{\text{FMAN}}$ (DSRD + DSAL + DMOV + DIG5 + DNWX DVSL + DAMS $\overline{\text{DVSL}}$ $\overline{\text{DPGI}}$)
+ DW02 (DLSC + DEPX + DCMM + DMFM + DINL + DAOX + DMOV + DVLR
+ DASB) + DW06 DDOM + DW14 (DEXP + DNE4 + DNZE) = DSW0

DW00 [FMAN + DVVB + $\overline{\text{DNWX}}$ (DCMM + DAOX + DMOV + DASB)]
+ DW05 DSGO DASG FCOR + DW04 (DVLR + DVVL)
+ DW01 [DVTF DXIM + DVCA (DIG4 + DGCP + DMOV)
+ DXIM FNXF (DMFM + DCMM + DAOX + DASB + DMOV)] = DSW1

DW00 (DVST + $\overline{\text{DAMS}}$ $\overline{\text{FMAN}}$ DVSS $\overline{\text{DPGI}}$) + DW01 (DVSV + DMIC + DVCH)
+ DW02 (DEDT + DVLR) + DW03 (DIMP + DVSW + DMVL + DSRD + DVSY)
+ DW11 (DMAD + DAOX) = DSW2

DW02 DEDT + DW00 (FMAN + DVST + $\overline{\text{DAMS}}$ DVTG)
+ DW01 [DVSV + DVTG DXIM + DVCA (DIG4 + DCMI + DCMM
+ DGEN FBSY)] + DW06 DVLR + DW03 (DVSW + DIMP) = DSW3

DTP0 $\overline{\text{FAC2}}$ = DTAF

DTP0 FAC2 = DTAT

DW02 DMOV = DTMV

FIR1 $\overline{\text{FIR0}}$ = DTPI

DTPI DPNQ DARS + $\overline{\text{FPL2}}$ FPL1 ($\overline{\text{DQDI}}$ DARS + DCEE)
+ DTPI DLSM + $\overline{\text{FD10}}$ DD09 DDES DSHG = DTP0

DW04 $\overline{\text{DDES}}$ DAAC DD08 DSHG FTL3 FAOI = DTSB

$\overline{\text{FCL2}}$ $\overline{\text{FCL1}}$ = DUSE

DCPG DLAO = DVBJ

PROGRAM PROCESSOR

FIG. 108.

LOGICAL COMBINATION SIGNALS (Cont.):

FNXF DIMM = DVCA

$\overline{DVEW}$ $\overline{DNWX}$ $\overline{FCL1}$ = DVCB

DVCB DIG5 = DVCD

DVSS DXIM + DVCA DVTH = DVCH

DW01 FCL1 $\overline{DNWX}$ $\overline{DVEW}$ = DVDF

DW07 DDVL FTB1 $\overline{FTB0}$ FMOD $\overline{(DNEZ}$ + DDAC DDC0)
+ DW04 FTL2 $\overline{(DVLR}$ + DCAC) + FTL2 DR01 (FPRM + FEMA) FMNX DRIT
+ $\overline{FMNX}$ DR13 FTL4 = DVEF

DR01 FMNX $\overline{FDCW}$ = DVEQ

DW01 DBRC DVCA = DVEV

FXS1 + $\overline{FXS2}$ = DVEW

FSP1 + FSP2 = DVFB

DW06 FTL0 + DW05 FTL0 + DW07 FTL0 $\overline{DDVL}$ + DW06 FTL2 DAOD
+ DW11 $\overline{DMVL}$ $\overline{DAOX}$ FTL2 = DVFK

DW14 FTL4 $\overline{FCOR}$ DUSE $\overline{DEDT}$
+ DW14 FTL4 FCOR $\overline{FRNZ}$ $\overline{DERM}$ FCL1 FCL2 DEDT = DVFL

DQE5 (DW14 $\overline{FCOR}$ $\overline{DIMP}$ + DW11 DMVL + DW14 FN00 $\overline{FN01}$
+ DW14 DQE5 (DSG0 DCAB $\overline{FCC2}$ $\overline{FCC3}$ + DDB0 DCAB FCC2 $\overline{FCC3}$
+ DTP0 DCE1 + DQD0 DCAB DCTV) + DW14 DQE5 DCEL = DVFW

DRSI + DBL0 = DVGC

DR04 + $\overline{DDIE}$ DBL2 $\overline{DINP}$ + FLCP $\overline{DDIE}$ (DRRV + DR05) = DVGD

FM00 DEEI + FE00 DSMP = DVGH

$\overline{FM00}$ DEEI + $\overline{FE00}$ DSMP = DVGJ

DW00 FTL5 DPGI = DVHF

DARS DAGL $\overline{FCOR}$ + FMPL DMVL FAC2 = DVHN

PROGRAM PROCESSOR
FIG. 109.

LOGICAL COMBINATION SIGNALS (Cont.):

```
DW01 FTL5 (DBNX + DBXM + DVCD + DBCL DIGH DVCA + DLAX + DXNW
    + DVDF) + DW02 FTL5 (DMFM + DIML + DCMM + DAOX + DASB
    + DBCL DIGB DMRM + DBRZ FRNZ) + DW03 FTL5 (DEXL + DLSL
    + DGCP + DSAL + DDAA) + DW04 FTL5 (DMAD + DSNZ
    + FCOR DASG DLAO) + DW09 FTL5 (FLSN + DVBJ + DCMM + DCMI)
    + DW15 FTL5 (DBRC FCRE + DMFM + DAMI + DVTD + DMTA + DMFI
    + DAOX + DASB)                                          = DVHW

DW15 FTL1 DSRD + DW14 FTL4 + DW01 FTL5 (DBXM + DLAX + DBNX + DXNW)
    + DW02 DMOV FTL1 + DW03 FTL1 DEXP DSAL + DW08 FTL1      = DVHZ

DW14 FTL2 FCOR FM01 FM03 FM04 FM00 FM02 + DW11 FTL4 DMVL
    + DW15 FTL5 DBCL DIGH + DW15 FTL5 DBRC FCRE + DW06 FTL3 DAOD
    + DW14 FTL2 FCOR FM01 FM03 FM04 FM02 FM05               = DVJB

FTL1 DW05 + DW01 FTL5 DVCA DBRC FCRE                        = DVJC

DW02 (DSTG + DBRM + DBRZ + DMOV)
    + DW03 (DVLR + DEDT + DEXP + DIMP + DLDG + DMND + DSRD DLLA)
    + DW04 [DARS + DMVL + DVLR + DSRD (FAOI + DLFT DAAC
    + DLFT DNL4)] + DW04 DSHG DLFT DLAO + DW05 DAMG + DW07 DVLR
    + DW15 (DEDT + DSRD + DVLR)                             = DVJD

DCLM [DW00 + DW01 + DW02 + DW04 + DW05 (DAAE + DMTA) + DW07
    + DW08 + DCW9 + DW14 + DR00
    + DW03 (DIMP + DEXP + DSAL + DVLR)]                     = DVJH

FRNZ FTL5 FSIN DSE9 DW05 FLSN DASG DLAO                     = DVJK

DW01 DIG3 DIMM DUSE                                         = DVJQ

DW01 FCL1 FCL2 DIMM                                         = DVJR

DW01 DIMM DVEW                                              = DVJS

DW01 DNWX FCL1 DIG3 DVEW                                    = DVJV

DMVL FMOD FTB1 FTB0                                         = DVKB

DW02 FTL0 DVLR DDAC                                         = DVLC

DDVL FCOR                                                   = DVLR

DW05 FTL2 DADC DAAC DMRM DERM + DW05 FTL2 DADC DAAC DMRM DERM
    + DW05 FTL2 DSDG DAAC DMRM DERM
    + DW05 FTL2 DSDG DAAC DMRM DERM                         = DVMA

DW09 FTL1 (DCPG DAAC + DCMI + DCMM)                         = DVMC

FSIN DSE4 DDC0 DCDA                                         = DVMF
```

PROGRAM PROCESSOR

FIG. 110.

LOGICAL COMBINATION SIGNALS (Cont.):

FLSN $\overline{\text{FSIN}}$ $\overline{\text{DDC0}}$ DCDA + MC23 DCAA	= DVMG
DCMI $\overline{\text{MC14}}$ $\overline{\text{DSE5}}$ + DCMM $\overline{\text{MC23}}$ $\overline{\text{DSE4}}$	= DVMJ
$\overline{\text{DSE4}}$ $\overline{\text{MC23}}$ DCAA + $\overline{\text{FSIN}}$ $\overline{\text{DSE4}}$ DDC0 DCDA	= DVML
FLSN FSIN $\overline{\text{DDC0}}$ DCDA	= DVMM
DCMI MC14 + DCMM MC23	= DVMP
DW09 FTL4 (DVMF + DVMG + $\overline{\text{FLSN}}$ FSIN + DVMJ)	= DVMR
DW09 FTL4 (DVML + DVMM + $\overline{\text{FLSN}}$ $\overline{\text{FSIN}}$ + DVMP)	= DVMS
FM05 $\overline{\text{FM04}}$ FM03 FM02 FM01 $\overline{\text{FM00}}$ + $\overline{\text{FM05}}$ $\overline{\text{FM04}}$ FM03 FM01	= DVRA
$\overline{\text{FM05}}$ $\overline{\text{FM04}}$ FM03 FM02 FM01 FM00 + FM05 $\overline{\text{FM04}}$ FM03 $\overline{\text{FM02}}$ FM01 $\overline{\text{FM00}}$	= DVRB
$\overline{\text{FM05}}$ $\overline{\text{FM04}}$ FM03 $\overline{\text{FM02}}$ FM01 FM00 DW14 FTL2 $\overline{\text{FCOR}}$ DEDT	= DVRC
$\overline{\text{FM05}}$ $\overline{\text{FM04}}$ FM03 FM02 FM01 $\overline{\text{FM00}}$ DW14 FTL2 $\overline{\text{FCOR}}$ DEDT	= DVRD
$\overline{\text{FM05}}$ $\overline{\text{FM04}}$ FM03 $\overline{\text{FM02}}$ FM01 $\overline{\text{FM00}}$ DW14 FTL2 $\overline{\text{FCOR}}$ DEDT	= DVRE
FXS1 $\overline{\text{FXS2}}$ + $\overline{\text{DVEW}}$	= DVRF
DW11 FTL3 $\overline{\text{DAOX}}$	= DVRM
DW02 FTL1 DVLR	= DVRN
$\overline{\text{DMBU}}$ + DMDA ($\overline{\text{DIRK}}$ + DSRK)	= DVRS
$\overline{\text{DMBU}}$ FTL0 $\overline{\text{FISP}}$	= DVRV
FTL2 $\overline{\text{FISP}}$ ($\overline{\text{DMBU}}$ + DMDA DSRK)	= DVRW
DW02 DSRD $\overline{\text{DNEZ}}$	= DVSA
DSRD DNL4	= DVSB
DW03 DVLR $\overline{\text{DAAC}}$	= DVSC
DW11 DGEN FWRS	= DVSF
DW15 DSRD	= DVSG
DW05 DMTA	= DVSH
DW04 (DVSB + DMVL $\overline{\text{FAC2}}$)	= DVSK
DMFM + DCMM + DAOX + DASB	= DVSL

PROGRAM PROCESSOR
FIG. III

LOGICAL COMBINATION SIGNALS (Cont.):

DW12 DGEN FWRS	=	DVSM
DVLR DCAC DW05	=	DVSP
DSAS + DASF + DMVL + DSRD	=	DVSS
$\overline{\text{DNWX}}$ $\overline{\text{FMAN}}$ DMOV	=	DVST
DXIM $\overline{\text{FNXF}}$ DMOV	=	DVSV
DEXP $\overline{\text{DLAO}}$	=	DVSW
DMOV FCRE	=	DVSY
DW11 DMVL DDAC	=	DVTA
DW07 DVLR DCAC	=	DVTB
DVLR ($\overline{\text{DDAC}}$ + $\overline{\text{FCRE}}$)	=	DVTC
DEDT $\overline{\text{FCOR}}$ DLAO	=	DVTD
DW14 (DEXP + DIMP)	=	DVTE
DLSC + DEXP + DIMP + DSAL + DEDT + DBRM + DBRZ + DSRD	=	DVTF
DCPG + DSRD	=	DVTG
DMFM + DBSA + DMTA + DASB + DMFI + DAMI	=	DVTH
DVTH DW15	=	DVTJ
DW02 DIMP	=	DVTL
DASF $\overline{\text{FCOR}}$	=	DVTM
($\overline{\text{DMRM}}$ + FE05 + $\overline{\text{FE04}}$) (DMRM + FE04 $\overline{\text{FE05}}$)	=	DVTR
DW01 FTL5 DVCA DMTA	=	DVTS
DCMI + DCMM	=	DVTV
FXS1 $\overline{\text{FXS2}}$ DFXW	=	DVTX
DW03 FTL2 DVLR DDAC	=	DVTY
$\overline{\text{DAMS}}$ DVTF + $\overline{\text{DNWX}}$ DMEC DIGA	=	DVVB
DBRF DXIM + $\overline{\text{FCL1}}$ DBCB DBRF	=	DVVH

PROGRAM PROCESSOR
FIG. 112

LOGICAL COMBINATION SIGNALS (Cont.):

DSRD $\overline{DNEZ}$ $\overline{DLFT}$ $\overline{DSBC}$ + DSRD $\overline{DNEZ}$ $\overline{DLFT}$ DLAO DNL4
+ DSRD DNEZ DLFT DNL4 (DSGO + DDBO DAAC + DTPO DBAC
+ DQDO DCAC) = DVVL

DW01 DVCA (DIG4 + DMOV + DCPO + DGEN FBSY) + DW07 DDOM
+ DW00 FMAN + DW14 (DIMP + DEDT) + DW01 DHLT DXIM
+ (DSAS + DASF + DVLR) (DW05 + DW06)
+ DW03 (DVSW + DIMP + DMVL + DSRD) = DVVP

FM00 $\overline{FE00}$ + $\overline{FM00}$ FE00 = DVVQ

FM00 FE00 + $\overline{FM00}$ $\overline{FE00}$ = DVVV

DW02 DSRD = DVVW

DSGI DARS + DCEE = DVWA

DW15 DSRD ($\overline{DCCZ}$ + $\overline{DLFT}$) = DVZG

$\overline{FWR3}$ $\overline{FWR2}$ $\overline{FWR1}$ $\overline{FWR0}$ DR00 = DW00

$\overline{FWR3}$ $\overline{FWR2}$ $\overline{FWR1}$ FWR0 DR00 = DW01

$\overline{FWR3}$ $\overline{FWR2}$ FWR1 $\overline{FWR0}$ DR00 = DW02

$\overline{FWR3}$ $\overline{FWR2}$ FWR1 FWR0 DR00 = DW03

$\overline{FWR3}$ FWR2 $\overline{FWR1}$ $\overline{FWR0}$ DR00 = DW04

$\overline{FWR3}$ FWR2 $\overline{FWR1}$ FWR0 DR00 = DW05

$\overline{FWR3}$ FWR2 FWR1 $\overline{FWR0}$ DR00 = DW06

$\overline{FWR3}$ FWR2 FWR1 FWR0 DR00 = DW07

FWR3 $\overline{FWR2}$ $\overline{FWR1}$ $\overline{FWR0}$ DR00 = DW08

FWR3 $\overline{FWR2}$ $\overline{FWR1}$ FWR0 DR00 = DW09

FWR3 $\overline{FWR2}$ FWR1 $\overline{FWR0}$ DR00 = DW10

FWR3 $\overline{FWR2}$ FWR1 FWR0 DR00 = DW11

FWR3 FWR2 $\overline{FWR1}$ $\overline{FWR0}$ DR00 = DW12

FWR3 FWR2 FWR1 $\overline{FWR0}$ DR00 = DW14

FWR3 FWR2 FWR1 FWR0 DR00 = DW15

DW02 $\overline{DDES}$ = DWDB

PROGRAM PROCESSOR

FIG.113.

LOGICAL COMBINATION SIGNALS (Cont.):

DW02 DDES $\overline{\text{DNL4}}$	=	DWDN
FWEN	=	DWEN
DW04 FTL1	=	DWL1
DW09 FTL5	=	DWL3
DW06 FTL1	=	DWL4
DW02 FTL5	=	DWL5
DW00 FTL2	=	DWL6
DW02 DNL4	=	DWNL
DW12 FTL4 FWRS	=	DWRR
DW07 DMVL	=	DWSM
DW00 FTL1	=	DWT1
DW04 FTL3	=	DWT3
DW15 FTL3	=	DWT4
DW01 FTL1	=	DWT5
DW08 FTL4	=	DWT6
DW09 FTL1	=	DWT7
DW11 FTL3	=	DWT8
DW07 FTL1	=	DWT9
DIMM DUSE + FXS2 $\overline{\text{FXS1}}$ $\overline{\text{DNWX}}$ $\overline{\text{FCL1}}$	=	DXIM
$\overline{\text{FCL2}}$ FCL1 DIMM $\overline{\text{FNXF}}$ + DIG5 DXTA	=	DXNW
$\overline{\text{FNXF}}$ DIMM DUSE	=	DXTA
FMUJ + $\overline{\text{FMRJ}}$ FSP2 (DGE2 + DIML + DGE1 + DW03 DMVL + DW03 DSRD $\overline{\text{DLFT}}$)	=	DZB2
$\overline{\text{FM00}}$ $\overline{\text{FM01}}$ $\overline{\text{FM02}}$ $\overline{\text{FM03}}$ $\overline{\text{FM04}}$ $\overline{\text{FM05}}$ + FM00 FM01 $\overline{\text{FM02}}$ FM03 FM04	=	DZCP

PROGRAM PROCESSOR

FIG. 114.

FULL ADDER INPUT SIGNALS:

FM00 FMSX + DW14 FCOR FSP1 FSP2 + DW04 DDST FCOR FIR0 = DX00

FM01 FMSX + DW14 FCOR FSP1 FSP2 + DW04 DDST FCOR FIR1 = DX01

FM02 FMSX + DW14 FCOR FSP2 $\overline{\text{FSP1}}$ = DX02

FM03 FMSX + DW14 FCOR FSP2 = DX03

FM04 FMSX $\overline{\text{DDCS}}$ + DW14 $\overline{\text{FCOR}}$ FCL2 FCL1 + DW14 FCOR FSP1 $\overline{\text{FSP2}}$ = DX04

FM05 FMSX $\overline{\text{DDCS}}$ + DW14 FCOR FSP2 = DX05

FM06 FMSX = DX06

FM07 FMSX = DX07

FM08 FMSX = DX08

FM09 FMSX = DX09

FM10 FMSX $\overline{\text{DDCS}}$ = DX10

FM11 FMSX $\overline{\text{DDCS}}$ = DX11

FM12 FMSX = DX12

FM13 FMSX = DX13

FM14 FMSX = DX14

FM15 FMSX $\overline{\text{DIWC}}$ + DR01 DCC5 + DMOV FIRE = DX15

FM16 FMSX $\overline{\text{DDCS}}$ $\overline{\text{DIWC}}$ + DR01 DCC6 = DX16

FM17 FMSX $\overline{\text{DDCS}}$ = DX17

FM18 FMSX = DX18

FM19 FMSX = DX19

FM20 FMSX = DX20

FM21 FMSX = DX21

FM22 FMSX $\overline{\text{DDCS}}$ = DX22

FM23 FMSX $\overline{\text{DDCS}}$ = DX23

PROGRAM PROCESSOR

FIG. 115.

FULL ADDER INPUT SIGNALS (Cont.):

FE00 DEST + FD00 FDSX FAEC + FCC0 FCS2 = DY00

FE01 DEST + FD01 FDSX FAEC + FCC1 FCS2 = DY01

FE02 DEST + FD02 FDSX FAEC + FCC2 FCS2 = DY02

FE03 DEST + FD03 DDST + FCC3 FCS2 = DY03

FE04 DEST + FD04 DDST + FP04 DPST = DY04

FE05 DEST + FD05 DDST + FP05 DPST = DY05

FE06 DEST + FD06 DDST + FP06 DPST = DY06

FE07 DEST + FD07 DDST + FP07 DPST = DY07

FE08 DEST + FD08 DDST + FP08 DPST = DY08

FE09 DEST + DD09 DDST + FP09 DPST = DY09

FE10 DEST + DD10 DDST + FP10 DPST = DY10

FE11 DEST + FD11 DDST + FP11 DPST = DY11

FE12 DEST + DD12 DDST + FP12 DPST = DY12

FE13 DEST + DD13 DDST + FP13 DPST = DY13

FE14 DEST + FD14 DDST + FP14 DPST = DY14

FE15 DEST + DCCS DR01 $\overline{\text{FMNX}}$ + DR03 $\overline{\text{FMEM}}$ + $\overline{\text{DMOV}}$ FIRE = DY15

FE16 DEST = DY16

FE17 DEST + $\overline{\text{DCCS}}$ DR01 $\overline{\text{FMNX}}$ $\overline{\text{DCC5}}$ $\overline{\text{DCC6}}$ = DY17

FE18 DEST + FCC0 FAEC FCS3 = DY18

FE19 DEST + FCC1 FAEC FCS3 = DY19

FE20 DEST + FCC2 FAEC FCS3 = DY20

FE21 DEST + FCC3 FAEC FCS3 = DY21

FE22 DEST = DY22

FE23 DEST = DY23

PROGRAM PROCESSOR
FIG. 116.

FULL ADDER INPUT SIGNALS (Cont.):

DRCA + (FCAY FAEC)	=	DZ00
$\overline{MC05}$ + $\overline{DDC3}$	=	DZ06
$\overline{MC11}$ + $\overline{DDC2}$	=	DZ12
$\overline{MC17}$ + $\overline{DDC1}$	=	DZ18

CLOCK GENERATOR INPUT SIGNALS:

QCCK $\overline{DSGP}$	=	QCCD
DSPG $\overline{FMRE}$ $\overline{FMRD}$ + QCCD	=	QCCK

GATED CLOCK DISTRIBUTION DRIVER INPUT SIGNALS:

QCCK FRCK	=	QCLK
QCCK $\overline{FRCK}$	=	QCKC

GATED CLOCK AMPLIFIER INPUT SIGNALS:

DCER QCLK	=	QCER
(DW11 FTL3 DMVL DDAC + DW04 FTL0 DSRD + DW03 FTL3 DSRD + DVRN) · QCLK	=	QCNX
DEAP QCLK	=	QEAX
QCLK (DVEV FTL5 FCRE)	=	QEEX
DWL6 QCLK	=	QMIX
(DW02 FTL0 DSRD + DW14 FSRI DQE5 + DNCS + DVKB) QCLK	=	QNCD
(DW02 FTL1 DSRD + DW03 FTL0 DVLR DAAC) QCLK	=	QNCX
(DW11 FTL2 DMVL DBAC + DW15 FTL2 DVLR DDAC) QCLK	=	QQNX
(DVLC + DW01 FTL5 DSHG + DW00 FTL4 DSHG + DMVC) QCLK	=	QSCX
QCLK DSDT	=	QSDX
[DVEV $\overline{MC23}$ $\underline{FTL4}$ + $\underline{DW08}$ $\underline{FTL3}$ + DHSM + DVFK + DW11 $\underline{DBAC}$ $\underline{DAOX}$ $\underline{DCAC}$ $\underline{FTL2}$ + DVKB + FTL4 (DW01 $\overline{DVEW}$ + DLAX + DW01 DXNW + DW00 + DW02 + DSMV)] · QCLK	=	QSEX

PROGRAM PROCESSOR

FIG. 117.

GATED CLOCK AMPLIFIER INPUT SIGNALS (Cont.):

(FSRI $\overline{\text{DEDT}}$ + $\overline{\text{FCL1}}$ FSRI) QCLK	= QSHE
FSRI QCLK	= QSHM
[DVEF + DW14 FTL2 FCOR DZCP DCEL + DVFL + DSML + DW03 FTL2 (DDAA + DEXP)] QCLK	= QSMA
(DVEF + DSML) QCLK	= QSMB
[DVEF + DW15 FTL2 (DVLR + DSRD + DMFM + DBRC + DASB)] QCLK	= QSMC
DSPT QCLK	= QSPX
QCKC DVRV	= QTC0
QCKC DVRW	= QTC2
QCLK	= QTCK
QCLK ($\overline{\text{FTL5}}$ + $\overline{\text{FSRI}}$ $\overline{\text{FMOD}}$)	= QTLP
QCKC $\overline{\text{FISP}}$	= QTRK
QCKC	= QTSP

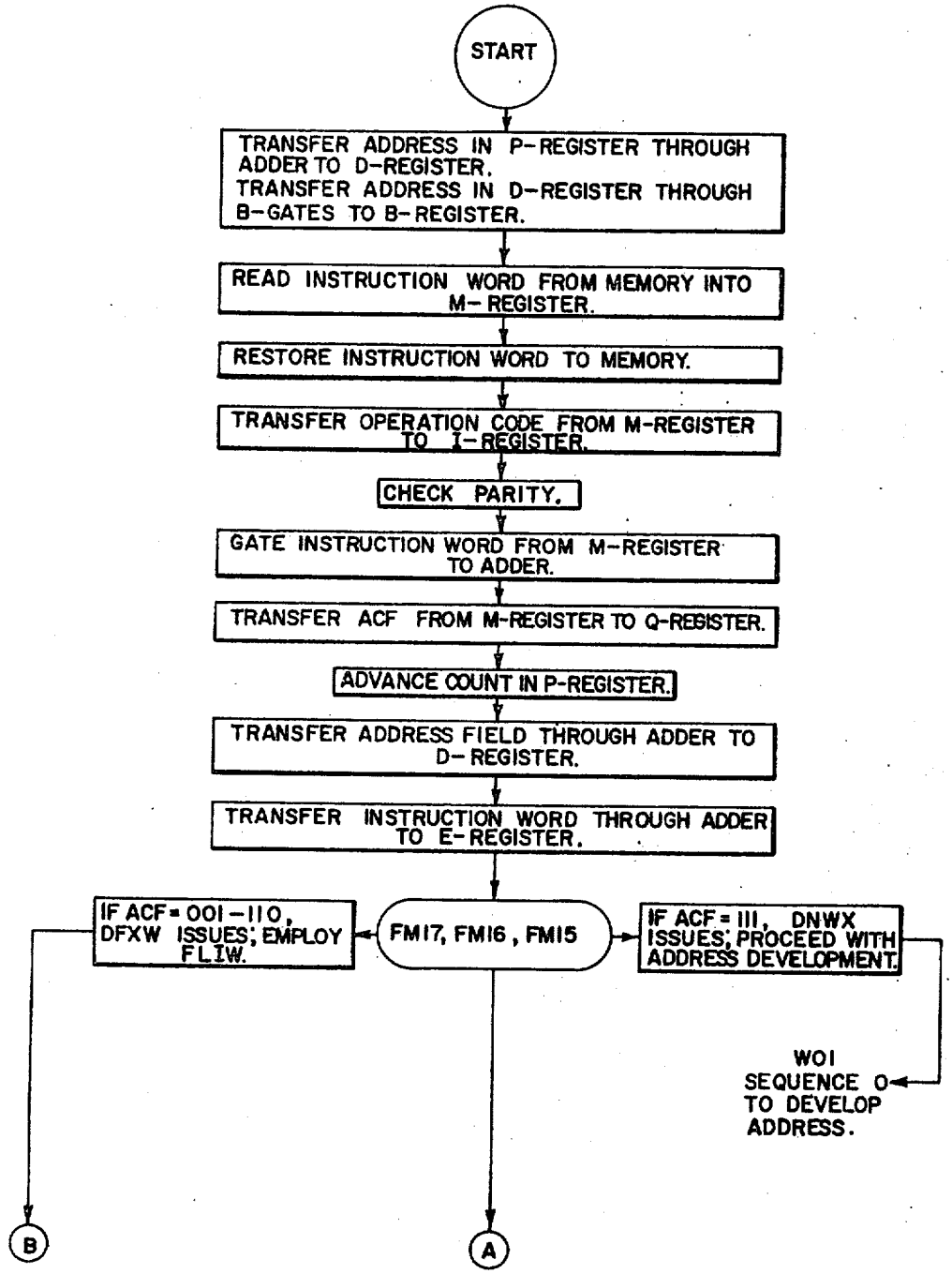
PROGRAM PROCESSOR
FIG. 118.

FIG. 119a
TIMING DIAGRAM - W00 BLOCK

TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
TL0			$\overline{FPIA}\Delta DDBX$	Instruction address to Memory. DDBX is present during entire W00 block.
QTCK	MRD			Initiate read operation in Memory.
		RNZ		
			QSDX	Address to D-Register from Adder.
TL1		CL1, CL2, NXF, CRE, DSX		
			DCM0, DCM1, DCM2, DCM3, DCLQ, DCLC	Clear M-, Q- and CC-Registers.
QTCK	LSN			Assume like-signs.
		ESX, COR, SP1, SP2, CAY, TB0, TB1, SIN, PSX		
		RCK		Stop Program Processor Clock Generator.
TL2				Wait for DMDA to start Program Processor Clock Generator.
QTRK	RCK			Start Program Processor Clock Generator.
QTCK	MWR			Initiate write operation to restore instruction word to Memory.
	MSX			Instruction word to Adder.
			QMIX	Operation code to I-Register.
			$(\overline{FM15}+FM16+FM17)\cdot\overline{FPIA}\Delta DAMS$	DAMS issues if ACF of instruction word = 1-7.
			$FM15\cdot\overline{FM16}\cdot FM17\cdot\overline{FPIA}\Delta DNWX$	DNWX issues if ACF of instruction word = 7.

FIG. 119b
TIMING DIAGRAM - W00 BLOCK (Cont.)

TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
TL3			$\overline{FPIA} \cdot \overline{FMAN} \Delta DPCU$	P-Register count up.
			$\overline{FPIA} \Delta DMQX$	ACF to Q-Register.
QTCK				
TL4				
QTCK			QSEX	Instruction word from Adder to E-Register.
			QSDX	Address field of instruction word to D-Register.
TL5	$DIG3 \cdot \overline{DNWX} \Delta NXF$			Set NXF if two-address instruction and ACF ≠ 7.
QTCK	(DNWX+DIG5) ΔPSX			Address of next P-sequence word from P-Register to Adder.
		RCK		Stop Program Processor Clock Generator to wait for memory synchronizing signal.
		MSX		
TL0				
QTC0	DFXWΔXS1			Set XS1 if ACF = 1-6 (decoded from Q-Register).
		AC1, AC2		Set Accumulator Counter Register to address accumulator word A location in Memory.
		$\overline{DFXW} \Delta XS1$		
		XS2		
		AOI		
QTRK	RCK			Start Program Processor Clock Generator on synchronizing signal DMBU from Memory.



FLOW DIAGRAM
WOO BLOCK OF MICRO-OPERATIONS

FIG. 120a

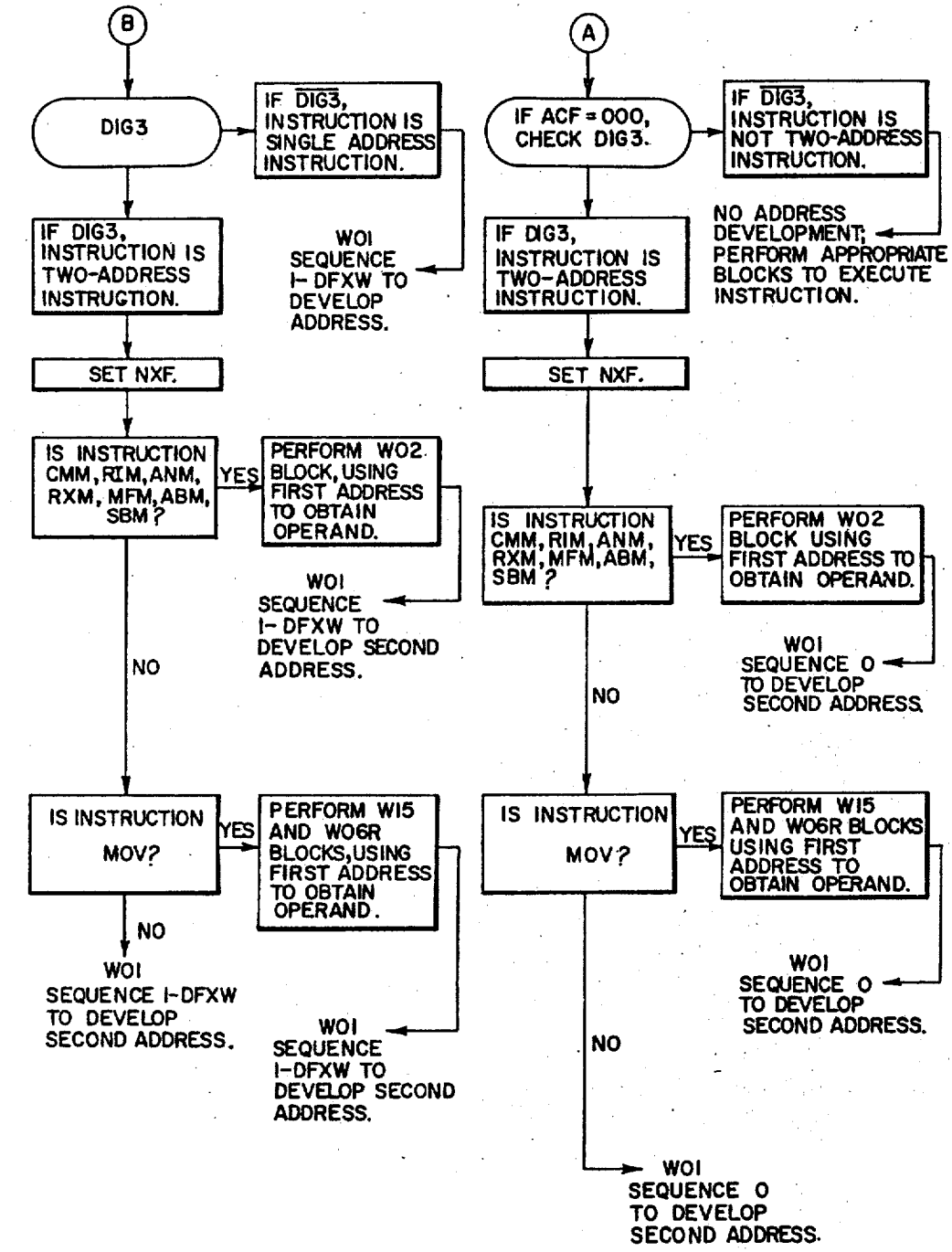

B
DIG3
IF $\overline{DIG3}$, INSTRUCTION IS SINGLE ADDRESS INSTRUCTION.
WOI SEQUENCE I- DFXW TO DEVELOP ADDRESS.
IF DIG3, INSTRUCTION IS TWO-ADDRESS INSTRUCTION.
SET NXF.
IS INSTRUCTION CMM, RIM, ANM, RXM, MFM, ABM, SBM?
YES
PERFORM WO2 BLOCK, USING FIRST ADDRESS TO OBTAIN OPERAND.
WOI SEQUENCE I- DFXW TO DEVELOP SECOND ADDRESS.
NO
IS INSTRUCTION MOV?
YES
PERFORM WI5 AND WO6R BLOCKS, USING FIRST ADDRESS TO OBTAIN OPERAND.
NO
WOI SEQUENCE I-DFXW TO DEVELOP SECOND ADDRESS.
WOI SEQUENCE I-DFXW TO DEVELOP SECOND ADDRESS.
A
IF ACF = 000, CHECK DIG3.
IF $\overline{DIG3}$, INSTRUCTION IS NOT TWO-ADDRESS INSTRUCTION.
NO ADDRESS DEVELOPMENT; PERFORM APPROPRIATE BLOCKS TO EXECUTE INSTRUCTION.
IF DIG3, INSTRUCTION IS TWO-ADDRESS INSTRUCTION.
SET NXF.
IS INSTRUCTION CMM, RIM, ANM, RXM, MFM, ABM, SBM?
YES
PERFORM WO2 BLOCK USING FIRST ADDRESS TO OBTAIN OPERAND.
WOI SEQUENCE 0 TO DEVELOP SECOND ADDRESS.
NO
IS INSTRUCTION MOV?
YES
PERFORM WI5 AND WO6R BLOCKS USING FIRST ADDRESS TO OBTAIN OPERAND.
WOI SEQUENCE 0 TO DEVELOP SECOND ADDRESS.
NO
WOI SEQUENCE 0 TO DEVELOP SECOND ADDRESS.

FIG. 120b

FIG. 121a
TIMING DIAGRAM - W01 BLOCK

TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
TL0			$\overline{FPIA}(\overline{FXS1}+FXS2+\overline{DFXW})\Delta DDBX$	Sequences 0, 1-$\overline{DFXW}$, 2, 3: Address in D-Register to Memory. DDBX is a binary 1 during entire W01 block.
			$\overline{FXS2}\cdot FXS1\cdot DFXW\Delta DQBX$	Sequence 1-DFXW: ACF in Q-Register to Memory. DQBX is a binary 1 during entire W01 block.
	QTCK	MRD		Initiate read operation in Memory.
			$(\overline{FXS2}\cdot\overline{FXS1}+FXS2\cdot FXS1)\Delta QSDX$	Sequences 0, 3: Address to D-Register from Adder.
TL1		DSX		
			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	QTCK	ESX, CAY, PSX, MSX		
		RCK		Stop Program Processor Clock Generator.
TL2				Wait for DMDA to start Program Processor Clock Generator.
	QTRK	RCK		Start Program Processor Clock Generator.
	QTCK	$(\overline{DVCA}+\overline{DMTA})\Delta MSX$		Auxiliary word from M-Register to Adder.
		$(\overline{DIG4}\cdot\overline{DGCP}+\overline{DVCA})\Delta MWR$		Initiate write operation in Memory to restore auxiliary word.
		$DIMM\cdot(\overline{FNXF}+\overline{DSPB}\cdot\overline{DBRC}\cdot\overline{DGCP})\Delta ESX$		Word in E-Register to Adder if DIMM issues and second address of two-address instruction is not being sought unless instruction SPB, BRC, GEN or CPO; index added to tentative address.

FIG. 121b

TIMING DIAGRAM - W01 BLOCK (Cont.)

TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
TL3	$\overline{FXS2}\cdot\overline{FXS1}\cdot DNWX\Delta CL1$ $(\overline{FXS2}\cdot\overline{FXS1}+FXS2\cdot FXS1)\cdot FM21\Delta CL2$			Sequence 0: Continuation of indexing if ACF of P-Sequence AMS Word = 7. Sequences 0, 3: Tentative address is employed to obtain Indirect Address Word if bit 21 of a P-Sequence AMS Word or an Indirect AMS Word is a binary 1.
		$\overline{FXS2}\cdot\overline{FXS1}\cdot DNWX\Delta CL1$ $(\overline{FXS2}\cdot\overline{FXS1}+FXS2\cdot FXS1)\cdot FM21\Delta CL2$		CL1 and CL2 remain reset except for above conditions.
QTCK			$\overline{FXS2}\cdot\overline{FXS1}\cdot\overline{FPIA}\Delta DPCU$	Sequence 0: Advance count in P-Register.
TL4				
QTCK			$(FXS2\cdot\overline{FXS1}+DXNW)\Delta QSEX$	Sequence 2: Indirect Address Word from Adder to E-Register.
TL5	$FXS2\cdot\overline{FXS1}\cdot DNWX\Delta DSX$			Sequence 2: D-Register to Adder if ACF of Indirect Address Word = 7 indicating that Indirect AMS Word occupies next sequential location.
QTCK	$FXS2\cdot\overline{FXS1}\cdot DNWX\Delta CAY$ $(FCL1\cdot\overline{DNWX}\cdot FXS2\cdot FXS1+DXNW)\Delta PSX$			Sequence 2: CAY set if ACF of Indirect Address Word = 7. P-Register to Adder if CL1 set to indicate continuation of indexing with P-Sequence AMS Word.
		RCK		Stop Program Processor Clock Generator to wait for synchronizing signal from Memory.
		DXNWΔESX		ESX reset if continuation of indexing with P-Sequence AMS Word.

FIG. 121c
TIMING DIAGRAM - W01 BLOCK (Cont.)

TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
		(FXS2·$\overline{FXS1}$·DNWX +DXNW+FCL1·$\overline{DNWX}$ ·$\overline{FXS2}$·$\overline{FSX1}$+$\overline{FXS2}$ ·FXS1·DNWX·$\overline{FCL1}$ ·DIG5)ΔMSX		Sequence 2: MSX reset if continuation of indexing with Indirect AMS Word or if instruction is one of instruction group 5 instructions. MSX also reset if continuation of indexing with P-Sequence AMS Word.
			($\overline{FXS2}$+FXS1+$\overline{DNWX}$) ·(DIG3+DVCA) ΔQSDX	Sequences 0, 1-$\overline{DFXW}$, 3: Adder to D-Register if not two-address instruction or if two-address instruction and have not yet obtained second address.
TL0				
QTC0	[($\overline{FXS2}$+FXS1) ·$\overline{DIMM}$+FXS2 ·FXS1·DNWX] ΔXS1			Sequences 0, 1-$\overline{DFXW}$, 3: XS1 set if final first address or, in two-address instruction, if final second address not yet obtained. Sequence 2: XS1 set if indexing to be continued with Indirect AMS Word.
	FCL2·DIMM ΔXS2			XS2 set if address development to be continued with Indirect Address Word.
	(DIG3·DIMM ·$\overline{DUSE}$+$\overline{DNWX}$ ·FCL1·$\overline{DIG3}$ ·FXS2·FXS1) ΔNXF			Set NXF if two-address instruction with development of first address complete and development of second address is to be initiated. Sequence 2: Set NXF if two-address instruction and development of first address does not require Indirect AMS Word.
	($\overline{FXS2}$+FXS1) ·$\overline{FM19}$·FM18 ·$\overline{FAOI}$ΔAOI			Sequences 0, 1-$\overline{DFXW}$, 3: Set AOI to indicate that auxiliary word is index pointer and next word read from Memory will be index.
		(DIG3·DIMM·DUSE +FCL1·$\overline{FCL2}$ ·DIMM+FCL2 ·DIMM)ΔXS1		XS1 reset if: a) two-address instruction with development of first address complete and development of second address to be initiated; b) if continuation of indexing with P-Sequence AMS Word; or c) if index is to be obtained by indirect addressing.

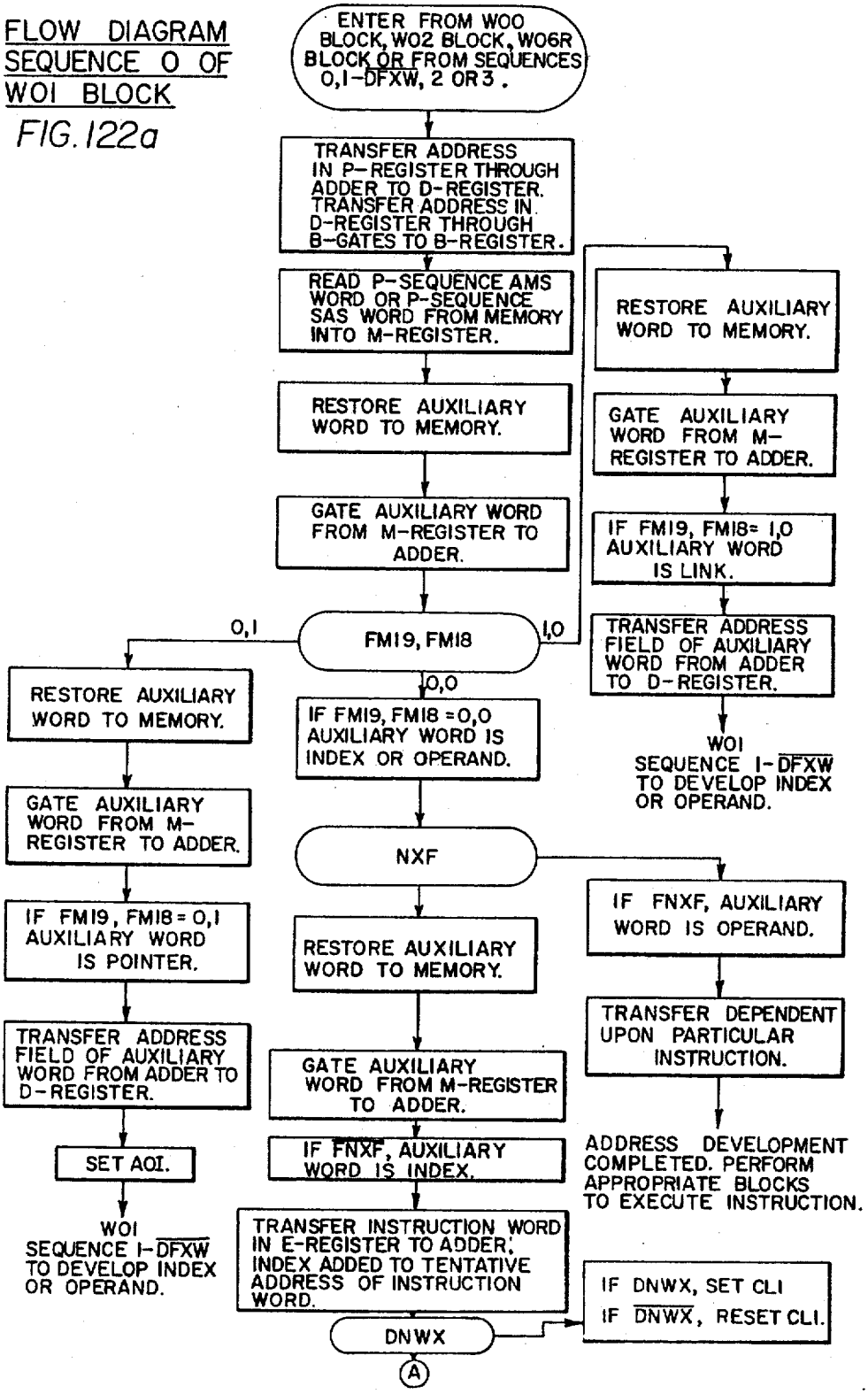
FIG. 121d

TIMING DIAGRAM - W01 BLOCK (Cont.)

TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
		(DIG3·DIMM ·DUSE+$\overline{DNWX}$ ·FCL1·FXS2 ·$\overline{FXS1}$+FCL1 ·$\overline{FCL2}$·DIMM +($\overline{FXS2}$·FXS1) ·$\overline{DIMM}$+FXS2 ·$\overline{FXS1}$·$\overline{DIG3}$ ·DNWX·FCL1) ΔXS2		XS2 reset if: a) two-address instruction with development of first address complete and development of second address to be initiated; b) Sequence 2: indexing to be continued with P-Sequence AMS Word or two-address instruction and development of first address complete; c) Sequences 0, 1-DFXW, 3: development of first address not yet completed, or d) indexing to be continued with P-Sequence AMS Word.
		DIMMΔAOI		AOI reset when DIMM issues.
QTRK	RCK			Start Program Processor Clock Generator.

FLOW DIAGRAM SEQUENCE 0 OF WOI BLOCK

FIG. 122a



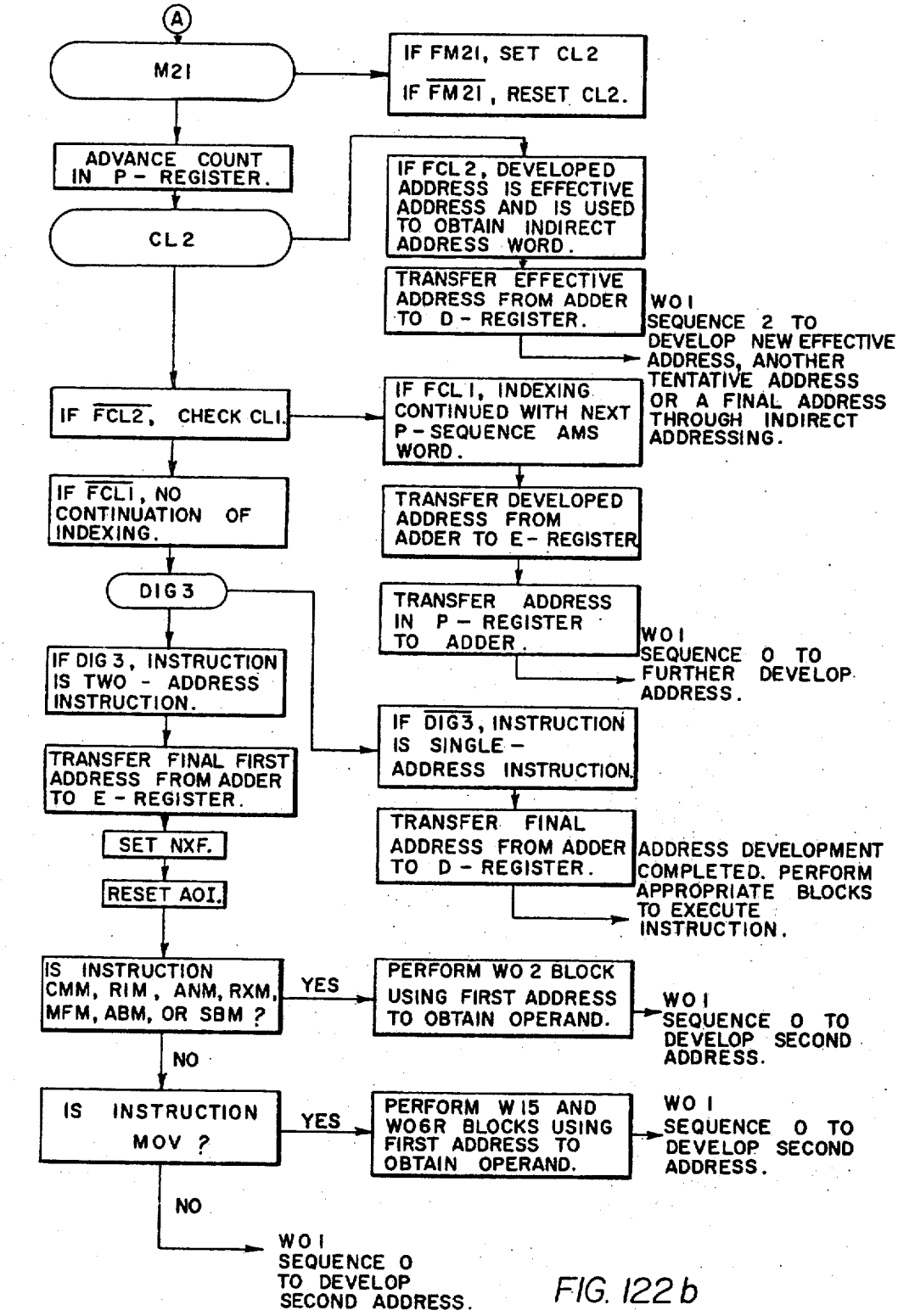

A
M21
IF FM21, SET CL2
IF $\overline{FM21}$, RESET CL2.
ADVANCE COUNT IN P - REGISTER.
CL2
IF FCL2, DEVELOPED ADDRESS IS EFFECTIVE ADDRESS AND IS USED TO OBTAIN INDIRECT ADDRESS WORD.
TRANSFER EFFECTIVE ADDRESS FROM ADDER TO D - REGISTER.
WO I SEQUENCE 2 TO DEVELOP NEW EFFECTIVE ADDRESS, ANOTHER TENTATIVE ADDRESS OR A FINAL ADDRESS THROUGH INDIRECT ADDRESSING.
IF $\overline{FCL2}$, CHECK CL1.
IF FCL1, INDEXING CONTINUED WITH NEXT P - SEQUENCE AMS WORD.
IF $\overline{FCL1}$, NO CONTINUATION OF INDEXING.
TRANSFER DEVELOPED ADDRESS FROM ADDER TO E - REGISTER
DIG 3
TRANSFER ADDRESS IN P - REGISTER TO ADDER.
WO I SEQUENCE 0 TO FURTHER DEVELOP ADDRESS.
IF DIG 3, INSTRUCTION IS TWO - ADDRESS INSTRUCTION.
IF $\overline{DIG3}$, INSTRUCTION IS SINGLE - ADDRESS INSTRUCTION.
TRANSFER FINAL FIRST ADDRESS FROM ADDER TO E - REGISTER.
TRANSFER FINAL ADDRESS FROM ADDER TO D - REGISTER.
ADDRESS DEVELOPMENT COMPLETED. PERFORM APPROPRIATE BLOCKS TO EXECUTE INSTRUCTION.
SET NXF.
RESET AOI.
IS INSTRUCTION CMM, RIM, ANM, RXM, MFM, ABM, OR SBM ?
YES
PERFORM WO 2 BLOCK USING FIRST ADDRESS TO OBTAIN OPERAND.
WO I SEQUENCE 0 TO DEVELOP SECOND ADDRESS.
NO
IS INSTRUCTION MOV ?
YES
PERFORM W15 AND WO6R BLOCKS USING FIRST ADDRESS TO OBTAIN OPERAND.
WO I SEQUENCE 0 TO DEVELOP SECOND ADDRESS.
NO
WO I SEQUENCE 0 TO DEVELOP SECOND ADDRESS.

FIG. 122b

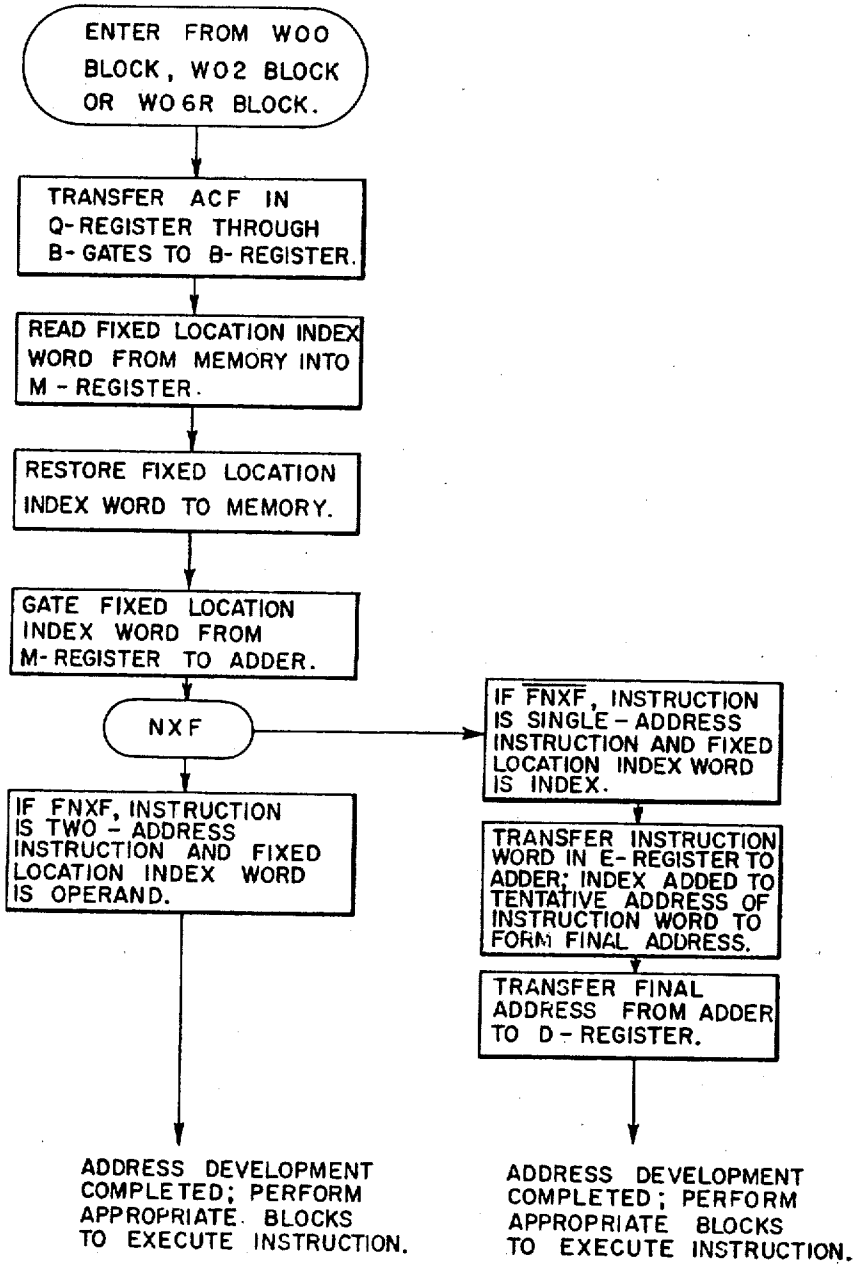
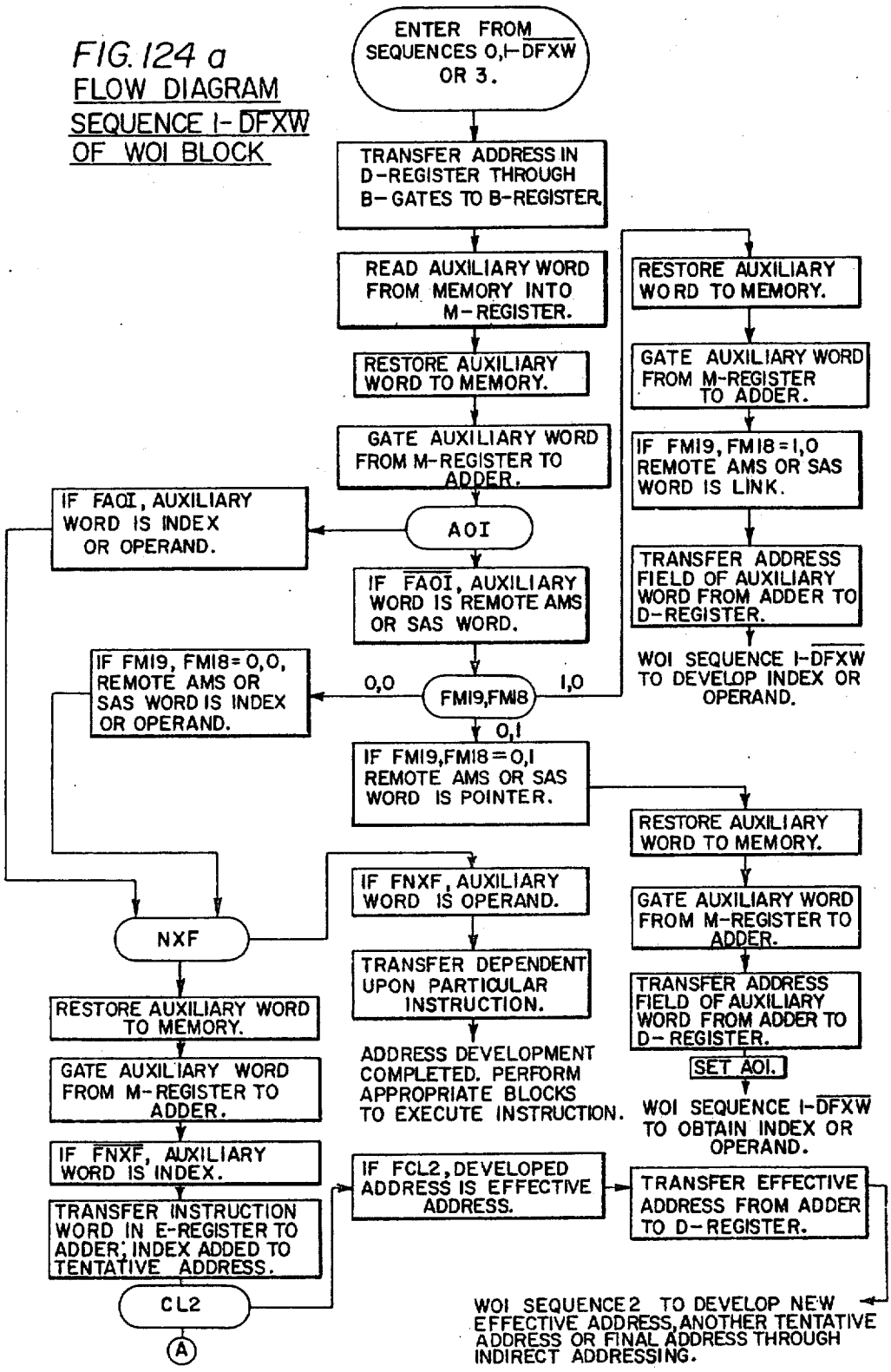


FIG. 123
FLOW DIAGRAM SEQUENCE I-DFXW OF WOI BLOCK

FIG. 124 a
FLOW DIAGRAM
SEQUENCE I-$\overline{\text{DFXW}}$
OF WOI BLOCK



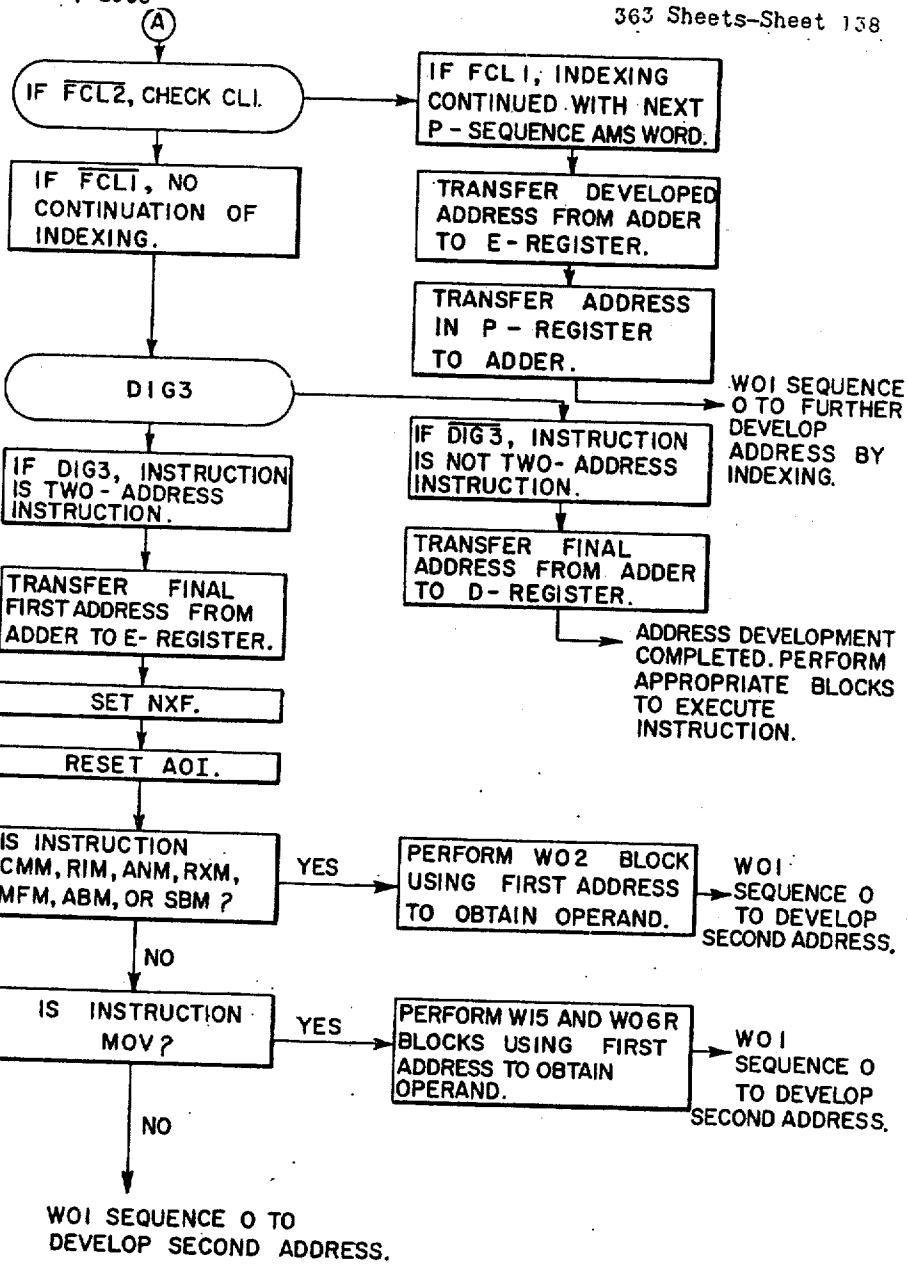
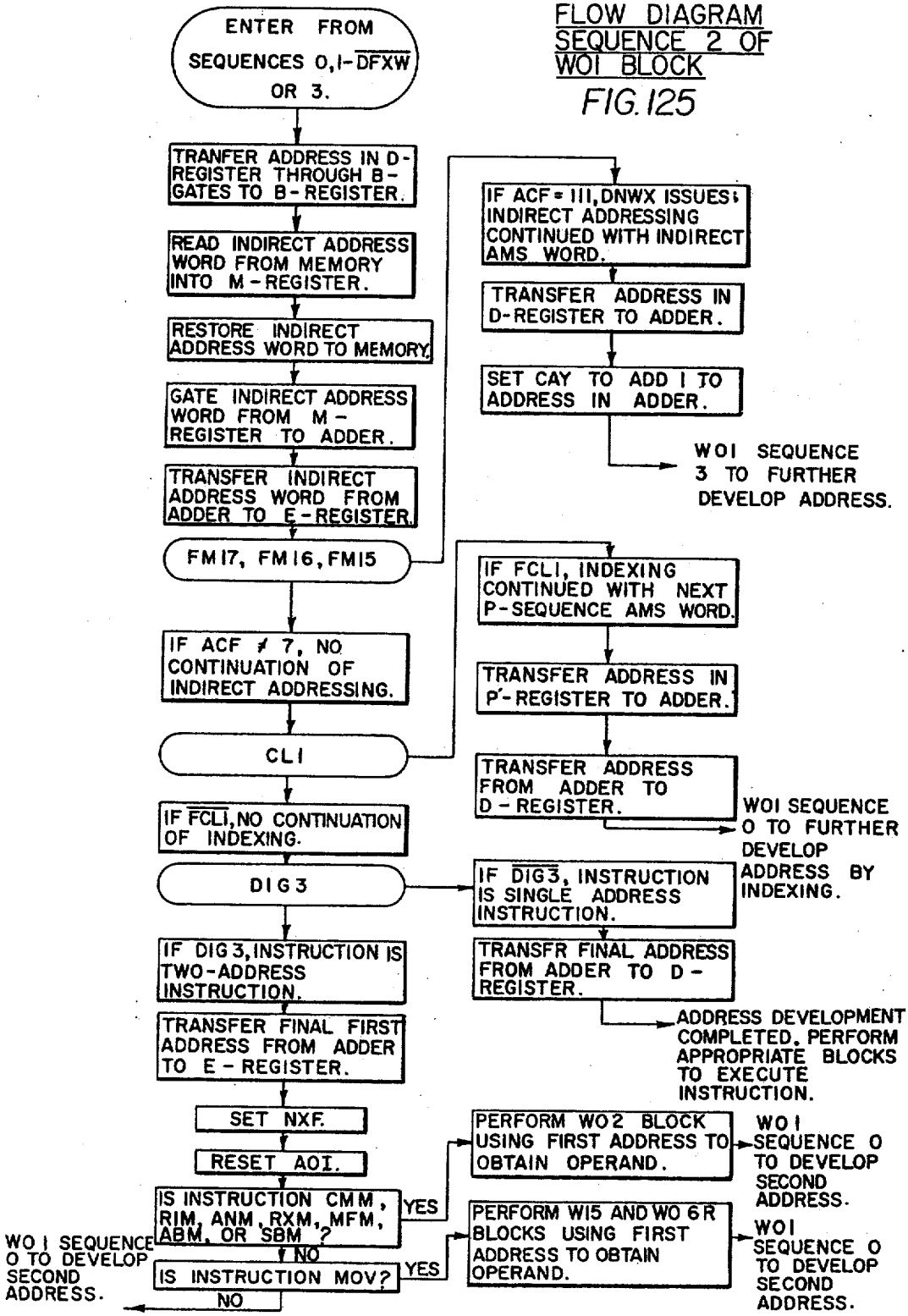

A
IF $\overline{FCL2}$, CHECK CLI.
IF FCL I, INDEXING CONTINUED WITH NEXT P - SEQUENCE AMS WORD.
IF $\overline{FCLI}$, NO CONTINUATION OF INDEXING.
TRANSFER DEVELOPED ADDRESS FROM ADDER TO E - REGISTER.
TRANSFER ADDRESS IN P - REGISTER TO ADDER.
DIG3
WOI SEQUENCE O TO FURTHER DEVELOP ADDRESS BY INDEXING.
IF $\overline{DIG3}$, INSTRUCTION IS NOT TWO - ADDRESS INSTRUCTION.
IF DIG3, INSTRUCTION IS TWO - ADDRESS INSTRUCTION.
TRANSFER FINAL ADDRESS FROM ADDER TO D - REGISTER.
TRANSFER FINAL FIRST ADDRESS FROM ADDER TO E - REGISTER.
ADDRESS DEVELOPMENT COMPLETED. PERFORM APPROPRIATE BLOCKS TO EXECUTE INSTRUCTION.
SET NXF.
RESET AOI.
IS INSTRUCTION CMM, RIM, ANM, RXM, MFM, ABM, OR SBM ?
YES
PERFORM WO2 BLOCK USING FIRST ADDRESS TO OBTAIN OPERAND.
WOI SEQUENCE O TO DEVELOP SECOND ADDRESS.
NO
IS INSTRUCTION MOV?
YES
PERFORM WI5 AND WO6R BLOCKS USING FIRST ADDRESS TO OBTAIN OPERAND.
WOI SEQUENCE O TO DEVELOP SECOND ADDRESS.
NO
WOI SEQUENCE O TO DEVELOP SECOND ADDRESS.

FIG. 124b

FLOW DIAGRAM SEQUENCE 2 OF WOI BLOCK

FIG. 125



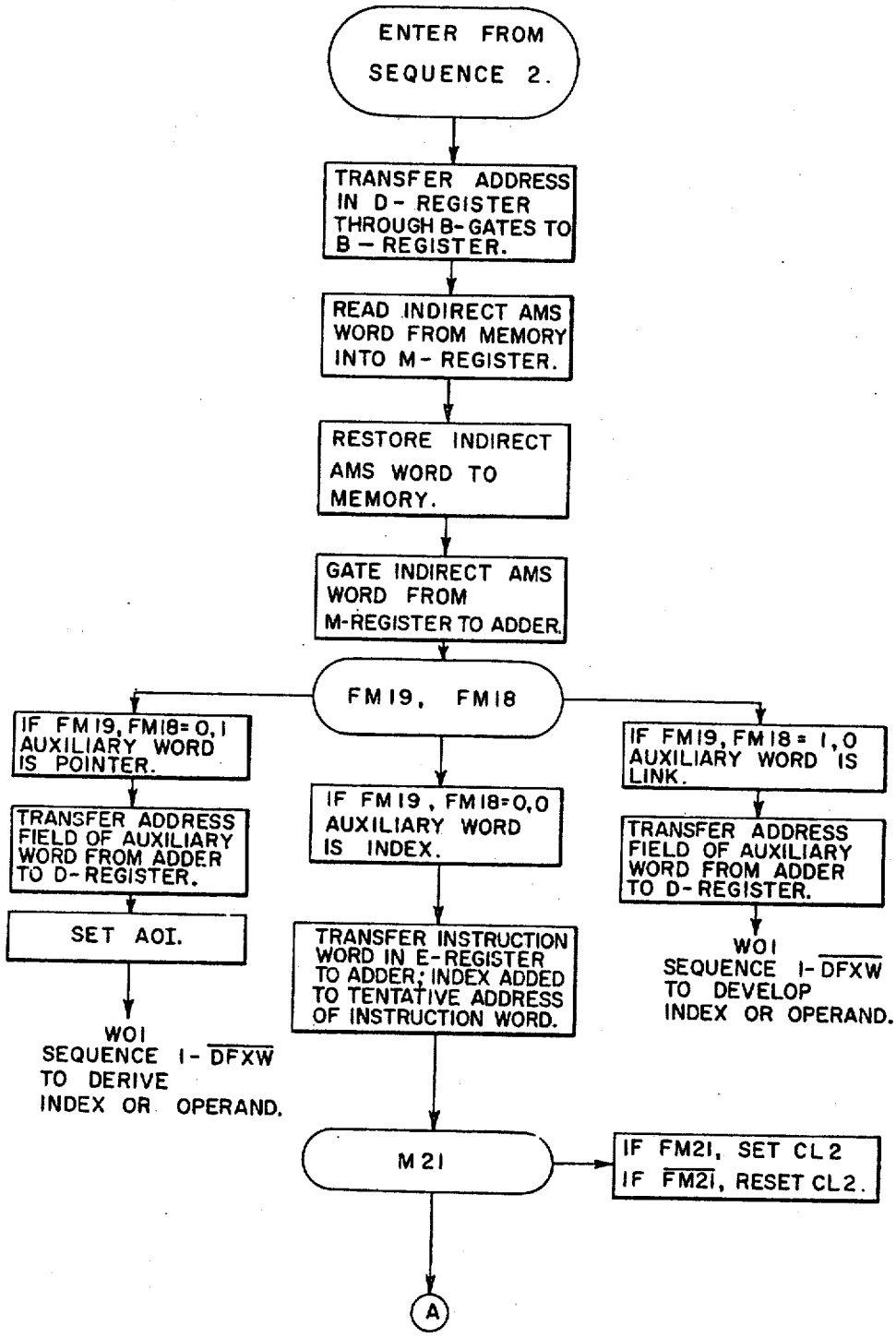


FIG. 126a
FLOW DIAGRAM
SEQUENCE 3 OF WOI BLOCK

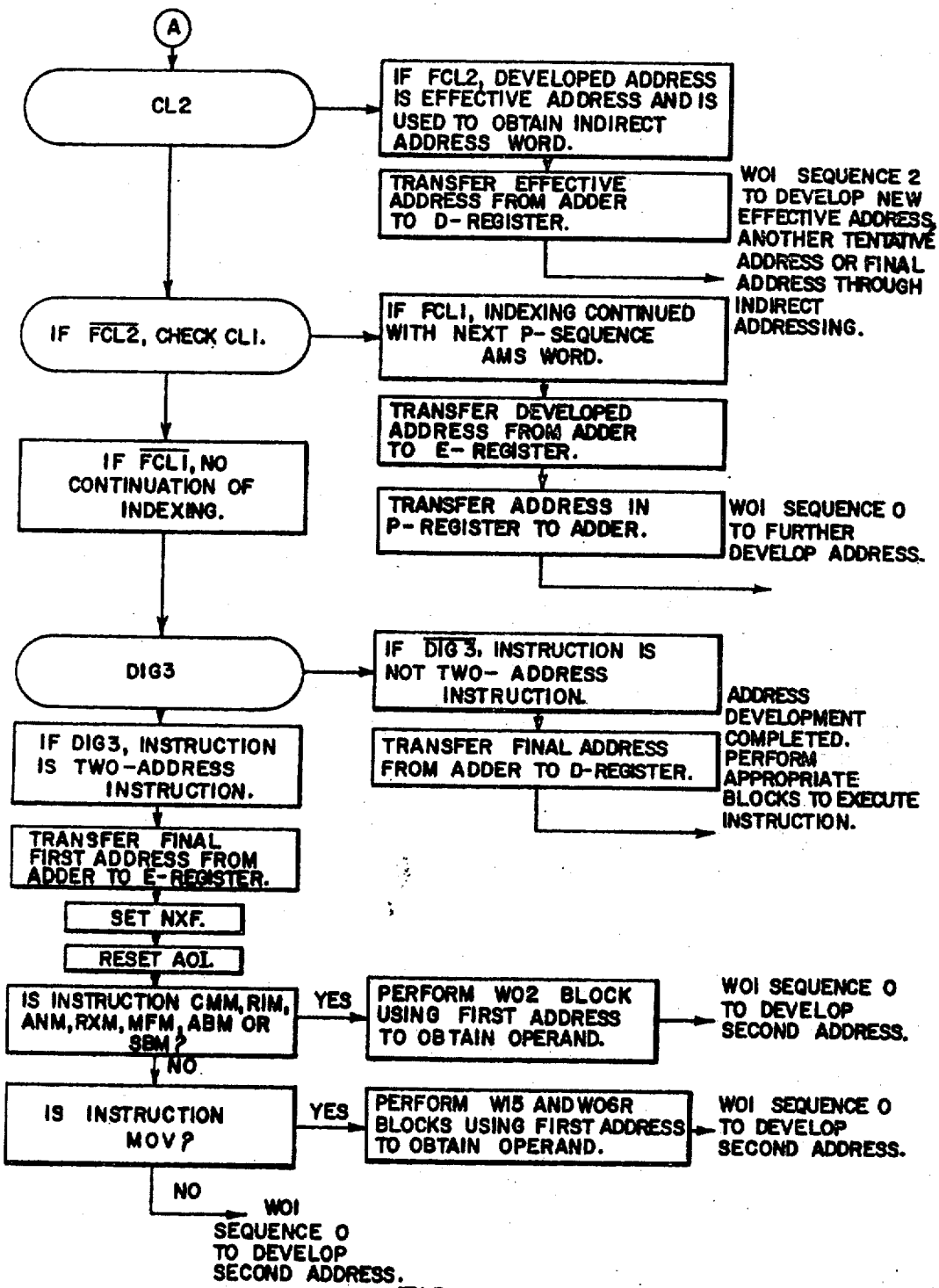

A
CL2
IF FCL2, DEVELOPED ADDRESS IS EFFECTIVE ADDRESS AND IS USED TO OBTAIN INDIRECT ADDRESS WORD.
TRANSFER EFFECTIVE ADDRESS FROM ADDER TO D-REGISTER.
WOI SEQUENCE 2 TO DEVELOP NEW EFFECTIVE ADDRESS, ANOTHER TENTATIVE ADDRESS OR FINAL ADDRESS THROUGH INDIRECT ADDRESSING.
IF $\overline{FCL2}$, CHECK CLI.
IF FCLI, INDEXING CONTINUED WITH NEXT P-SEQUENCE AMS WORD.
TRANSFER DEVELOPED ADDRESS FROM ADDER TO E-REGISTER.
IF $\overline{FCLI}$, NO CONTINUATION OF INDEXING.
TRANSFER ADDRESS IN P-REGISTER TO ADDER.
WOI SEQUENCE 0 TO FURTHER DEVELOP ADDRESS.
DIG3
IF $\overline{DIG3}$, INSTRUCTION IS NOT TWO-ADDRESS INSTRUCTION.
ADDRESS DEVELOPMENT COMPLETED. PERFORM APPROPRIATE BLOCKS TO EXECUTE INSTRUCTION.
IF DIG3, INSTRUCTION IS TWO-ADDRESS INSTRUCTION.
TRANSFER FINAL ADDRESS FROM ADDER TO D-REGISTER.
TRANSFER FINAL FIRST ADDRESS FROM ADDER TO E-REGISTER.
SET NXF.
RESET AOI.
IS INSTRUCTION CMM, RIM, ANM, RXM, MFM, ABM OR SBM?
YES
PERFORM WO2 BLOCK USING FIRST ADDRESS TO OBTAIN OPERAND.
WOI SEQUENCE 0 TO DEVELOP SECOND ADDRESS.
NO
IS INSTRUCTION MOV?
YES
PERFORM WI5 AND WO6R BLOCKS USING FIRST ADDRESS TO OBTAIN OPERAND.
WOI SEQUENCE 0 TO DEVELOP SECOND ADDRESS.
NO
WOI SEQUENCE 0 TO DEVELOP SECOND ADDRESS.

FIG. 126 b

FIG. 127a

TIMING DIAGRAM - INSTRUCTIONS 40 (LDS), 41 (LDD), 42 (LDT) AND 43 (LDQ)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119 except as indicated)	TL5			DILX	Transfer contents of flip-flops IR1, IR0 to flip-flops PL2, PL1 to set accumulator working length.
W01 If address development required. (Fig. 121)					
W02	TL0			DDBX	Operand address to Memory. DDBX is present during entire W02 block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1		DSX	DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	QTCK		ESX RCK		Stop Program Processor Clock Generator.
	TL2				Wait for DMDA to start Program Processor Clock Generator.
	QTRK	RCK			Start Program Processor Clock Generator.
	QTCK	MWR			Initiate write operation to restore operand to Memory.

FIG. 127b

TIMING DIAGRAM - INSTRUCTIONS 40 (LDS), 41 (LDD), 42 (LDT) AND 43 (LDQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL3				
	QTCK				
	TL4				
	QTCK				
	TL5				
	QTCK		MSX RCK		Stop Program Processor Clock Generator and wait for memory synchronizing signal.
	TL0				
	QTC0	WR0			Go to W03 block.
	QTRK	RCK			Start Program Processor Clock Generator on synchronizing signal DMBU from Memory.
W03	TL0			$\overline{\text{DAMR}}$	Inhibit transfer of word from Memory to M-Register. $\overline{\text{DAMR}}$ is present during entire W03 block.
				DABX	Transfer accumulator address from A-Register and Accumulator Counter Register to Memory.
	QTCK	MRD			Initiate read operation in Memory to clear addressed accumulator location in Memory.

FIG. 127c

TIMING DIAGRAM - INSTRUCTIONS 40 (LDS), 41 (LDD), 42 (LDT) AND 43 (LDQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL1			$\overline{DCM0}$, $\overline{DCM1}$, $\overline{DCM2}$, $\overline{DCM3}$	Inhibit clearing of M-Register. Operand is stored in M-Register.
	QTCK		ESX RCK		Stop Program Processor Clock Generator.
	TL2				Wait for DMDA to start Program Processor Clock Generator.
	QTRK	RCK			Start Program Processor Clock Generator.
	QTCK	MWR			Initiate write operation to store operand in addressed accumulator location in Memory.
	TL3				
	QTCK				
	TL4				
	QTCK				
	TL5			DDCD	Decrease address in D-Register by one to address next word of field.
	QTCK	DLAOΔPSX	RCK		If last accumulator operation (AC2, AC1 = PL2, PL1), transfer next instruction address from P-Register to Adder. Stop Program Processor Clock Generator and wait for memory synchronizing signal.

FIG. 127d

TIMING DIAGRAM - INSTRUCTIONS 40 (LDS), 41 (LDD), 42 (LDT) AND 43 (LDQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				
	QTC0		DLAOΔ(WR1, WR0)		If last accumulator operation, return to W00 block.
			$\overline{\text{DLAO}}\Delta$WR0		If not last accumulator operation, return to W02 block.
				DACU	Advance count in Accumulator Counter Register by one to address next word in Accumulator.
	QTRK	RCK			Start Program Processor Clock Generator on synchronizing signal DMBU from Memory.

FIG. 128a

TIMING DIAGRAM - INSTRUCTIONS 44(STS), 45(STD), 46(STT) AND 47(STQ)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - (Fig. 119)					
W01 - If address development required (Fig. 121)					
W02	TL0			DABX	Transfer accumulator address from A-Register and Accumulator Counter Register to Memory. DABX is present during entire W02 block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register
	QTCK		ESX RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Program Processor Clock Generator.
	QTCK	MWR			Initiate write operation to restore accumulator word to Memory.
	TL3				
	QTCK				
	TL4				
	QTCK				

FIG. 128b

TIMING DIAGRAM - INSTRUCTIONS 44(STS), 45(STD), 46(STT) AND 47(STQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	$\overline{QTCK}$		$\overline{MSX}$		
			RCK		Stop Program Processor Clock Generator and wait for memory synchronizing signal.
	TL0				
	$\overline{QTCO}$	$\overline{WRO}$			Go to W03 block.
	QTRK	RCK			Start Program Processor Clock Generator on synchronizing signal DMBU from Memory.
W03	TL0			$\overline{DAMR}$	Inhibit transfer of word from Memory to M-Register. $\overline{DAMR}$ is present during entire W03 block.
				DDBX	Transfer address from D-Register to Memory.
	$\overline{QTCK}$	$\overline{MRD}$			Initiate read operation in Memory to clear addressed memory location.
	TL1			$\overline{DCM0}$, $\overline{DCM1}$, $\overline{DCM2}$, $\overline{DCM3}$	Inhibit clearing of M-Register. Accumulator word to be stored in addressed memory location is in M-Register.
	$\overline{QTCK}$		$\overline{ESX}$		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for DMDA to start Program Processor Clock Generator.
	$\overline{QTRK}$	RCK			Start Program Processor Clock Generator.
	$\overline{QTCK}$	$\overline{MWR}$			Initiate write operation to store word in addressed memory location.

FIG. 128c

TIMING DIAGRAM - INSTRUCTIONS 44(STS), 45(STD), 46(STT) AND 47(STQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL3				
	QTCK				
	TL4				
	QTCK				
	TL5			DDCD	Decrease address in D-Register by one to address next word of field.
	QTCK	DLAO△PSX			If last accumulator operation (AC2, AC1 = PL2, PL1), transfer next instruction address from P-Register to Adder.
			RCK		Stop Program Processor Clock Generator and wait for memory synchronizing signal.
	TL0				
	QTC0		DLAO△(WR1, WR0)		If last accumulator operation, return to W00 block.
			$\overline{\text{DLAO}}$△WR0		If not last accumulator operation, return to W02 block.
				DACU	Advance count in Accumulator Counter Register by one to address next word in Accumulator.
	QTRK	RCK			Start Program Processor Clock Generator on synchronizing signal $\overline{\text{DMBU}}$ from Memory.

FIG. 129a

TIMING DIAGRAM - INSTRUCTION 30 (MFM)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					
W01 - If development of first address required. (Fig. 121)					
W02	TL0			DDBX	First operand address to Memory. DDBX is present during entire W02 block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1		DSX	DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	QTCK	MSX			Transfer operand from M-Register to Adder when operand stored in M-Register.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for DMDA to start Program Processor Clock Generator.
	QTRK	RCK			Start Program Processor Clock Generator.
	QTCK	MWR			Initiate write operation to restore operand to Memory.
	TL3				
	QTCK				
	TL4				
	QTCK			QSEX	Transfer operand through Adder to E-Register.

FIG. 129b

TIMING DIAGRAM - INSTRUCTION 30 (MFM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK	PSX			Address of next P-sequence word from P-Register to Adder. This is P-Sequence SAS Word used to develop second address.
			MSX		
			RCK		Stop Program Processor Clock Generator and wait for memory synchronizing signal.
	TL0				
	QTC0	WR1	WR0		Go to W01 block to develop second address.
	QTRK	RCK			Start Clock Generator on synchronizing signal $\overline{\text{DMBU}}$ from Memory.
W01 To develop second address (Fig. 121 except as indicated)	TL2				
	QTCK	$\overline{\text{DVCA}}\Delta\text{MWR}$			Inhibit restoration of second operand when address of second memory location obtained.
		$\text{DIMM}\cdot\overline{\text{DSPB}}\cdot\overline{\text{DBRC}}\cdot\overline{\text{DGCP}}\Delta\text{ESX}$			If second address obtained, transfer operand in E-Register to Adder.
	TL5				
	QTCK		DVCAΔMSX		Reset MSX when address of second memory location obtained.
	TL0				
	QTC0	WR3, WR2, WR1, TL2	TL0		Go to W15S block.

FIG. 129c

TIMING DIAGRAM - INSTRUCTION 30 (MFM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W15S	TL2			$\overline{\text{DFXW}}\Delta\text{DDBX}$	If ACF of instruction word ≠ 1-6, address of second location transferred from D-Register to Memory.
				DFXWΔDQBX	If ACF of instruction word = 1-6, fixed index location address transferred from Q-Register to Memory.
	QTCK	MWR			Initiate write operation in Memory to transfer operand to second memory location.
				QSMA, QSMB, QSMC	Transfer operand through Adder to M-Register.
	TL3				
	QTCK		ESX, MSX, PSX, CAY		
	TL4				
	QTCK				
	TL5				
	QTCK	PSX			Transfer address of next instruction to Adder
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		WR3, WR2, WR1, WR0		Go to W00 block.
	QTRK	RCK			Start Program Processor Clock Generator on synchronizing signal $\overline{\text{DMBU}}$ from Memory.

FIG. 130a

TIMING DIAGRAM - INSTRUCTION 31 (MFI)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					
W01 - If development of first address required. (Fig. 121)					
W01 - To develop second address (Fig. 121 except as indicated)	TL2				
	QTCK	$\overline{DVCA}\Delta MWR$			Inhibit restoration of operand when address of second location obtained.
		$DIMM \cdot \overline{DSPB} \cdot \overline{DBRC} \cdot \overline{DGCP} \Delta ESX$			If second address obtained, first address transferred from E-Register to Adder.
	TL5				
	QTCK		DVCAΔMSX		Reset MSX when address of second memory location obtained.
	TL0				
	QTC0	WR3, WR2, WR1, TL2	TL0		Go to W15S block.
W15S	TL2			$\overline{DFXW}\Delta DDBX$	If ACF of instruction word $\neq$ 1-6, address of second location transferred from D-Register to Memory.
				DFXWΔDQBX	If ACF of instruction word = 1-6, fixed index location address transferred from Q-Register to Memory.
	QTCK	MWR			Initiate write operation in Memory to transfer address to second location.
				QSMA, QSMB	Transfer address (bits 0-14) through Adder to M-Register.

FIG. 130b

TIMING DIAGRAM - INSTRUCTION 31 (MFI) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL3		IRE, DCE		
	QTCK		ESX, MSX, PSX, CAY		
	TL4				
	QTCK				
	TL5				
	QTCK	PSX			Transfer address of next instruction to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		WR3, WR2, WR1, WR0		Go to W00 block.
	QTRK	RCK			Start Program Processor Clock Generator on synchronizing signal DMBU from Memory.

FIG. 131a

TIMING DIAGRAM - INSTRUCTION 32 (MTA)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					
W01 - If development of first address required. (Fig. 121)					
W01 - To develop second address. (Fig. 121 except as indicated)	TL2				
	QTCK	$\overline{DVCA}\Delta MSX$			Transfer of operand word read from second location from M-Register to Adder inhibited, when second address obtained.
		MWR			Initiate write operation in Memory to restore contents of second memory location.
		$DIMM \cdot \overline{DSPB} \cdot \overline{DBRC} \cdot \overline{DGCP} \Delta ESX$			When second address obtained, transfer address of first memory location in E-Register to Adder.
	TL4				
	QTCK			DVCAΔQSDX	When second address obtained, transfer address of first memory location to D-Register.
	TL5				
	QTCK	DVCAΔMSX			When second address obtained, transfer operand word read from second location from M-Register to Adder.
			DVCAΔESX		When second address obtained, inhibit transfer of address of first memory location from E-Register to S-Register.
	TL0				
	QTC0	WR2			Go to W05R block.

FIG. 131b

TIMING DIAGRAM - INSTRUCTION 32 (MTA) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W05R	TL0			DDBX	Address of first memory location to Memory. DDBX is present during entire W05R block.
	QTCK	MRD			Initiate read operation in Memory to read contents of first memory location from Memory.
				QSEX	Transfer operand word read from second memory location through Adder to E-Register.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	QTCK	ESX			Transfer operand word from E-Register to Adder.
			MSX		Reset MSX.
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for DMDA to start Program Processor Clock Generator.
	QTRK	RCK			Start Program Processor Clock Generator.
	QTC2	WR3, WR1			Go to W15S block.
W15S	TL2			DDBX	Address of first memory location to Memory.
	QTCK	MWR			Initiate write operation in Memory.
				QSMA, QSMB	Transfer address field of operand word (bits 0-14) through Adder to M-Register.

FIG. 131c

TIMING DIAGRAM - INSTRUCTION 32 (MTA) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL3		IRE, DCE		
	QTCK		ESX, MSX, PSX, CAY		
	TL4				
	QTCK				
	TL5				
	QTCK	PSX			Transfer address of next instruction to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0	WR3, WR2, WR1, WR0			Go to W00 block.
	QTRK	RCK			Start Program Processor Clock Generator on synchronizing signal DMBU from Memory.

FIG. 132a
TIMING DIAGRAM - INSTRUCTION 06 (MOV)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					
W01 - If development of first address required. (Fig. 121)					
W15	TL0			DFBS	Enable B-Gate B03 to address fixed location 10 (octal) in Memory. DFBS is present during entire W15 block.
	QTCK				
	TL1			DAEX	Transfer address and working length of Accumulator to E-Register.
	QTCK	ESX			Transfer address and working length of Accumulator from E-Register to Adder.
			MSX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for signal DMDA from Memory to start Program Processor Clock Generator.
	QTRK				Start Clock Generator.
	QTCK	MWR			Initiate write operation in Memory to store address and working length of Accumulator in fixed location 10 (octal).
				QSMA, QSMB	Transfer address and working length of Accumulator from Adder to M-Register.

FIG. 132b
TIMING DIAGRAM - INSTRUCTION 06 (MOV) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL3	DSX			Transfer address of most-significant word of first memory field (to address) from D-Register to Adder.
			IRE, DCE		
	QTCK		ESX, PSX, CAY		
	TL4				
	QTCK				
	TL5				
	QTCK		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		WR3, WR0		Go to W06R block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W06R	TL0				
	QTCK			QSEX	Transfer address from Adder to E-Register.
	TL1		DSX		
	QTCK	PSX			Transfer address of P-Sequence SAS Word to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory to start Clock Generator.
	QTC2	WR0, TL0	WR2, WR1, TL2		Go to W01 block.
	QTRK	RCK			Start Clock Generator.

FIG. 132c

TIMING DIAGRAM - INSTRUCTION 06 (MOV) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W01 - To develop second address. (Fig. 121 except as indicated)	TL2				
	QTCK	DIMMΔESX			Apply first memory field address from E-Register to Adder.
	TL5				
	QTCK		DVCAΔMSX		Inhibit transfer of control word to Adder.
	TL0				
	QTC0	WR2, WR1, TL2	WR0, TL0		Go to W06S block.
W06S	TL2				
	QTCK	MSX			Transfer control word from M-Register to Adder.
			ESX		
				QSDX	Transfer first memory field address from Adder to D-Register.
	TL3			DCLA	Clear A-Register.
	QTCK				
	TL4				
	QTCK			QSEX	Transfer control word to E-Register.
	TL5				
	QTCK		MSX		
				QEAX	Transfer bits 2-14 of control word to A-Register.
				DEAP	Transfer bits 0, 1 of control word to Accumulator Length Indicator Register.
			RCK		Stop Program Processor Clock Generator.

FIG. 132d

TIMING DIAGRAM - INSTRUCTION 06 (MOV) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				Wait for memory synchronizing signal.
	QTC0		WR2		Go to W02 block.
	QTRK	RCK			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.
W02	TL0			DABX	Transfer address of word of second memory field from A-Register and Accumulator Length Indicator Register to Memory.
	QTCK	MRD			Initiate read operation in Memory.
	TL1	IRE			Set IRE to provide input DX15 to Adder, adding one to count field in E-Register.
				DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	QTCK	ESX			Transfer control word to Adder to increment count field.
		CAY			Add one to address field of control word.
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory to start Clock Generator.
	QTRK	RCK			Start Clock Generator.
	QTCK	MWR			Initiate write operation in Memory to restore word.
	TL3				
	QTCK				

FIG. 132e

TIMING DIAGRAM - INSTRUCTION 06 (MOV) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4			DCLA	Clear A-Register.
	QTCK			QSEX	Transfer updated control word with incremented count and address fields to E-Register.
		$\overline{MC23}$·QSEX ΔCRE			CRE set when count field of control word rolls over to zero.
	TL5				
	QTCK			QEAX, DEAP	Transfer address of next word of second memory field to A-Register and Accumulator Length Indicator Register.
			MSX		
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0	WR0			Go to W03 block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W03	TL0			DDBX	Transfer address of word of first memory field to Memory.
				$\overline{DAMR}$	Inhibit transfer of word in addressed memory location to M-Register. $\overline{DAMR}$ is present during entire W03 block.
	QTCK	MRD			Initiate read operation in Memory to clear addressed memory location.
	TL1				
	QTCK		ESX		
			RCK		Stop Program Processor Clock Generator.

FIG. 132f

TIMING DIAGRAM - INSTRUCTION 06 (MOV) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Clock Generator.
	QTCK	MWR			Initiate write operation in Memory to store word of second memory field in M-Register into addressed location of first memory field.
	TL3	DSX			Transfer address of first memory field to Adder. CAY, set during W02 block, increments address by one.
	QTCK				
	TL4				
	QTCK				
	TL5				
	QTCK		CAY RCK	QSDX	Transfer incremented address to D-Register. Stop Program Processor Clock Generator.
	TL0				Wait for synchronizing signal from Memory.
	QTC0		$\overline{FCRE}\Delta WR0$		If count field of control word ≠ 0, return to W02 and repeat W02 and W03 blocks.
	QTRK	FCREΔWR2	FCREΔ(WR1, WR0)		If count field = 0, go to W04 block. Start Clock Generator on signal DMBU from Memory.

FIG. 132g

TIMING DIAGRAM - INSTRUCTION 06 (MOV) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W04R	TL0			DFBS	Enable B-Gate B03 to address fixed location 10 (octal) in Memory. DFBS is present during entire W04R block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1		DSX IRE	DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	QTCK	MSX	ESX RCK		Transfer previously stored accumulator location and length to Adder. Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Clock Generator.
	QTC2	WR1			Go to W06S block.
W06S	TL2				
	QTCK	MSX	ESX		Transfer accumulator location and length to Adder.
	TL3			DCLA	Clear A-Register.
	QTCK				
	TL4				
	QTCK			QSEX	Transfer accumulator location and length to E-Register.

FIG. 132h

TIMING DIAGRAM - INSTRUCTION 06 (MOV) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK	FCRE△PSX			Transfer address of next instruction to Adder.
				QEAX, DEAP	Transfer accumulator location and length to A-Register and flip-flops PL2, PL1 respectively.
			MSX		
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		WR2, WR1		Go to W00 block.
	QTRK	RCK			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.

FIG. 133a
TIMING DIAGRAM - INSTRUCTION 20 (EXP)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					
W01 - If address development required. (Fig. 121)					
W02	TL0			DDBX	Transfer operand address to Memory. DDBX is present during entire W02 block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1			DCLQ	Clear Q-Register.
				DCM0, DCM1, DCM2, DCM3	Clear M-Register.
			DSX		
	QTCK	MSX			Transfer operand from M-Register to Adder when operand is stored in M-Register.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for DMDA to start Program Processor Clock Generator.
	QTRK	RCK			Start Clock Generator.
	QTCK	MWR			Initiate write operation in Memory.
	TL3				
	QTCK				
	TL4				
	QTCK			QSEX	Transfer operand to E-Register.

FIG. 133b

TIMING DIAGRAM - INSTRUCTION 20 (EXP) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK		MSX		
			RCK		Stop Program Processor Clock Generator and wait for memory synchronizing signal.
	TL0				
	QTCO	WRO			Go to W03 block.
	QTRK	RCK			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.
W03	TL0			DABX	Transfer accumulator word address to Memory.
				$\overline{\text{DAMR}}$	Inhibit transfer of addressed accumulator word to M-Register. $\overline{\text{DAMR}}$ is present during entire W03 block.
	QTCK	MRD			Initiate read operation in Memory to clear addressed accumulator word location.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	QTCK	ESX			Transfer operand to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RKC			Start Clock Generator.
	QTCK			QSMA	Transfer bits 0-5 from Adder to M-Register.
		MWR			Initiate write operation to store bits 0-5 in addressed Accumulator.

FIG. 133c
TIMING DIAGRAM - INSTRUCTION 20 (EXP) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL3				
	QTCK				
	TL4				
	QTCK				
	TL5				
	QTCK	DLAO△PSX			Address of next instruction to Adder if last accumulator operation.
			RCK		Stop Program Processor Clock Generator and wait for memory synchronizing signal.
	TL0				
	QTC0	$\overline{\text{DLAO}}$△(WR3, WR2, TL2)	$\overline{\text{DLAO}}$△(WR0, TL0)		If not last accumulator operation, go to W14S block.
			DLAO△(WR1, WR0)		If last accumulator operation, go to W00 block.
	QTRK	RKC			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.
W14S	TL2				
	QTCK	SRI			Set SRI flip-flop.
	TL3				
	QTCK	Q00			Count in Q-Register advanced to one.
				QSHE	Contents of E-Register shifted right one bit position.
	TL4				
	QTCK	Q01	Q00		Count in Q-Register advanced to two.
				QSHE	Contents of E-Register shifted right one bit position.

FIG. 133d

TIMING DIAGRAM - INSTRUCTION 20 (EXP) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK			FSRIΔQTLP	FSRI inhibits QTLP and advance of TL-Counter.
		Q00			Count in Q-Register advanced to three.
				QSHE	Contents of E-Register shifted right one bit position.
	QTCK	Q02	Q01, Q00		Count in Q-Register advanced to four.
				QSHE	Contents of E-Register shifted right one bit position.
	QTCK	Q00			Count in Q-Register advanced to five.
				QSHE	Contents of E-Register shifted right one bit position.
	QTCK		Q00, Q02		Count in Q-Register advanced to zero.
				QSHE	Contents of E-Register shifted right one bit position.
			SRI		Stop shift.
	QTCK		RCK		Stop Program Processor Clock Generator and wait for memory synchronizing signal.
	TL0				
	QTC0	WR0	WR3, WR2		Go to W03 block.
				DACU	Advance count in Accumulator Counter Register by one.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.

FIG. 134a

TIMING DIAGRAM - INSTRUCTION 21 (IMP)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - (Fig. 119)					
W01 - If address development required (Fig. 121)					
W02R	TL0			DDBX	Transfer operand address to Memory. DDBX is present during entire W02 block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
				DCLQ	Clear Q-Register.
			DSX		
	QTCK		ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for DMDA from Memory to start Clock Generator.
	QTRK	RCK			Start Clock Generator.
	QTC2	WR0, TL0	TL2		Go to W03 block.
W03	TL0			DABX	Transfer accumulator word address to Memory.
	QTCK	MRD			Initiate read operation in Memory.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	QTCK		ESX		
			RCK		Stop Program Processor Clock Generator.

FIG. 134b

TIMING DIAGRAM - INSTRUCTION 21 (IMP) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Clock Generator.
	QTCK	MWR			Initiate write operation in Memory.
	TL3				
	QTCK				
	TL4				
	QTCK				
	TL5				
	QTCK		CAY RCK		Stop Program Processor Clock Generator and wait for memory synchronizing signal.
	TL0				
	QTC0	WR3, WR2, TL2	WR0, TL0		Go to W14S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W14S	TL2				
	QTCK				
	TL3				
	QTCK				
	TL4				
	QTCK		ESX		

FIG. 134c

TIMING DIAGRAM - INSTRUCTION 21 (IMP) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK		RCK		Stop Program Processor Clock Generator and wait for memory synchronizing signal.
	TL0				
	QTC0			DCM0, DCM1, DCM2	Clear bit positions 6-23 of M-Register.
		TL2	WR3, WR2, TL0		Go to W02S block.
	QTRK	RCK			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.
W02S	TL2			DDBX	Transfer operand address to Memory.
	QTCK	MWR			Initiate write operation in Memory.
	TL3				
	QTCK				
	TL4				
	QTCK	$\overline{\text{DLA0}} \cdot \text{FSP2} \cdot \text{MJEO} \triangle \text{SP2}$	$\overline{\text{DLA0}} \cdot \text{FSP2} \cdot \text{MJEO} \triangle \text{SP2}$		Check parity before each character is written into Memory. If not last accumulator operation, state of SP2 changed each time character contains odd number of binary 1's. On last accumulator operation, generate parity based on last character and state of SP2 and store in 25th bit position of memory location.

FIG. 134d

TIMING DIAGRAM - INSTRUCTION 21 (IMP) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK	DLAO⊿PSX			Transfer address of next instruction to Adder, if last accumulator operation.
			MSX		
			RCK		Stop Program Processor Clock Generator and wait for memory synchronizing signal.
	TL0				
	QTC0		DLAO⊿WR1		Go to block W00 if last accumulator operation.
		$\overline{\text{DLAO⊿WR0}}$			Go to block W03 if not last accumulator operation.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
				DACU	Advance count in Accumulator Counter Register by one.
W03 - (Same as W03 block above)					Next accumulator word read into M-Register.
W14S	TL2			DNS0	Clear N-Register.
	QTCK	$\overline{\text{DAAC⊿SRI}}$			If not accumulator word A, set SRI.
	TL3	DBAC⊿(N00, N01) DCAC⊿N01 DDAC⊿N00			Set N-Register for correct number of character shifts.
	QTCK	Q00			Count in Q-Register advanced to one.
				QSHM	Circular shift contents of M-Register right one bit position.

FIG. 134e

TIMING DIAGRAM - INSTRUCTION 21 (IMP) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4				
	QTCK	Q01	Q00		Count in Q-Register advanced to two.
				QSHM	Circular shift contents of M-Register right one bit position.
	TL5				
	QTCK			$\overline{\text{FSRI}}$ QTLP	FSRI inhibits QTLP and advance of TL-Counter.
		Q00			Count in Q-Register advanced to three.
				QSHM	Circular shift contents of M-Register right one bit position.
	QTCK	Q02	Q01, Q00		Count in Q-Register advanced to four.
				QSHM	Circular shift contents of M-Register right one bit position.
	QTCK	Q00			Count in Q-Register advanced to five.
				QSHM	Circular shift contents of M-Register right one bit position.
	QTCK			DQE5⊿QNCD	Count in N-Register reduced by one each time that count in Q-Register = 5.

FIG. 134f

TIMING DIAGRAM - INSTRUCTION 21 (IMP) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	QTCK		Q00, Q02		Count in Q-Register returned to zero.
				QSHM	Circular shift contents of M-Register right one bit position.
		$\overline{FN00} \cdot \overline{FN01}$ ΔSRI			Continue circular right shifting until count in N-Register = 0. At the time character of accumulator word in M-Register will be in correct position for writing into appropriate character position of addressed memory location.
	QTCK		RCK		Stop Program Processor Clock Generator and wait for memory synchronizing signal.
	TL0				
	QTC0			DBACΔ(DCM0, DCM1, DCM3)	If accumulator word B, clear bit positions 0-5 and 12-23 of M-Register.
				DCACΔ(DCM0, DCM2, DCM3)	If accumulator word C, clear bit positions 0-11 and 18-23 of M-Register.
				DDACΔ(DCM1, DCM2, DCM3)	If accumulator word D, clear bit positions 0-17 of M-Register.
		TL2	WR3, WR2, TL0		Go to W02S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W02S - (Same as W02S block above).					

FIG. 135a
TIMING DIAGRAM - INSTRUCTION 22 (SHG)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00	TL0			DDBX	Transfer address of instruction word to Memory. DDBX is present during entire W00 block.
	QTCK	MRD			Initiate read operation in Memory.
			RNZ	QSDX	Transfer instruction address to D-Register.
	TL1			DCM0, DCM1, DCM2, DCM3, CLQ, CLC	Clear M-, Q- and CC-Registers.
			CL1, CL2, NXF, CRE, DSX		
	QTCK		ESX, COR, SP1, SP2, CAY, TB0, TB1, SIN, PSX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Program Processor Clock Generator.
	QTCK	MWR			Initiate write operation in Memory.
		MSX			Apply instruction word to Adder.
				QMIX	Transfer operation code to I-Register.
				(FM15+FM16+FM17) ΔDAMS	DAMS issues if ACF of instruction word = 1-7.
				FM15·FM16·FM17 ΔDNWX	DNWX issues if ACF of instruction word = 7.
	TL3			$\overline{\text{FMAN}}\Delta$DPCU	Count P-Register up.
				DMQX	Transfer ACF of instruction word to Q-Register.
	QTCK				

FIG. 135b
TIMING DIAGRAM - INSTRUCTION 22 (SHG) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4				
	$\overline{QTCK}$			QSEX	Transfer instruction word to E-Register.
				QSCX	Transfer bits 0-4 of instruction word to CC-Register.
				QSDX	Transfer bits 0-14 of instruction word to D-Register.
	TL5	DIG3·$\overline{DNWX}$ ΔNXF			Set NXF if two-address instruction and ACF ≠ 7.
	$\overline{QTCK}$	$\overline{DNWX}$ΔPSX			Transfer address of next P-sequence word to Adder if development of bits 0-14 of instruction word required.
			MSX		
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for synchronizing signal from Memory.
	$\overline{QTCO}$	DFXWΔXS1			Set XS1 if ACF = 1-6 (decoded from Q-Register).
			AC1, AC2, AOI		
			$\overline{DFXW}$Δ(XS1, XS2)		
		DAMSΔWR0			If development of bits 0-14 of instruction word required, go to W01 block.
		$\overline{DAMS}$Δ(WR3, WR2, WR1, WR0)			If development of bits 0-14 of instruction word not required, go to W15R block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W01 - If address field development required (Fig. 121)					

FIG. 135c
TIMING DIAGRAM - INSTRUCTION 22 (SHG) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W15R	TL0				
	$\overline{QTCK}$				
	TL1			FD11·$\overline{FD13}$ΔDDLX	If not binary shift and if bit position 11 is a binary 1, set working accumulator length to accumulator length to be shifted. (D10, D09 to PL2, PL1).
	$\overline{QTCK}$		MSX, ESX RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	$\overline{QTRK}$	RCK		DLFTΔDCAD	Start Clock Generator. If left shift, change count in Accumulator Counter Register to address highest-order accumulator word affected by shift, as specified in bit positions 9 and 10 of D-Register.
	$\overline{QTC2}$	$(\overline{DLFT}+\overline{DCCZ})$ ΔTL0	DLFT·DCCZ Δ(WR3, WR0) $(\overline{DLFT}+\overline{DCCZ})$ Δ(WR3, WR1, WR0, TL2)		If left shift and contents of CC-Register = 0, go to W06S block. If right shift or if count in CC-Register ≠ 0, go to W04R block.
W06S	TL2				
	$\overline{QTCK}$				
	TL3				
	$\overline{QTCK}$				
	TL4				
	$\overline{QTCK}$				
	TL5				
	$\overline{QTCK}$	PSX	RCK		Apply address of next instruction to Adder. Stop Program Processor Clock Generator.

FIG. 135d

TIMING DIAGRAM - INSTRUCTION 22 (SHG) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				Wait for synchronizing signal from Memory.
	$\overline{QTC0}$		WR2, WR1		Go to W00 block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W04R	TL0			FAOI(DLFT·DAAC +$\overline{DLFT}$·DLA0·$\overline{DNL4}$) ΔDFBS	Enable gate B03 to address memory location 10 (octal) if AOI set and if either left shift and accumulator word A being processed or right shift, last accumulator operation and count in N-Register > 3 if non-binary shift or > 23 if binary shift. DFBS is present during entire W04 block.
				DSIG·DSIM·FLSN ·DNL4ΔDRE4	Force adder output signals $\overline{DS55}$ and DS54 which represent sign to binary 0 if decimal shift, LSN set, count in N-Register > 3 and either right shift and last accumulator operation or left shift and accumulator word B being addressed or signle accumulator word operation. DRE4 is present during entire W04 block.
				($\overline{FAOI}$+DLFT·$\overline{DAAC}$ +$\overline{DLFT}$·DNL4+$\overline{DLFT}$ ·$\overline{DLA0}$)ΔDABX	Transfer address of accumulator word to Memory if AOI reset, if left shift and accumulator word A not being addressed, if right shift and count in N-Register < 4 if non-binary shift or < 24 if binary shift or if right shift and not last accumulator operation. DABX is present during entire W04 block.
	$\overline{QTCK}$	MRD			Initiate read operation in Memory.
				QCNX	Transfer count in CC-Register to N-Register.

FIG. 135e

TIMING DIAGRAM - INSTRUCTION 22 (SHG) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
			DSX		
	QTCK	MSX			Apply accumulator word in M-Register to Adder.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				
	QTC2	AOI			Set AOI to enable DFBS during subsequent W04S or W04R block.
				$\overline{\text{DLFT}}\Delta\text{DCAD}$	If right shift, reduce count in Accumulator Counter Register by one.
				$\text{DLFT}\cdot\overline{\text{DNL4}}\Delta\text{DCAU}$	If left shift, and if count in N-Register > 3 if non-binary shift and if > 23 if binary shift, advance count in Accumulator Counter Register by one.
		DNL4Δ(WR3, WR1, WR0)	DNL4ΔWR2		If count in N-Register < 4 if non-binary shift or < 24 if binary shift, go to W11S block. Otherwise continue in W04S block.
	QTRK	RCK			Start Clock Generator.
W04S	TL2			DAAC·DSIG·FSIN ·DALSΔ$\overline{\text{DS55}}$	$\overline{\text{DS55}}$ issues to insert minus sign in accumulator word A location if sign of accumulator before decimal shift was minus and if accumulator last store cycle.
				DAAC·DSIG·DALS ΔDS54	Force adder output signal $\overline{\text{DS54}}$ to binary 0 if sign being stored in accumulator word A location in Memory.

FIG. 135f

TIMING DIAGRAM - INSTRUCTION 22 (SHG) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	QTCK	MWR			Initiate write operation in Memory.
				QSMA, QSMB, QSMC	Transfer adder outputs to M-Register.
		DSIM·DSIG ·DNL4·FLSN ·DMRMΔSIN			Set SIN to store sign during decimal shift if sign of accumulator word A in M-Register is minus, if count in N-Register > 3, and if either left shift and accumulator word B being processed or right shift and last accumulator operation.
	TL3	$\overline{DDES}$·DAAC ·$\overline{DD08}$·FAOI ·$\overline{FM00}$ΔLES			Set LES to indicate "less than" condition if binary shift, bit position 8 (test bit) is binary 1, M00 is binary 0, accumulator word A being addressed and AOI set.
			$\overline{DDES}$·DAAC·DD08 ·FAOI·FM00ΔLES		Reset LES under above conditions if M00 is binary 1 to indicate "equal" condition.
			$\overline{DDES}$·DAAC·DD08 ·FAOIΔGRE		Reset GRE.
	QTCK		DNL4·$\overline{DLFT}$·$\overline{DNEZ}$ ΔSP2		Reset SP2 if right shift and count in N-Register = 1-3, if non-binary shift, or = 1-23 if binary shift.
			DSIG·DSIM·$\overline{DNL4}$ ΔLSN		Reset LSN to indicate sign has been stored in SIN during decimal shift.
			MSX, ESX, CAY		
	TL4				
	QTCK	DNL4·$\overline{DLFT}$ ·$\overline{DNEZ}$·MJE0 ΔSP2			Set SP2 if right shift, MJE0 present and count in N-Register = 1-3, if non-binary shift, or 1-23 if binary shift.

FIG. 135g
TIMING DIAGRAM - INSTRUCTION 22 (SHG) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK	DNEZ∆PSX			If count in N-Register = 0, apply address of next instruction to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for synchronizing signal from Memory.
	QTC0			$\overline{DLFT}$·DNL4∆DACU	Addvance count in Accumulator Counter Register if right shift and count in N-Register < 4, if non-binary shift, or < 24, if binary shift.
			DLFT∆DBYD		Count Accumulator Counter Register down by two if left shift.
			$\overline{DLFT}$·$\overline{DNL4}$∆DBYU		Count Accumulator Counter Register up by two if right shift.
			AOI		
			DNEZ∆WR2		If count in N-Register is zero, go to W00 block.
		DVVL∆WR1	DVVL∆WR2		If last accumulator, go to W02R block. If neither W00 nor W02R block, repeat W04 block.
	QTRK	RCK			Start Clock Generator on signal $\overline{DMBU}$ from Memory.

FIG. 135h

TIMING DIAGRAM - INSTRUCTION 22 (SHG) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W02R	TL0			DFBS	Enable gate B03 to address memory location 10 (octal).
	QTCK	MRD			Initiate read operation in Memory.
				QNCD	If count in N-Register < 4, if non-binary shift, or < 24, if binary shift, count N-Register to zero. If count in N-Register > 3, if non-binary shift, or > 23, if binary shift, count N-Register down by four.
			DLFTΔSP2		If left shift, reset SP2.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
			DSX		
	QTCK	FD12ΔMSX			If circular shift, apply contents of M-Register to Adder.
		DLFT·FSP1 ΔSP2			If left shift and FSP1, set SP2.
				QNCX	Transfer count in N-Register to CC-Register.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTC2			$\overline{\text{DLFT}}\Delta$DCAD	If right shift, reduce count in Accumulator Counter Register by one to address highest-order accumulator word affected by shift.
				DLFTΔDCAU	If left shift, advance count in Accumulator Counter Register by one to render count zero, addressing accumulator word A.

FIG. 135i

TIMING DIAGRAM - INSTRUCTION 22 (SHG) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	QTC2 (Cont.)	DNEZΔWR2	DNEZΔWR1		If count in N-Register = 0, go to W04S block.
		$\overline{\text{DNEZ}}\Delta$(WR3, WR2, WR0)			If count in N-Register $\neq$ 0, go to W15S block.
	$\overline{\text{QTRK}}$	$\overline{\text{RCK}}$			Start Clock Generator.
W15S	TL2			DABX	Transfer address of accumulator word to Memory.
	$\overline{\text{QTCK}}$	MWR			Initiate write operation in Memory.
				QSMA, QSMB, QSMC	Transfer adder output signals to M-Register.
	TL3				
	$\overline{\text{QTCK}}$		$\overline{\text{ESX}}$, $\overline{\text{MSX}}$, $\overline{\text{PSX}}$, CAY		
	TL4				
	$\overline{\text{QTCK}}$				
	TL5				
	$\overline{\text{QTCK}}$		$\overline{\text{RCK}}$		Stop Program Processor Clock Generator.
	TL0				Wait for synchronizing signal from Memory.
	QTC0			$\overline{\text{DLFT}}\Delta$DACU	If right shift, advance count in Accumulator Counter Register by one.
				DLFTΔDACD	If left shift, count Accumulator Counter Register down by one.
			WR3, WR1, WR0		Go to W04R block.
	QTRK	RCK			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.

FIG. 135j
TIMING DIAGRAM - INSTRUCTION 22 (SHG) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W11S	TL2			DSIG·DSIL·FLSNΔDRE4	Force adder output signals $\overline{DS55}$ and DS54 to zero if decimal shift, LSN set and either right shift and last accumulator operation or left shift and accumulator word A location being addressed. DRE4 is present during entire W11S block.
	QTCK	DMRM·DSIG ·DSIL·FLSN ΔSIN			If accumulator word A is minus, decimal shift, LSN set, and if either right shift and last accumulator operation or left shift and accumulator word A location being addressed, set SIN.
				QSEX	Transfer accumulator word to E-Register.
	TL3			DCLQ	Clear Q-Register.
				DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	QTCK	$\overline{DNEZ}$ΔSRI			If count in N-Register ≠ 0, to enable contents of M- and E-Registers to be shifted.
			DSIG·DSILΔLSN		Reset LSN if decimal shift, and either right shift and last accumulator operation or left shift and accumulator word A location being addressed.
			DLFTΔSP2		If left shift, reset SP2.
			MSX		

FIG. 135k
TIMING DIAGRAM - INSTRUCTION 22 (SHG) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4				
	QTCK			$\overline{FSRI}\Delta(QSHM, QSHE)$	Shift accumulator words in M- and E-Registers right one bit position.
		FE00ΔM23			Shift E-Register into M-Register.
		DLFT·FSP1 ΔSP2			If left shift and FSP1, set SP2.
		FSRIΔQ00			Count in Q-Register advanced to one.
	TL5				
	QTCK			$\overline{FSRI}\Delta QTLP$	FSRI inhibits QTLP and advance of TL-Counter.
		FSRIΔQ01	FSRIΔQ00		Count in Q-Register advanced to two.
				FSRIΔ(QSHM, QSHE)	Shift accumulator words in M- and E-Registers right one bit position.
		FE00ΔM23			Shift E-Register into M-Register.
	QTCK				QSHM and QSHE issue and the count in Q-Register advanced for each clock signal QTCK.
				$(\overline{FSRI}\cdot DQE5+FSRI\cdot DDES)\Delta QNCD$	The count in N-Register is reduced by one for each six shifts, if non-binary shift, or for each binary shift.
			(DNE1·DQE5·DDES +DNE1·DDES) ΔSRI		Reset SRI when counts in N- and Q-Registers are one and five respectively, if non-binary shift, or when count in N-Register is one, if binary shift. Delay in flip-flop SRI permits one more shift, reducing count in N-Register to zero and count in Q-Register to zero.
			$\overline{FSRI}\Delta RCK$		Stop Program Processor Clock Generator.

FIG. 135L

TIMING DIAGRAM - INSTRUCTION 22 (SHG) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				Wait for memory synchronizing signal.
	QTCO	TL2	WR3, TL0		Go to W03S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W03S	TL2			(DLFT+$\overline{\text{DLAO}}$)ΔDABX	Transfer accumulator word address to Memory if left shift or if not last accumulator operation.
				$\overline{\text{DLFT}}$·DLAOΔDFBS	If right shift and last accumulator operation, transfer address of memory location 10 (octal) to Memory.
	QTCK	MWR			Initiate write operation in Memory.
	TL3				
	QTCK	ESX			Apply accumulator word in E-Register to Adder.
			DLFTΔSP1		If left shift, reset SP1.
				QCNX	Transfer count in CC-Register to N-Register.
	TL4				
	QTCK	DLFT·MJE0 ΔSP1			Set SP1 if accumulator word stored during write operation contains odd number of binary 1's.
	TL5				
	QTCK		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTCO	WR2, TL2	WR1, WR0, TL0		Go to W04S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.

FIG. 136 a
FLOW DIAGRAM
INSTRUCTION 22 (SHG)

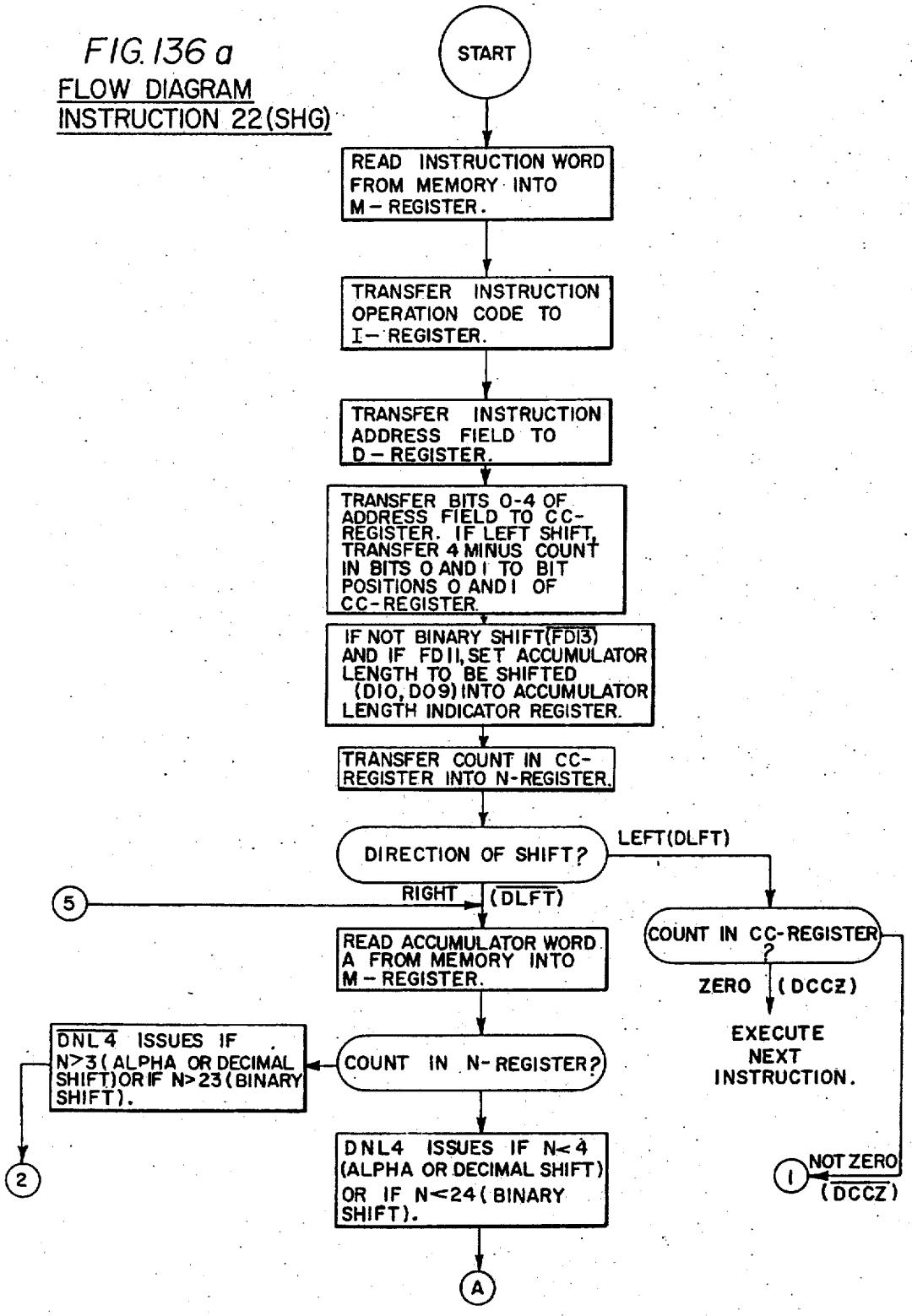


FIG. 136b

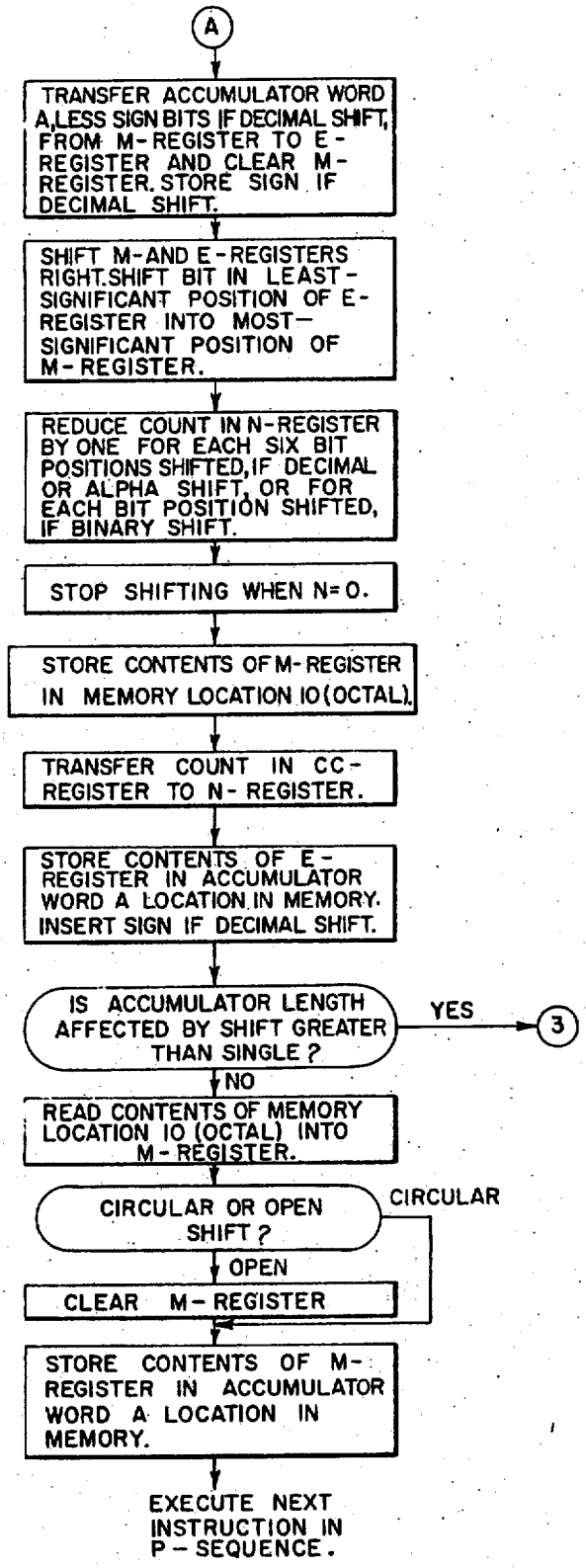
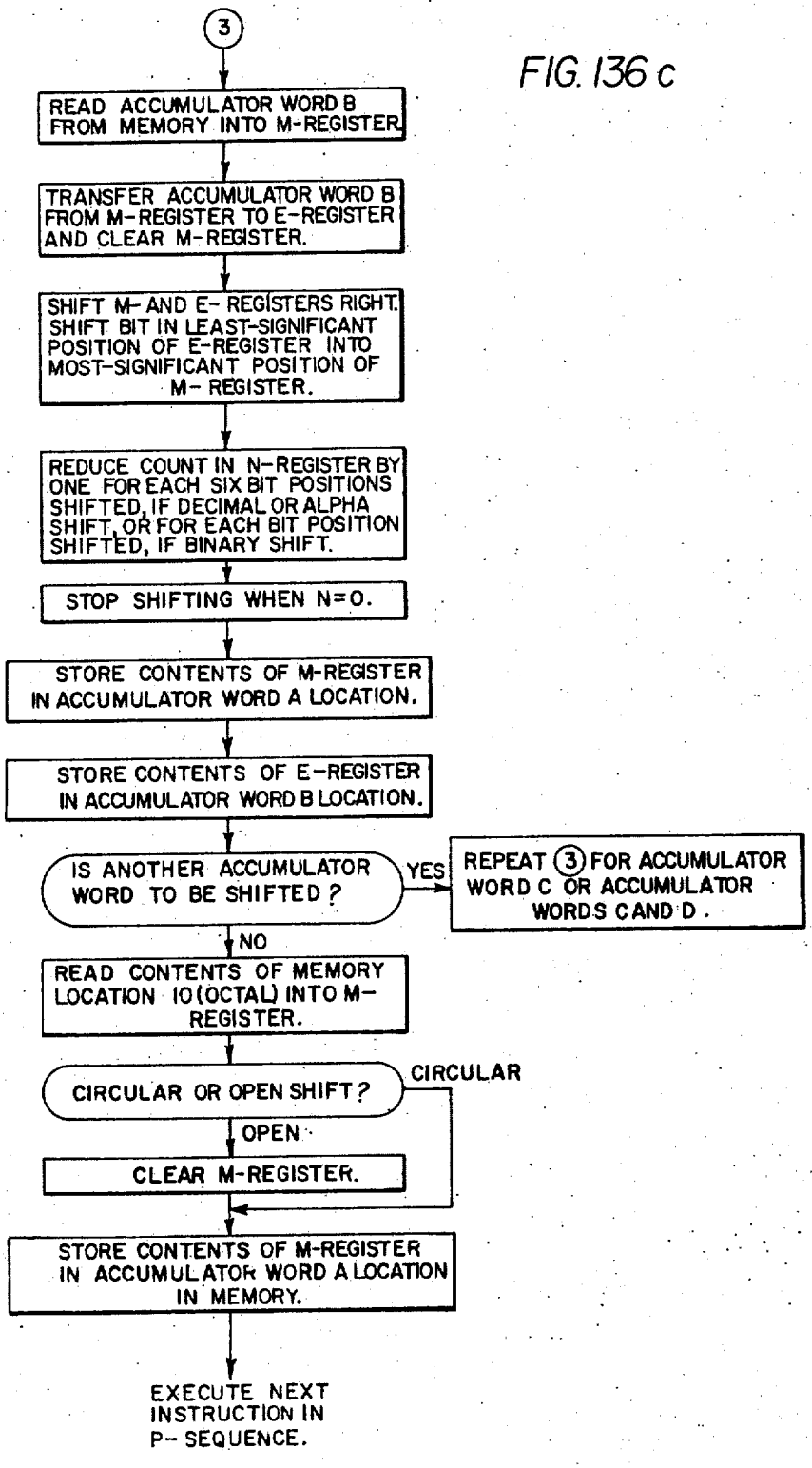

A
TRANSFER ACCUMULATOR WORD A, LESS SIGN BITS IF DECIMAL SHIFT, FROM M-REGISTER TO E-REGISTER AND CLEAR M-REGISTER. STORE SIGN IF DECIMAL SHIFT.
SHIFT M- AND E-REGISTERS RIGHT. SHIFT BIT IN LEAST-SIGNIFICANT POSITION OF E-REGISTER INTO MOST-SIGNIFICANT POSITION OF M-REGISTER.
REDUCE COUNT IN N-REGISTER BY ONE FOR EACH SIX BIT POSITIONS SHIFTED, IF DECIMAL OR ALPHA SHIFT, OR FOR EACH BIT POSITION SHIFTED, IF BINARY SHIFT.
STOP SHIFTING WHEN N=0.
STORE CONTENTS OF M-REGISTER IN MEMORY LOCATION 10 (OCTAL).
TRANSFER COUNT IN CC-REGISTER TO N-REGISTER.
STORE CONTENTS OF E-REGISTER IN ACCUMULATOR WORD A LOCATION IN MEMORY. INSERT SIGN IF DECIMAL SHIFT.
IS ACCUMULATOR LENGTH AFFECTED BY SHIFT GREATER THAN SINGLE?
YES
3
NO
READ CONTENTS OF MEMORY LOCATION 10 (OCTAL) INTO M-REGISTER.
CIRCULAR OR OPEN SHIFT?
CIRCULAR
OPEN
CLEAR M-REGISTER
STORE CONTENTS OF M-REGISTER IN ACCUMULATOR WORD A LOCATION IN MEMORY.
EXECUTE NEXT INSTRUCTION IN P-SEQUENCE.

FIG. 136 c


3
READ ACCUMULATOR WORD B FROM MEMORY INTO M-REGISTER.
TRANSFER ACCUMULATOR WORD B FROM M-REGISTER TO E-REGISTER AND CLEAR M-REGISTER.
SHIFT M- AND E- REGISTERS RIGHT. SHIFT BIT IN LEAST-SIGNIFICANT POSITION OF E-REGISTER INTO MOST-SIGNIFICANT POSITION OF M- REGISTER.
REDUCE COUNT IN N-REGISTER BY ONE FOR EACH SIX BIT POSITIONS SHIFTED, IF DECIMAL OR ALPHA SHIFT, OR FOR EACH BIT POSITION SHIFTED, IF BINARY SHIFT.
STOP SHIFTING WHEN N=0.
STORE CONTENTS OF M-REGISTER IN ACCUMULATOR WORD A LOCATION.
STORE CONTENTS OF E-REGISTER IN ACCUMULATOR WORD B LOCATION.
IS ANOTHER ACCUMULATOR WORD TO BE SHIFTED ?
YES
REPEAT (3) FOR ACCUMULATOR WORD C OR ACCUMULATOR WORDS C AND D.
NO
READ CONTENTS OF MEMORY LOCATION 10 (OCTAL) INTO M-REGISTER.
CIRCULAR OR OPEN SHIFT?
CIRCULAR
OPEN
CLEAR M-REGISTER.
STORE CONTENTS OF M-REGISTER IN ACCUMULATOR WORD A LOCATION IN MEMORY.
EXECUTE NEXT INSTRUCTION IN P- SEQUENCE.

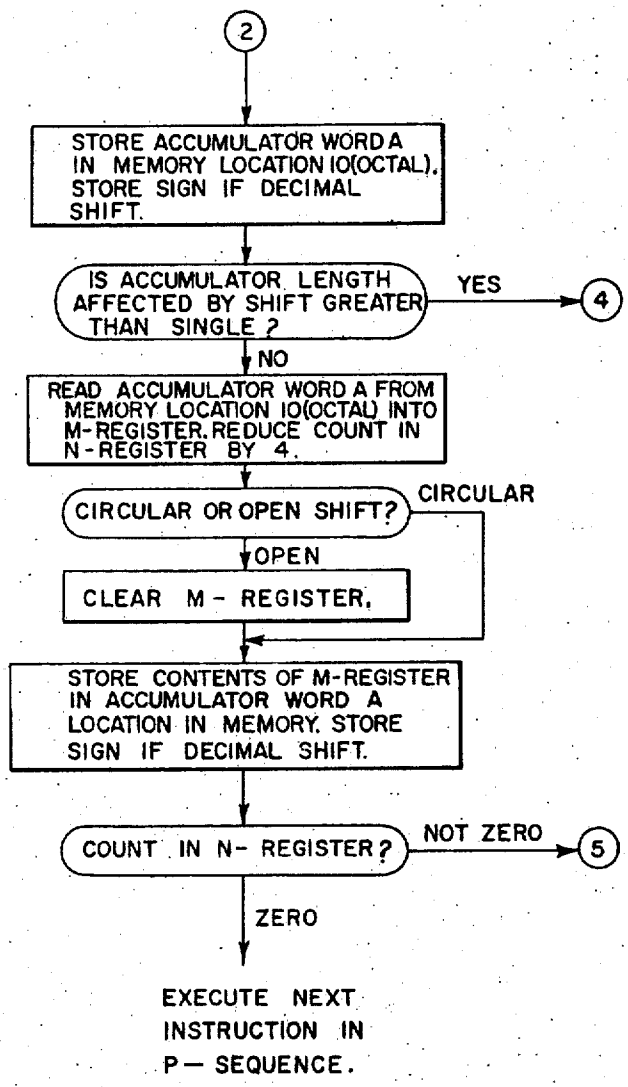

2
STORE ACCUMULATOR WORD A IN MEMORY LOCATION 10(OCTAL). STORE SIGN IF DECIMAL SHIFT.
IS ACCUMULATOR LENGTH AFFECTED BY SHIFT GREATER THAN SINGLE ?
YES
4
NO
READ ACCUMULATOR WORD A FROM MEMORY LOCATION 10(OCTAL) INTO M-REGISTER. REDUCE COUNT IN N-REGISTER BY 4.
CIRCULAR OR OPEN SHIFT?
CIRCULAR
OPEN
CLEAR M - REGISTER.
STORE CONTENTS OF M-REGISTER IN ACCUMULATOR WORD A LOCATION IN MEMORY. STORE SIGN IF DECIMAL SHIFT.
COUNT IN N- REGISTER?
NOT ZERO
5
ZERO
EXECUTE NEXT INSTRUCTION IN P— SEQUENCE.

FIG. 136d

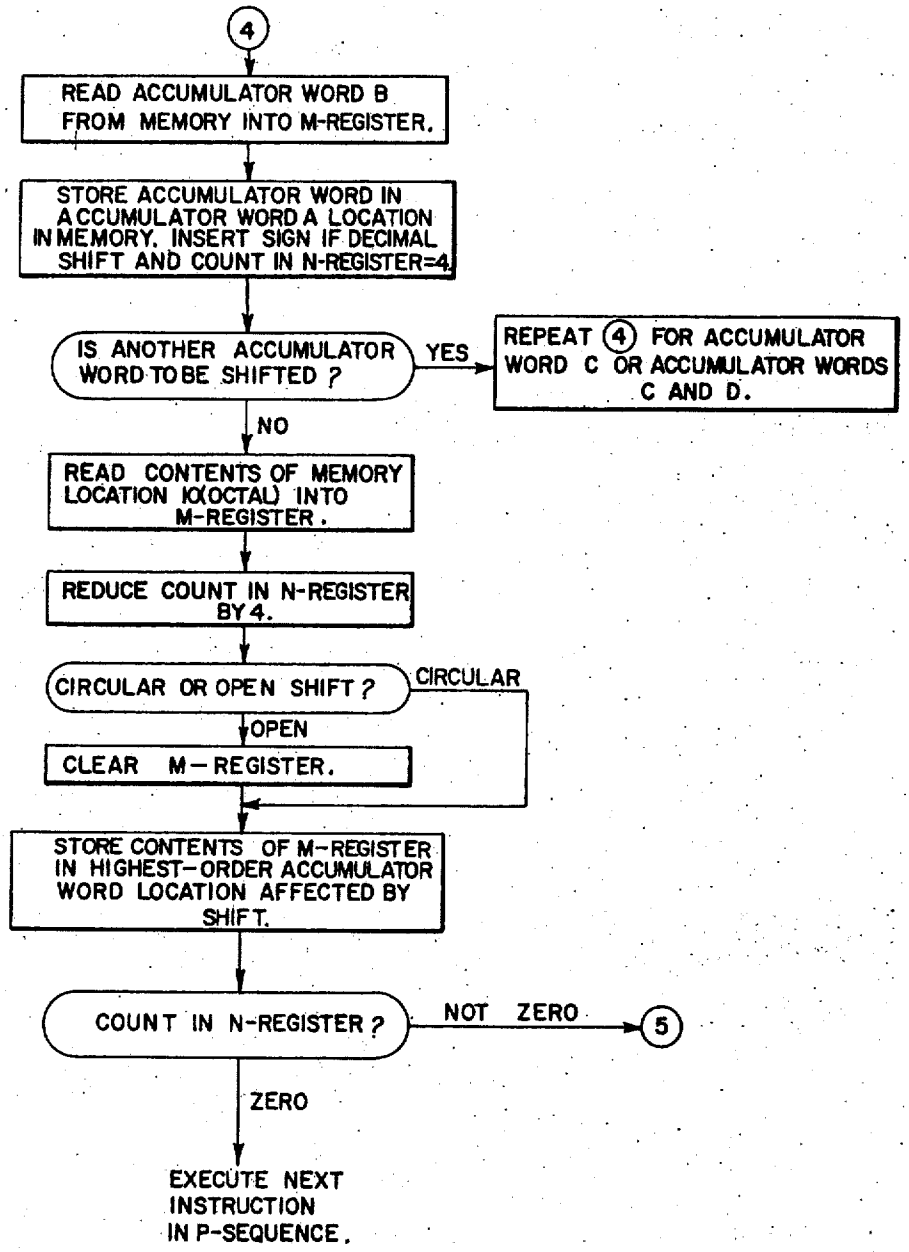
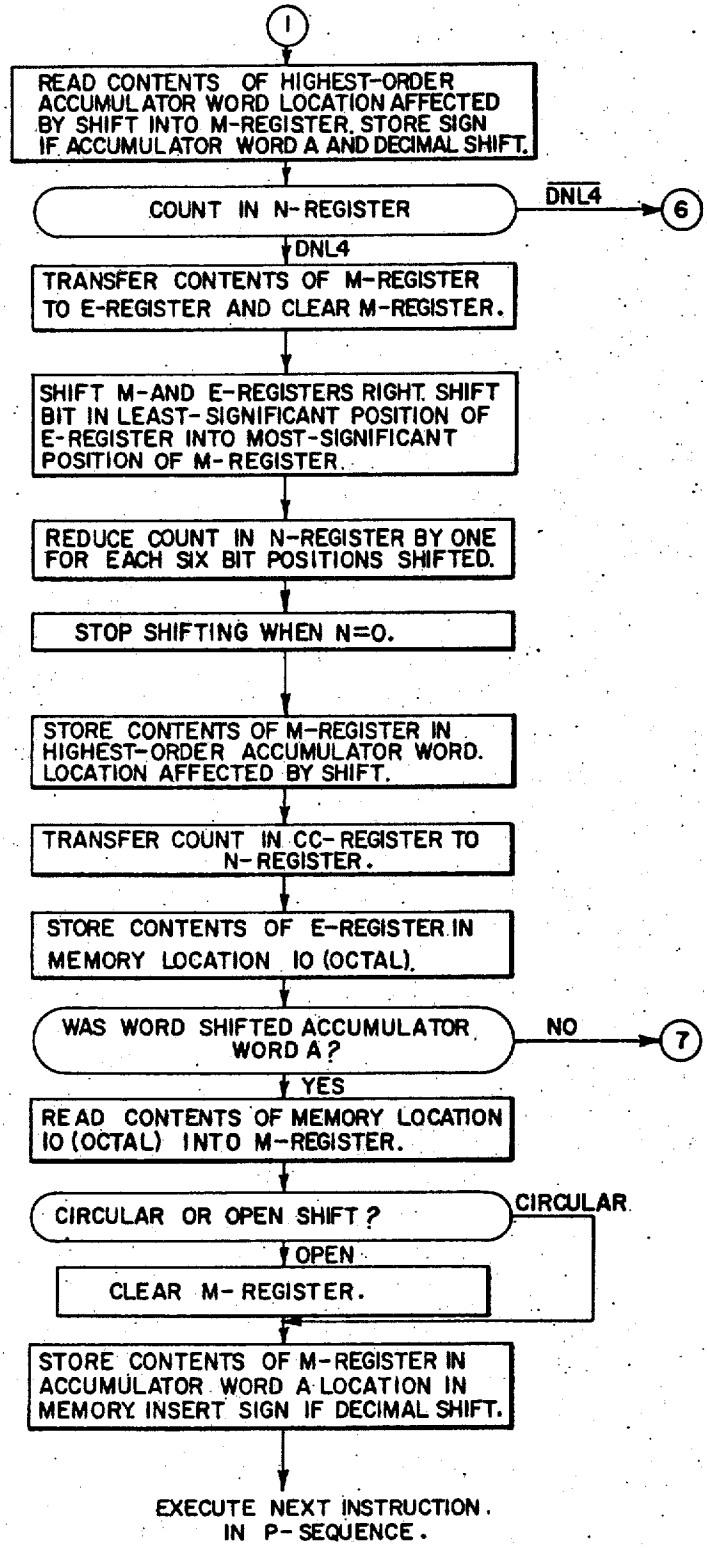

4
READ ACCUMULATOR WORD B FROM MEMORY INTO M-REGISTER.
STORE ACCUMULATOR WORD IN ACCUMULATOR WORD A LOCATION IN MEMORY. INSERT SIGN IF DECIMAL SHIFT AND COUNT IN N-REGISTER=4.
IS ANOTHER ACCUMULATOR WORD TO BE SHIFTED ?
YES
REPEAT (4) FOR ACCUMULATOR WORD C OR ACCUMULATOR WORDS C AND D.
NO
READ CONTENTS OF MEMORY LOCATION K(OCTAL) INTO M-REGISTER.
REDUCE COUNT IN N-REGISTER BY 4.
CIRCULAR OR OPEN SHIFT ?
CIRCULAR
OPEN
CLEAR M-REGISTER.
STORE CONTENTS OF M-REGISTER IN HIGHEST-ORDER ACCUMULATOR WORD LOCATION AFFECTED BY SHIFT.
COUNT IN N-REGISTER ?
NOT ZERO
5
ZERO
EXECUTE NEXT INSTRUCTION IN P-SEQUENCE.

FIG. 136e

FIG. 136 f


1
READ CONTENTS OF HIGHEST-ORDER ACCUMULATOR WORD LOCATION AFFECTED BY SHIFT INTO M-REGISTER. STORE SIGN IF ACCUMULATOR WORD A AND DECIMAL SHIFT.
COUNT IN N-REGISTER
DNL4
6
DNL4
TRANSFER CONTENTS OF M-REGISTER TO E-REGISTER AND CLEAR M-REGISTER.
SHIFT M-AND E-REGISTERS RIGHT. SHIFT BIT IN LEAST-SIGNIFICANT POSITION OF E-REGISTER INTO MOST-SIGNIFICANT POSITION OF M-REGISTER.
REDUCE COUNT IN N-REGISTER BY ONE FOR EACH SIX BIT POSITIONS SHIFTED.
STOP SHIFTING WHEN N=0.
STORE CONTENTS OF M-REGISTER IN HIGHEST-ORDER ACCUMULATOR WORD. LOCATION AFFECTED BY SHIFT.
TRANSFER COUNT IN CC-REGISTER TO N-REGISTER.
STORE CONTENTS OF E-REGISTER IN MEMORY LOCATION 10 (OCTAL).
WAS WORD SHIFTED ACCUMULATOR WORD A?
NO
7
YES
READ CONTENTS OF MEMORY LOCATION 10 (OCTAL) INTO M-REGISTER.
CIRCULAR OR OPEN SHIFT?
CIRCULAR
OPEN
CLEAR M-REGISTER.
STORE CONTENTS OF M-REGISTER IN ACCUMULATOR WORD A LOCATION IN MEMORY. INSERT SIGN IF DECIMAL SHIFT.
EXECUTE NEXT INSTRUCTION IN P-SEQUENCE.

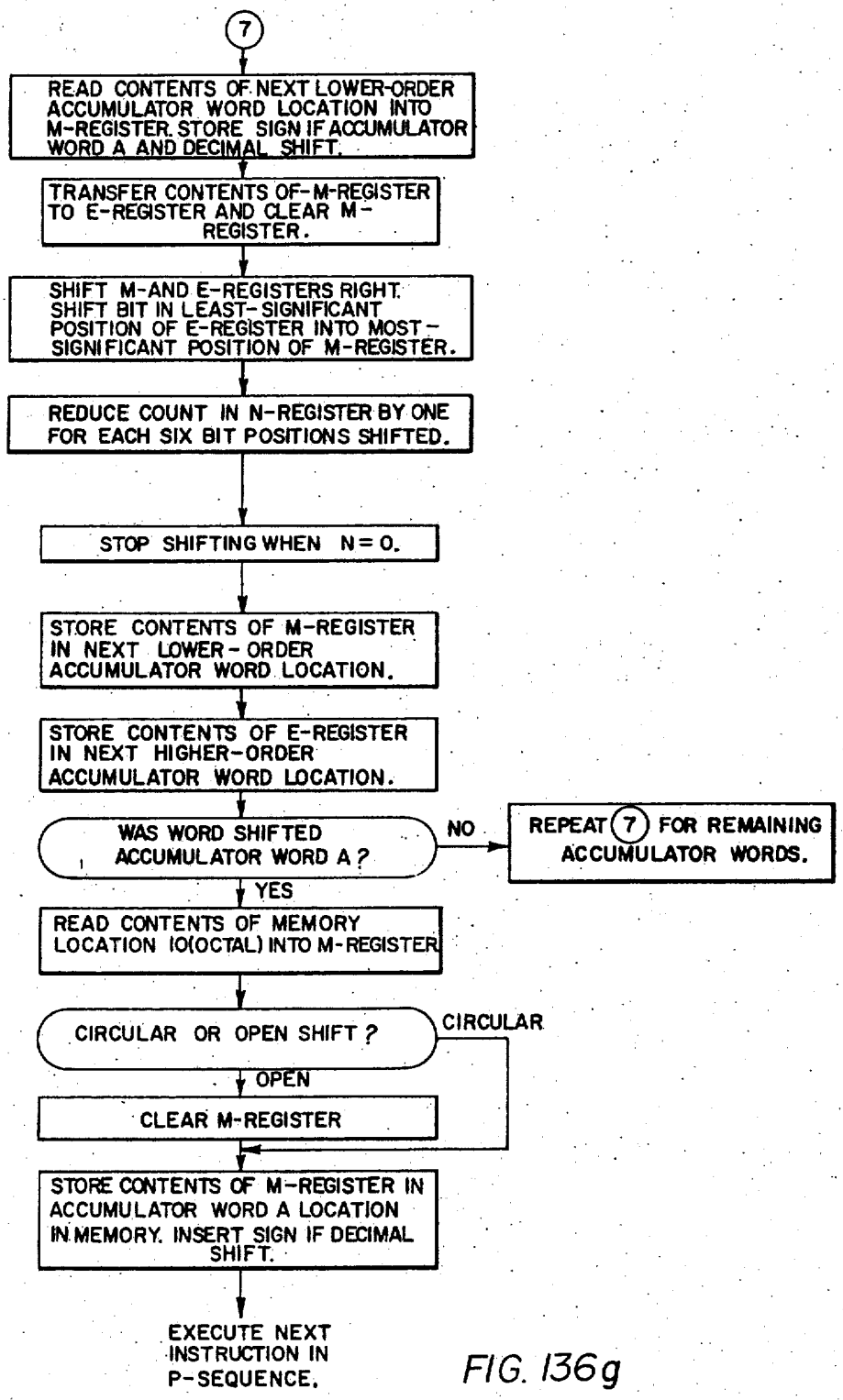

7
READ CONTENTS OF NEXT LOWER-ORDER ACCUMULATOR WORD LOCATION INTO M-REGISTER. STORE SIGN IF ACCUMULATOR WORD A AND DECIMAL SHIFT.
TRANSFER CONTENTS OF M-REGISTER TO E-REGISTER AND CLEAR M-REGISTER.
SHIFT M-AND E-REGISTERS RIGHT. SHIFT BIT IN LEAST-SIGNIFICANT POSITION OF E-REGISTER INTO MOST-SIGNIFICANT POSITION OF M-REGISTER.
REDUCE COUNT IN N-REGISTER BY ONE FOR EACH SIX BIT POSITIONS SHIFTED.
STOP SHIFTING WHEN N=0.
STORE CONTENTS OF M-REGISTER IN NEXT LOWER-ORDER ACCUMULATOR WORD LOCATION.
STORE CONTENTS OF E-REGISTER IN NEXT HIGHER-ORDER ACCUMULATOR WORD LOCATION.
WAS WORD SHIFTED ACCUMULATOR WORD A?
NO
REPEAT 7 FOR REMAINING ACCUMULATOR WORDS.
YES
READ CONTENTS OF MEMORY LOCATION 10(OCTAL) INTO M-REGISTER
CIRCULAR OR OPEN SHIFT?
CIRCULAR
OPEN
CLEAR M-REGISTER
STORE CONTENTS OF M-REGISTER IN ACCUMULATOR WORD A LOCATION IN MEMORY. INSERT SIGN IF DECIMAL SHIFT.
EXECUTE NEXT INSTRUCTION IN P-SEQUENCE.

FIG. 136g

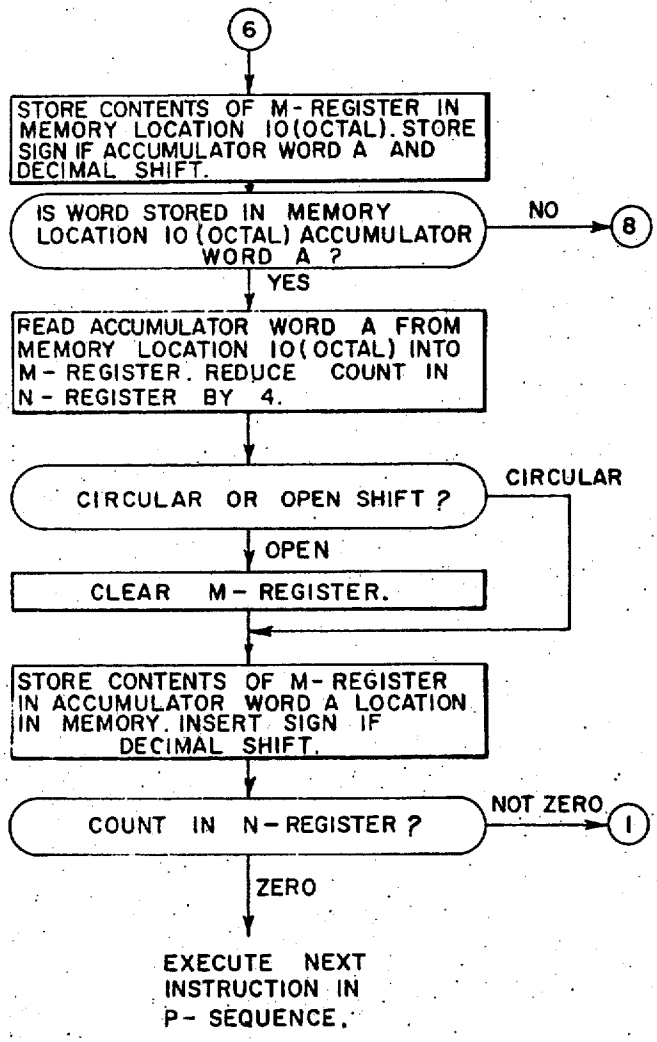

6
STORE CONTENTS OF M-REGISTER IN MEMORY LOCATION 10(OCTAL). STORE SIGN IF ACCUMULATOR WORD A AND DECIMAL SHIFT.
IS WORD STORED IN MEMORY LOCATION 10 (OCTAL) ACCUMULATOR WORD A ?
NO
8
YES
READ ACCUMULATOR WORD A FROM MEMORY LOCATION 10(OCTAL) INTO M-REGISTER. REDUCE COUNT IN N-REGISTER BY 4.
CIRCULAR OR OPEN SHIFT ?
CIRCULAR
OPEN
CLEAR M-REGISTER.
STORE CONTENTS OF M-REGISTER IN ACCUMULATOR WORD A LOCATION IN MEMORY. INSERT SIGN IF DECIMAL SHIFT.
COUNT IN N-REGISTER ?
NOT ZERO
1
ZERO
EXECUTE NEXT INSTRUCTION IN P-SEQUENCE.

FIG136h

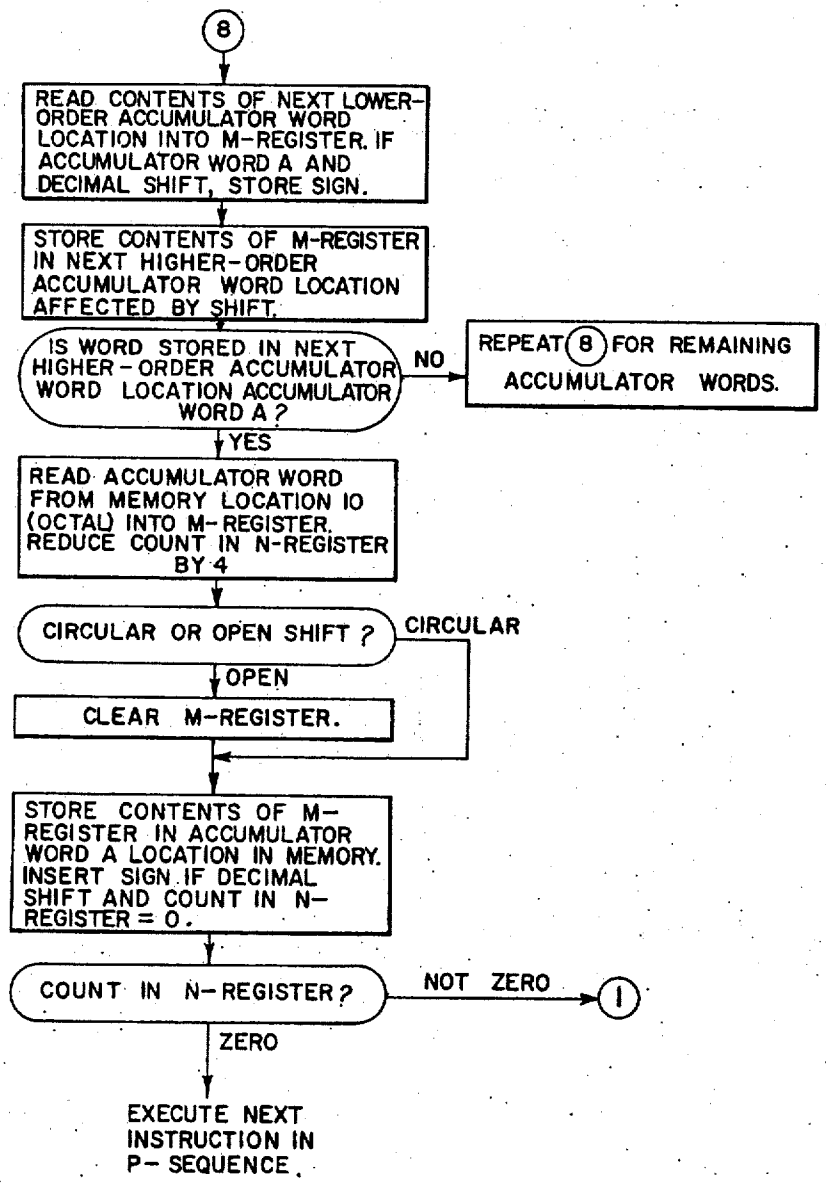

8
READ CONTENTS OF NEXT LOWER-ORDER ACCUMULATOR WORD LOCATION INTO M-REGISTER. IF ACCUMULATOR WORD A AND DECIMAL SHIFT, STORE SIGN.
STORE CONTENTS OF M-REGISTER IN NEXT HIGHER-ORDER ACCUMULATOR WORD LOCATION AFFECTED BY SHIFT.
IS WORD STORED IN NEXT HIGHER-ORDER ACCUMULATOR WORD LOCATION ACCUMULATOR WORD A ?
NO
REPEAT (8) FOR REMAINING ACCUMULATOR WORDS.
YES
READ ACCUMULATOR WORD FROM MEMORY LOCATION 10 (OCTAL) INTO M-REGISTER. REDUCE COUNT IN N-REGISTER BY 4
CIRCULAR OR OPEN SHIFT ?
CIRCULAR
OPEN
CLEAR M-REGISTER.
STORE CONTENTS OF M-REGISTER IN ACCUMULATOR WORD A LOCATION IN MEMORY. INSERT SIGN IF DECIMAL SHIFT AND COUNT IN N-REGISTER = 0.
COUNT IN N-REGISTER ?
NOT ZERO
1
ZERO
EXECUTE NEXT INSTRUCTION IN P-SEQUENCE.

FIG.136i

FIG. 137a

TIMING DIAGRAM - INSTRUCTION 50 (ADS)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					
W01 - If address development required (Fig. 121)					
W04R	TL0			DABX	Transfer address of accumulator word A to Memory. DABX is present during entire W04 block.
	$\overline{QTCK}$	MRD			Initiate read operation in Memory.
	TL1		DSX	DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	$\overline{QTCK}$	MSX			Transfer accumulator word A to Adder when it is stored in M-Register.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for DMDA from Memory to start Clock Generator.
	$\overline{QTCO}$	WR0, TL0	TL2		Go to W05 block.
	QTRK	RCK			Start Clock Generator.
W05	TL0			DDBX	Transfer operand address to Memory. DDBX is present during entire W05 block.
	$\overline{QTCO}$	MRD			Initiate read operation in Memory.
				QSEX	Transfer accumulator word A to E-Register.

FIG. 137b
TIMING DIAGRAM - INSTRUCTION 50 (ADS) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL1	DCE			DCE set to indicate decimal operation.
				$\overline{DAAE}\Delta$(DCM0, DCM1, DCM2, DCM3)	Clear M-Register, if operand and accumulator addresses are not the same.
	$\overline{QTCK}$	$\overline{ESX}$			Apply accumulator word A to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	$\overline{QTRK}$	$\overline{RCK}$			Start Clock Generator.
	QTCK	$\overline{DAAE}\Delta MWR$			Initiate write operation in Memory, if operand and accumulator addresses are not the same.
		DMRMΔSIN			Set SIN if sign of operand in M-Register is minus.
				$(\overline{DMRM}\cdot DERM+DMRM\cdot\overline{DERM})\Delta QCER$	Complement E-Register if unlike-signs addition to form complement of accumulator word A.
			$(\overline{DMRM}\cdot DERM+DMRM\cdot\overline{DERM})\cdot QCER\Delta LSN$		Reset LSN if unlike-signs addition.
		$(\overline{DMRM}\cdot DERM+DMRM\cdot\overline{DERM})\cdot QCER\Delta CAY$			Add one to complement of accumulator word A to form 10's complement.
	TL3		CRE		
	$\overline{QTCK}$				
	TL4				
	$\overline{QTCK}$				

FIG. 137c
TIMING DIAGRAM - INSTRUCTION 50 (ADS) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK	$(\overline{DLSN}\cdot\overline{DDC0}$ $+DSTZ\cdot FSIN$ $\cdot\overline{DSE9}\cdot\overline{FRNZ}$ $\cdot\overline{DLSN})\Delta COR$			COR set during unlike-signs addition if no carry from character 0 sum or if sum is minus zero.
			RCK		Stop Program Processor Clock Generator. Wait for memory synchronizing signal.
	TL0				
	QTC0	FCORΔ(WR1, TL2)	FCORΔ(WR0, TL0)		If decimal correction required, go to W06S block.
		$\overline{FCOR}\Delta$TL2	$\overline{FCOR}\Delta$(WR0, TL0)		If no decimal correction required, go to W04S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W06S	TL2				
	QTCK			QSEX	Transfer result to E-Register.
	TL3	DCE			DCE set to indicate decimal operation.
	QTCK			QCER	Complement of result formed in E-Register.
			MSX		
		ESX			Complement of result applied to Adder.
		CAY			Add one to complement of result to form 10's complement of result.
	TL4		CRE		
	QTCK				
	TL5				
	QTCK		RCK		Stop Program Processor Clock Generator.

FIG. 137d
TIMING DIAGRAM - INSTRUCTION 50 (ADS) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				Wait for memory synchronizing signal.
	$\overline{QTC0}$	TL2	WR1, TL0		Go to W04S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W04S	TL2			DABX	Transfer address of accumulator word A location to Memory. DABX is present during entire W04 block.
				(FSIN·$\overline{FCOR}$+$\overline{FSIN}$ ·FCOR)ΔDS55	Sign of result is minus if operand is minus and no correction of sum required or if operand is plus and correction required.
	$\overline{QTCK}$	MWR			Initiate write operation in Memory.
				QSMA, QSMB, QSMC	Transfer result to M-Register.
		DDC0·QSMC ΔCRE			CRE set if carry from result character 0.
	TL3		DCE		
	$\overline{QTCK}$		$\overline{MSX}$, $\overline{COR}$, $\overline{ESX}$, CAY		
	TL4	FCRE·FLSN ΔFVO			Reset Overflow flip-flop if carry from result character 0 during like-signs addition.
	$\overline{QTCK}$				
	TL5				
	QTCK	$\overline{FCOR}$ΔPSX			Transfer address of next instruction to Adder.
			RCK		Stop Program Processor Clock Generator.

FIG. 137e
TIMING DIAGRAM - INSTRUCTION 50 (ADS) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				Wait for memory synchronizing signal.
	QTC0		WR2		Go to W00 block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.

FIG. 138a

TIMING DIAGRAM - INSTRUCTION 60 (SDS)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - (Fig. 119)					
W01 - If development of address required (Fig. 121)					
W04R	TL0			DABX	Transfer address of accumulator word A to Memory. DABX is present during entire W04 block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1		DSX	DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	QTCK	MSX			Transfer accumulator word A to Adder when it is stored in M-Register.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for DMDA from Memory to start Clock Generator.
	QTC0	WR0, TL0	TL2		Go to W05 block.
	QTRK	RCK			Start Clock Generator.
W05	TL0			DDBX	Transfer operand address to Memory. DDBX is present during entire W05 block.
	QTC0	MRD			Initiate read operation in Memory.
				QSEX	Transfer accumulator word A to E-Register.

FIG. 138b

TIMING DIAGRAM - INSTRUCTION 60 (SDS) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL1	DCE			DCE set to indicate decimal operation.
				$\overline{DAAE}\Delta$(DCM0, DCM1, DCM2, DCM3)	Clear M-Register if operand and accumulator addresses are not the same.
	QTCK	ESX			Apply accumulator word A to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Clock Generator.
	QTCK	$\overline{DAAE}\Delta MWR$			Initiate write operation in Memory if operand and accumulator addresses are not the same.
		$\overline{DMRM}\Delta SIN$			Set SIN if sign of operand in M-Register is plus.
				$(DMRM \cdot DERM + \overline{DMRM} \cdot \overline{DERM}) \Delta QCER$	Complement E-Register if subtraction operation requires unlike-signs addition to form complement of accumulator word A.
			$(DMRM \cdot DERM + \overline{DMRM} \cdot \overline{DERM}) \cdot QCER \Delta LSN$		Reset LSN if substraction operation requires unlike-signs addition.
		$(DMRM \cdot DERM + \overline{DMRM} \cdot \overline{DERM}) \cdot QCER \Delta CAY$			Add one to complement of accumulator word A to form 10's complement.
	TL3		CRE		
	QTCK				

FIG. 138c
TIMING DIAGRAM - INSTRUCTION 60 (SDS) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4				
	QTCK				
	TL5				
	QTCK	$(\overline{DLSN}\cdot\overline{DDCO}+DSTZ\cdot\underline{FSIN}\cdot\overline{DSE9}\cdot FRNZ\cdot\overline{DLSN})\Delta COR$			COR set during the unlike-signs addition if no carry from character 0 result or result is minus zero.
			RCK		Stop Program Processor Clock Generator. Wait for memory synchronizing signal.
	TL0				
	QTC0	FCORΔ(WR1, TL2)	FCORΔ(WR0, TL0)		If decimal correction required, go to W06S block.
		$\overline{FCOR}\Delta TL2$	$\overline{FCOR}\Delta$(WR0, TL0)		If no decimal correction required, go to W04S block.
	QTRK	RCK			Start Clock Generator on signal $\overline{DMBU}$ from Memory.
W06S	TL2				
	QTCK			QSEX	Transfer result to E-Register.
	TL3	DCE			DCE set to indicate decimal operation.
	QTCK			QCER	Complement of result formed in E-Register.
			MSX		
		ESX			Complement of result applied to Adder.
		CAY			Add one to complement of result to form 10's complement of result.

FIG. 138d

TIMING DIAGRAM - INSTRUCTION 60 (SDS) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4		CRE		
	QTCK				
	TL5				
	QTCK		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0	TL2	WR1, TL0		Go to W04S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W04S	TL2			DABX	Transfer address of accumulator word A location to Memory. DABX is present during entire W04 block.
				(FSIN·$\overline{FCOR+FSIN}$ ·FCOR)ΔDS55	Sign of result is minus if operand is plus and no correction or if operand is minus and correction of result is required.
	QTCK	MWR			Initiate write operation in Memory.
				QSMA, QSMB, QSMC	Transfer result to M-Register.
		DDC0·QSMC ΔCRE			CRE set if carry from result character 0.
	TL3		DCE		
	QTCK		MSX, COR, ESX, CAY		
	TL4	FCRE·FLSNΔFVO			Set Overflow flip-flop if carry from result character 0 during like-signs addition.
	QTCK				

FIG. 138e

TIMING DIAGRAM - INSTRUCTION 60 (SDS) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK	$\overline{FCOR}\Delta PSX$			Transfer address of next instruction to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		WR2		Go to W00 block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.

FIG. 139a

TIMING DIAGRAM - INSTRUCTION 54 (AMS)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - (Fig. 119)					
W01 - If development of address required (Fig. 121)					
W04R	TL0			DDBX	Transfer operand address to Memory. DDBX is present during entire W04 block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1		DSX	DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	QTCK	MSX			Transfer operand word to Adder when it is stored in M-Register.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for DMDA from Memory to start Clock Generator.
	QTC0	WR0, TL0	TL2		Go to W05 block.
	QTRK	RCK			Start Clock Generator.
W05	TL0			DABX	Transfer address of accumulator word A to Memory. DABX is present during entire W05 block.
	QTC0	MRD			Initiate read operation in Memory.
				QSEX	Transfer operand word to E-Register.

FIG. 139b

TIMING DIAGRAM - INSTRUCTION 54 (AMS) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL1	DCE			DCE set to indicate decimal operation.
				$\overline{\text{DAAE}}\Delta$(DCM0, DCM1, DCM2, DCM3)	Clear M-Register if operand and accumulator addresses are not the same.
	QTCK	ESX			Apply operand word to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Clock Generator.
	QTCK	$\overline{\text{DAAE}}\Delta$MWR			Initiate write operation in Memory if operand and accumulator addresses are not the same.
		DMRMΔSIN			Set SIN if sign of accumulator word A in M-Register is minus.
				$(\overline{\text{DMRM}}\cdot\text{DERM}+\text{DMRM}\cdot\overline{\text{DERM}})\Delta$QCER	Complement E-Register if unlike-signs addition to form complement of operand.
			$(\overline{\text{DMRM}}\cdot\text{DERM}+\text{DMRM}\cdot\overline{\text{DERM}})\cdot$QCER ΔLSN		Reset LSN if unlike-signs addition.
		$(\overline{\text{DMRM}}\cdot\text{DERM}+\text{DMRM}\cdot\overline{\text{DERM}})\cdot\text{QCER}\Delta\text{CAY}$			Add one to complement of operand word to form 10's complement.
	TL3		CRE		
	QTCK				
	TL4				
	QTCK				

FIG. 139c

TIMING DIAGRAM - INSTRUCTION 54 (AMS) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK	$(\overline{DLSN} \cdot \overline{DDC0} + DSTZ \cdot \underline{FSIN} \cdot \underline{DSE9} \cdot \overline{FRNZ} \cdot \overline{DLSN}) \Delta COR$			COR set during unlike-signs addition if no carry from character 0 sum or if sum is minus zero.
			RCK		Stop Program Processor Clock Generator. Wait for memory synchronizing signal.
	TL0				
	QTC0	FCORΔ(WR1, TL2)	FCORΔ(WR0, TL0)		If decimal correction required, go to W06S block.
		$\overline{FCOR}\Delta$TL2	$\overline{FCOR}\Delta$(WR0, TL0)		If no decimal correction required, to to W04S block.
	QTRK	RCK			Start Clock Generator on signal $\overline{DMBU}$ from Memory.
W06S	TL2				
	QTCK			QSEX	Transfer result to E-Register.
	TL3	DCE			DCE set to indicate decimal operation.
	QTCK			QCER	Complement of result formed in E-Register.
			MSX		
		ESX			Complement of result applied to Adder.
		CAY			Add one to complement of result to form 10's complement of result.
	TL4		CRE		
	QTCK				

FIG. 139d

TIMING DIAGRAM - INSTRUCTION 54 (AMS) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0	TL2	WR1, TL0		Go to W04S block.
	QTRK	RCK			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.
W04S	TL2			DDBX	Transfer operand address to Memory. DDBX is present during entire W04 block.
				$(FSIN \cdot \overline{FCOR} + \overline{FSIN} \cdot FCOR) \Delta \overline{DS55}$	Sign of result is minus if accumulator word A was minus and no correction of sum required or if accumulator word A was plus and correction required.
	QTCK	MWR			Initiate write operation in Memory to store result in operand location.
				QSMA, QSMB, QSMC	Transfer result to M-Register.
		DDC0 · QSMC ΔCRE			CRE set if carry from result character 0.
	TL3		DCE		
	QTCK		MSX, COR, ESX, CAY		
	TL4	FCRE · FLSN ΔFVO			Set Overflow flip-flop if carry from result character 0 during like-signs addition.
	QTCK				

FIG. 139e

TIMING DIAGRAM - INSTRUCTION 54 (AMS) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK	$\overline{FCOR}\wedge PSX$			Transfer address of next instruction to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		WR2		Go to W00 block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.

FIG. 140a

TIMING DIAGRAM - INSTRUCTIONS 51(ADD), 52(ADT) AND 53(ADQ)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					
W01 - If development of address required (Fig. 121)					
W04R	TL0			DABX	Transfer address of accumulator word to Memory. DABX is present during entire W04 block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1		DSX	DCM0, DCM1, DCM2, DCM3	Clear M-Register.
				$\overline{\text{DAAC}}\Delta$DDCD	Count D-Register down.
	QTCK	$\overline{\text{DVHN}}\Delta$MSX			Transfer accumulator word to Adder when it is stored in M-Register, unless accumulator word is outside working Accumulator.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for DMDA from Memory to start Clock Generator.
	QTC0	WR0, TL0	TL2		Go to W05 block.
	QTRK	RCK			Start Clock Generator.

FIG. 140b

TIMING DIAGRAM - INSTRUCTIONS 51(ADD), 52(ADT) AND 53(ADQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W05	TL0			DDBX	Transfer operand address to Memory. DDBX is present during entire W05 block.
	QTC0	MRD			Initiate read operation in Memory.
				QSEX	Transfer accumulator word to E-Register.
	TL1	DCE			DCE set to indicate decimal operation.
				$\overline{\text{DAAE}}\Delta$(DCM0, DCM1, DCM2, DCM3)	Clear M-Register, if operand and accumulator addresses are not the same.
	QTCK	ESX			Apply accumulator word to Adder.
		FCREΔCAY			Set CAY if carry from previous sum.
			DAGIΔMSX		Inhibit transfer of operand to Adder if addressed word of working Accumulator is greater than length specified by instruction.
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Clock Generator.

FIG. 140c

TIMING DIAGRAM - INSTRUCTIONS 51(ADD), 52(ADT) AND 53(ADQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	QTCK	$\overline{\text{DAAE}}\Delta\text{MWR}$			Initiate write operation in Memory, if operand and accumulator addresses are not the same.
		DMRMΔSIN			Set SIN if sign of operand in M-Register is minus.
				$(\overline{\text{DMRM}}\cdot\text{DERM}+\text{DMRM}\cdot\overline{\text{DERM}})\Delta\text{QCER}$	Complement E-Register if unlike-signs addition to form complement of accumulator word.
			$(\overline{\text{DMRM}}\cdot\text{DERM}+\text{DMRM}\cdot\overline{\text{DERM}})\text{QCER}\Delta\text{LSN}$		Reset LSN if unlike-signs addition.
		$(\overline{\text{DMRM}}\cdot\text{DERM}+\text{DMRM}\cdot\overline{\text{DERM}})\cdot\text{QCER}\Delta\text{CAY}$			Add one to complement of accumulator word in E-Register to form 10's complement.
	TL3		CRE	DLAO(DDBI·FPL1·FPL2+DTPI·DPNQ+DQDI)ΔDILX	During last accumulator operation, change accumulator working length if less than that specified by instruction.
	QTCK				
	TL4				
	$\overline{\text{QTCK}}$				
	TL5				
	QTCK	$\overline{\text{DLSN}}\cdot\text{DLAO}\cdot(\overline{\text{DDCO}}+\text{DSTZ}\cdot\text{FSIN}\cdot\text{DSE9}\cdot\overline{\text{FRNZ}})\Delta\text{COR}$			COR set during last accumulator operation of unlike-signs addition if no carry from character 0 sum or if sum is minus zero.
			RCK		Stop Program Processor Clock Generator. Wait for memory synchronizing signal.

FIG. 140d

TIMING DIAGRAM - INSTRUCTIONS 51(ADD), 52(ADT) AND 53(ADQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				
	QTC0	TL2	WR0, TL0		Go to W04S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W04S	TL2			DABX	Transfer address of accumulator word location to Memory. DABX is present during entire W04 block.
				DAAC(FSIN·$\overline{FCOR}$ +$\overline{FSIN}$·FCOR)ΔDS55	Sign of result is minus if operand 1 was minus and no correction of result required or if operand 1 was plus and correction required. Issues during addition of accumulator word A.
	QTCK	MWR			Initiate write operation in Memory to store result in addressed accumulator word location.
				QSMA, QSMB, QSMC	Transfer result to M-Register.
		DDC0·QSMC ΔCRE			CRE set if carry from result character 0.
	TL3		DCE		
	QTCK		MSX, ESX, CAY		
	TL4	FCRE·FLSN ·DLA0ΔFV0			Set Overflow flip-flop if carry from result character 0 during last accumulator operation of like-signs addition.
	QTCK				
	TL5				
	QTCK	DLA0·$\overline{FCOR}$ ΔPSX			Transfer address of next instruction to Adder, if last accumulator operation and no correction required.
			RCK		Stop Program Processor Clock Generator.

FIG. 140e

TIMING DIAGRAM - INSTRUCTIONS 51(ADD), 52(ADT) AND 53(ADQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				Wait for memory synchronizing signal.
	QTC0		$\overline{FCOR}$·DLAO△WR2		If FCOR+$\overline{DLAO}$, go to W04R block. Go to W00 block, if last accumulator operation and no correction.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
				DACU	Advance count in Accumulator Counter Register.
W04R					Repeat W04R block to read accumulator word B from Memory.
W05					Repeat W05 block to read second operand word from Memory and add to accumulator word B.
W04S					Repeat W04S block to store result 2 in accumulator word B location in Memory.
W04R					Repeat W04R block to read result 1 in accumulator word A location from Memory.
W06S	TL2				
	$\overline{QTCK}$			QSEX	Transfer result to E-Register.
	TL3	DCE			DCE set to indicate decimal operation.
	$\overline{QTCK}$			QCER	Complement of result formed in E-Register.
			MSX		
		ESX			Complement of result applied to Adder.
		DAAC△CAY			Add one to complement of result 1 to form 10's complement of result 1.
			DLAO△COR		
	TL4		CRE		
	$\overline{QTCK}$				

FIG. 140f

TIMING DIAGRAM - INSTRUCTIONS 51(ADD), 52(ADT) AND 53(ADQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0	TL2	WR1, TL0		Go to W04S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W04S					Repeat W04S block to store corrected result 1 in accumulator word A location in Memory.
W04R					Repeat W04R block to read result 2 in accumulator word B location from Memory.
W06S					Repeat W06S block to correct result 2.
W04S					Repeat W04S block to store corrected result 2 in accumulator word B location in Memory. Go to W00 block.

FIG. 141a

TIMING DIAGRAM - INSTRUCTIONS 61(SDD), 62(SDT) AND 63(SDQ)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					
W01 - If development of address required (Fig. 121)					
W04R	TL0			DABX	Transfer address of accumulator word to Memory. DABX is present during entire W04 block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1		DSX	DCM0, DCM1, DCM2, DCM3	Clear M-Register.
				$\overline{\text{DAAC}\Delta\text{DDCD}}$	Count D-Register down.
	QTCK	$\overline{\text{DVHN}\Delta\text{MSX}}$			Transfer accumulator word to Adder when it is stored in M-Register, unless accumulator word is outside working Accumulator.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for DMDA from Memory to start Clock Generator.
	QTC0	WR0, TL0	TL2		Go to W05 block.
	QTRK	RCK			Start Clock Generator.
W05	TL0			DDBX	Transfer operand address to Memory. DDBX is present during entire W05 block.
	$\overline{\text{QTC0}}$	$\overline{\text{MRD}}$			Initiate read operation in Memory.
				QSEX	Transfer accumulator word to E-Register.

FIG. 141b

TIMING DIAGRAM - INSTRUCTIONS 61(SDD), 62(SDT) AND 63(SDQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL1	DCE			DCE set to indicate decimal operation.
				$\overline{\text{DAAE}}\Delta$(DCM0, DCM1, DCM2, DCM3)	Clear M-Register, if operand and accumulator addresses are not the same.
	$\overline{\text{QTCK}}$	ESX			Apply accumulator word to Adder.
		FCREΔCAY			Set CAY if carry from previous sum.
			DAGIΔMSX		Inhibit transfer of operand to Adder if addressed word of working Accumulator is greater than length specified by instruction.
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	$\overline{\text{QTRK}}$	RKC			Start Clock Generator.
	QTCK	$\overline{\text{DAAE}}\Delta$MWR			Initiate write operation in Memory, if operand and accumulator addresses are not the same.
		$\overline{\text{DMRM}}\Delta$SIN			Set SIN if sign of operand in M-Register is plus.
				(DMRM·DERM+$\overline{\text{DMRM}}$ ·$\overline{\text{DERM}}$)ΔQCER	Complement E-Register if unlike-signs addition to form complement of accumulator word.
			(DMRM·DERM+$\overline{\text{DMRM}}$ ·$\overline{\text{DERM}}$) QCERΔLSN		Reset LSN if unlike-signs addition.
		(DMRM·DERM +$\overline{\text{DMRM}}$·$\overline{\text{DERM}}$) ·QCERΔCAY			Add one to complement of accumulator word A to form 10's complement.

FIG. 141c

TIMING DIAGRAM - INSTRUCTIONS 61(SDD), 62(SDT) AND 63(SDQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL3		CRE	DLAO(DDBI·FPL1 ·FPL2+DTPI·DPNQ +DQDI)ΔDILX	During last accumulator operation, change accumulator working length if less than that specified by instruction.
	QTCK				
	TL4				
	QTCK				
	TL5				
	QTCK	$\overline{DLSN}\cdot DLAO \cdot(\overline{DDC0}+DSTZ \cdot\overline{FSIN}\cdot DSE9 \cdot\overline{FRNZ})\Delta COR$			COR set during last accumulator operation of unlike-signs addition if no carry from character 0 sum or sum is minus zero.
			RCK		Stop Program Processor Clock Generator. Wait for memory synchronizing signal.
	TL0				
	QTC0	TL2	$\overline{WR0}$, $\overline{TL0}$		Go to W04S block.
	QTRK	RCK			Start Clock Generator on signal $\overline{DMBU}$ from Memory.
W04S	TL2			DABX	Transfer address of accumulator word location to Memory. DABX is present during entire W04 block.
				DAAC(FSIN·$\overline{FCOR}$ +$\overline{FSIN}$·FCOR)ΔDS55	Sign of result is minus if operand 1 was plus and no correction or if operand 1 was minus and correction of sum is required. Issues during addition of accumulator word A.
	QTCK	MWR			Initiate write operation in Memory to store result in addressed accumulator word location.
		DDC0·QSMC ΔCRE		QSMA, QSMB, QSMC	Transfer result to M-Register. CRE set if carry from result character 0.

FIG. 141d

TIMING DIAGRAM - INSTRUCTIONS 61(SDD), 62(SDT) AND 63(SDQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL3		DCE		
	$\overline{QTCK}$		MSX, ESX, CAY		
	TL4	FCRE·FLSN ·DLA0ΔFV0			Set Overflow flip-flop if carry from result character 0 during last accumulator operation of like-signs addition.
	$\overline{QTCK}$				
	TL5				
	QTCK	DLA0·$\overline{FCOR}$ ΔPSX			Transfer address of next instruction to Adder if last accumulator operation and no correction required.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		$\overline{FCOR}$·DLA0ΔWR2		If FCOR+$\overline{DLA0}$, go to W04R block. Go to W00 block if last accumulator operation and no correction of result.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
				DACU	Advance count in Accumulator Counter Register.
W04R					Repeat W04R block to read accumulator word B from Memory.
W05					Repeat W05 block to read second operand word from Memory and add to accumulator word B.
W04S					Repeat W04S block to store result 2 in accumulator word B location in Memory.

FIG. 141e

TIMING DIAGRAM - INSTRUCTIONS 61(SDD), 62(SDT) AND 63(SDQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W04R					Repeat W04R block to read result 1 in accumulator word A location from Memory.
W06S	TL2				
	$\overline{QTCK}$			QSEX	Transfer result to E-Register.
	TL3	DCE			DCE set to indicate decimal operation.
	$\overline{QTCK}$			QCER	Complement of result formed in E-Register.
		ESX	MSX		
		DAAC△CAY			Complement of result applied to Adder. Add one to complement of result 1 to form 10's complement of result.
			DLAO△COR		
	TL4		CRE		
	$\overline{QTCK}$				
	TL5				
	$\overline{QTCK}$		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	$\overline{QTC0}$	TL2	WR1, TL0		Go to W04S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W04S					Repeat W04S block to store corrected result 1 in accumulator word A location in Memory.
W04R					Repeat W04R block to read result 2 in accumulator word B location from Memory.

FIG. 141F

TIMING DIAGRAM - INSTRUCTIONS 61(SDD), 62(SDT) AND 63(SDQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W06S					Repeat W06S block to correct result 2.
W04S					Repeat W04S block to store corrected result 2 in accumulator word B location in Memory. Go to W00 block.

FIG. 142a

TIMING DIAGRAM - INSTRUCTIONS 55(AMD), 56(AMT) AND 57(AMQ)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					
W01 - If development of address required (Fig. 121)					
W04R	TL0			DDBX	Transfer operand address to Memory. DDBX is present during entire W04 block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1		DSX	DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	QTCK	MSX			Transfer operand word to Adder when it is stored in M-Register.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for DMDA from Memory to start Clock Generator.
	QTC0	WR0, TL0	TL2		Go to W05 block.
	QTRK	RCK			Start Clock Generator.
W05	TL0			DABX	Transfer address of accumulator word to Memory. DABX is present during entire W05 block.
	QTC0	MRD			Initiate read operation in Memory.
				QSEX	Transfer operand word to E-Register.

FIG. 142b

TIMING DIAGRAM - INSTRUCTIONS 55(AMD), 56(AMT) AND 57(AMQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL1	DCE			DCE set to indicate decimal operation.
				$\overline{DAAE}\Delta$(DCM0, DCM1, DCM2, DCM3)	Clear M-Register if operand and accumulator addresses are not the same.
	$\overline{QTCK}$	FCREΔCAY			Set CAY if carry from previous sum.
		ESX			Apply operand word to Adder.
			DAGLΔMSX		Inhibit transfer of accumulator word to Adder if addressed accumulator word is outside working Accumulator.
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	$\overline{QTRK}$	RCK			Start Clock Generator.
	QTCK	$\overline{DAAE}\Delta$MWR			Initiate write operation in Memory, if operand and accumulator addresses are not the same.
		DMRMΔSIN			Set SIN if sign of accumulator word in M-Register is minus.
				$(\overline{DMRM}\cdot DERM+DMRM\cdot\overline{DERM})\Delta QCER$	Complement E-Register if unlike-signs addition to form complement of operand.
			$(\overline{DMRM}\cdot DERM+DMRM\cdot\overline{DERM})$ QCERΔLSN		Reset LSN if unlike-signs addition.
		$(\overline{DMRM}\cdot\overline{DERM}+DMRM\cdot\overline{DERM})\cdot QCER\Delta CAY$			Add one to complement of operand word to form 10's complement.
	TL3		CRE		
	$\overline{QTCK}$				

FIG. 142c

TIMING DIAGRAM - INSTRUCTIONS 55(AMD), 56(AMT) AND 57(AMQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4				
	$\overline{QTCK}$				
	TL5				
	QTCK	$\overline{DLSN}\cdot DLAO \cdot(DDCO+DSTZ \cdot FSIN\cdot DSE9 \cdot\overline{FRNZ})\Delta COR$			COR set during last accumulator operation of unlike-signs addition if no carry from character 0 sum or if sum is minus zero.
			RCK		Stop Program Processor Clock Generator. Wait for memory synchronizing signal.
	TL0				
	$\overline{QTCO}$	TL2	WRO, TL0		If decimal correction required, go to W04S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W04S	TL2			DDBX	Transfer operand address to Memory. DDBX is present during entire W04 block.
				$DAAC(FSIN\cdot\overline{FCOR} + \overline{FSIN}\cdot FCOR)\Delta\overline{DS55}$	Sign of result is minus if accumulator word A was minus and no correction of result required or if accumulator word A was plus and correction required. Issues during addition of accumulator word A.
	QTCK	MWR			Initiate write operation in Memory to store result in addressed operand location.
				QSMA, QSMB, QSMC	Transfer result to M-Register.
		DDCO·QSMC ΔCRE			CRE set if carry from result character 0.
	TL3		DCE		
	QTCK		MSX, ESX, CAY		

FIG. 142d

TIMING DIAGRAM - INSTRUCTIONS 55(AMD), 56(AMT) AND 57(AMQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4	FCRE·FLSN ·DLA0ΔFV0			Set Overflow flip-flop if carry from result character 0 during last accumulator operation of like-signs addition.
		DLA0·FCOR ΔDSX			Transfer address in D-Register to Adder if addition completed and correction of result necessary.
	$\overline{QTCK}$				
	TL5			$\overline{DLA0}$ΔDDCD	Count D-Register down if not last accumulator operation.
				FDSX·FIR0ΔDX00 FDSX·FIR1ΔDX01	Address in D-Register modified in Adder to address operand 1 location in Memory.
	QTCK	DLA0·$\overline{FCOR}$ ΔPSX			Transfer address of next instruction to Adder if last accumulator operation and no correction required.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		$\overline{FCOR}$·DLA0ΔWR2		If FCOR+$\overline{DLA0}$, go to W04R block. Go to W00 block, if last accumulator operation and no correction of result.
	QTRK	RCK			Start Clock Generator on signal $\overline{DMBU}$ from Memory.
				DACU	Advance count in Accumulator Counter Register.
W04R					Repeat W04R block to read operand 2 from Memory.

FIG. 142e

TIMING DIAGRAM - INSTRUCTIONS 55(AMD), 56(AMT) AND 57(AMQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W05					Repeat W05 block to read accumulator word B from Memory and add to operand 2.
W04S					Repeat W04S block to store result 2 in operand 2 location in Memory.
W04R					Repeat W04R block to read result 1 in operand 1 location from Memory.
W06S	TL2				
	$\overline{QTCK}$			QSEX	Transfer result to E-Register.
	TL3	DCE			DCE set to indicate decimal operation.
	$\overline{QTCK}$			QCER	Complement of result formed in E-Register.
			MSX		
		ESX			Complement of result applied to Adder.
		DAAC△CAY			Add one to complement of result to form 10's complement of result.
			DLAO△COR		
	TL4		CRE		
	$\overline{QTCK}$				
	TL5				
	$\overline{QTCK}$		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	$\overline{QTCO}$	TL2	WR1, TL0		Go to W04S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W04S					Repeat W04S block to store corrected result 1 in operand 1 location in Memory.

FIG. 142f

TIMING DIAGRAM - INSTRUCTIONS 55(AMD), 56(AMT) AND 57(AMQ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W04R					Repeat W04R block to read result 2 in operand 2 location from Memory.
W06S					Repeat W06S block to correct result 2.
W04S					Repeat W04S block to store corrected result 2 in operand 2 location in Memory. Go to W00 block.

FIG. 143a

TIMING DIAGRAM - INSTRUCTIONS 34 (ABM) AND 35 (SBM)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - (Fig. 119)					
W01 - If development of first address required (Fig. 12I)					
W02	TL0			DDBX	Transfer address of first memory location to Memory.
	QTCK	MRD			Initiate read operation in Memory.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
			DSX		
	QTCK	MSX			Transfer operand 1 from M-Register to Adder.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for signal DMDA from Memory to start Clock Generator.
	QTRK	RCK			Start Clock Generator.
	QTCK	MWR			Initiate write operation in Memory to restore operand 1 to first memory location.
	TL3				
	QTCK				

FIG. 143b

TIMING DIAGRAM - INSTRUCTIONS 34 (ABM) AND 35 (SBM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4				
	QTCK			QSEX	Transfer operand 1 to E-Register.
	TL5				
	QTCK			DSBMΔQCER	Complement contents of E-Register if instruction 35.
			MSX		
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTCO	WR0	WR1		Go to W01 block.
	QTRK	RCK			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.
W01 - To develop second address (Fig. 121 except as indicated)	TL2				
	QTCK	$\overline{\text{DVCA}}$ΔMWR			Inhibit restoration of operand 2 when address of second memory location obtained.
		MSX			Transfer operand 2 from M-Register to Adder.
		DIMMΔESX			If second address obtained, transfer operand 1 in E-Register to Adder to perform addition.
	TL3				
	QTCK	DSBM·DVCA ΔCAY			Add one to 1's complement in E-Register to form 2's complement.

FIG. 143c

TIMING DIAGRAM - INSTRUCTIONS 34 (ABM) AND 35 (SBM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				
	QTC0	WR3, WR2, WR1, TL2	TL0		Go to W15S block.
W15S	TL2			$\overline{DFXW}\Delta DDBX$ DFXWΔDQBX	If ACF of instruction word $\neq$ 1-6, transfer address of second memory location from D-Register to Memory; if ACF of instruction word = 1-6, transfer fixed index location address from Q-Register to Memory.
	QTCK	(DABM·$\overline{MC23}$ +DSBM·MC23) ΔCRE			Set CRE if carry from sum.
		MWR			Initiate write operation to store result in second memory location.
				QSMA, QSMB, QSMC	Transfer result from Adder to M-Register.
	TL3				
	QTCK		ESX, MSX, PSX, CAY		
	TL4	FCREΔFVO			Set FVO if CRE set.
	QTCK				
	TL5				
	QTCK	PSX			Transfer address of next instruction to Adder.
			RCK		Stop Program Processor Clock Generator.

FIG. 143d
TIMING DIAGRAM - INSTRUCTIONS 34 (ABM) AND 35 (SBM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				
	QTC0		WR3, WR2, WR1, WR0		Go to W00 block.
	QTRK	RCK			Start Program Processor Clock Generator on synchronizing signal DMBU from Memory.

FIG. 144a

TIMING DIAGRAM - INSTRUCTION 33 (AMI)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					
W01 - If development of first address required (Fig. 121)					
W01 - To develop second address (Fig. 121 except as indicated).	TL2				
	QTCK	$\overline{DVCA}\Delta MWR$			Inhibit restoration of operand when second address obtained.
		MSX			Apply operand to adder inputs.
		DIMMΔESX			Apply instruction word in E-Register to adder inputs to perform addition.
	TL0				
	QTC0	WR3, WR2, WR1, TL2	TL0		Go to W15S block.
W15S	TL2			$\overline{DFXW}\Delta DDBX$	If ACF of instruction word $\neq$ 1-6, address of second location transferred from D-Register to Memory.
				DFXWΔDQBX	If ACF of instruction word = 1-6, fixed index location address transferred from Q-Register to Memory.
	QTCK	MWR			Initiate write operation in Memory.
				QSMA, QSMB	Transfer address sum (bits 0-14) from Adder to M-Register.
	TL3				
	QTCK		ESX, MSX, PSX, CAY		
	TL4				
	QTCK				

FIG. 144b

TIMING DIAGRAM - INSTRUCTION 33 (AMI) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK	PSX			Transfer address of next instruction to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0	WR3, WR2, WR1, WR0			Go to W00 block.
	QTRK	RCK			Start Program Processor Clock Generator on synchronizing signal DMBU from Memory.

FIG. 145a

TIMING DIAGRAM - INSTRUCTION 03 (CAA)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - (Fig. 119)					
W01 - If address development required (Fig. 121)					
W08	TL0			DABX	Transfer address of accumulator word to Memory. DABX is present during entire W08 block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
				$\overline{\text{DAAC}\Delta\text{DDCD}}$	Count D-Register down if not accumulator word A.
	QTCK	MSX			Transfer accumulator word to Adder when it is stored in M-Register.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for DMDA from Memory to start Clock Generator.
	QTRK	RCK			Start Clock Generator.
	QTCK	MWR			Initiate write operation in Memory to restore accumulator word.
	TL3				
	QTCK			QSEX	Transfer accumulator word to E-Register.

FIG. 145b

TIMING DIAGRAM - INSTRUCTION 03 (CAA) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4				
	QTCK			QCER	Complement E-Register to form 1's complement of accumulator word.
			MSX		
	TL5				
	QTCK		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0	WR0			Go to W09 block.
	QTRK	RCK			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.
W09	TL0			DDBX	Transfer operand address to Memory. DDBX is present during entire W09 block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1		DAACAGRE,LES		Reset comparison indicator flip-flops.
				DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	QTCK	ESX			Apply 1's complement of accumulator word in E-Register to Adder.
		CAY			Add one to 1's complement of accumulator word in Adder to form 2's complement.
		MSX			Apply operand in M-Register to Adder.
			RCK		Stop Program Processor Clock Generator.

FIG. 145c

TIMING DIAGRAM - INSTRUCTION 03 (CAA) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Clock Generator.
	QTCK	MWR			Initiate write operation in Memory to restore operand.
	TL3				
	QTCK				
	TL4	MC23ΔGRE	MC23ΔLES		If borrow from result, set GRE and reset LES.
		$\overline{DSE4} \cdot \overline{MC23}$ ΔLES	$\overline{DSE4} \cdot \overline{MC23}$ΔGRE		If carry from result and result ≠ 0, set LES and reset GRE.
	QTCK				
	TL5				
	QTCK	DLAOΔPSX			If last accumulator operation, transfer address of next instruction word to Adder.
			CAY, MSX, ESX		
			RCK		Stop Program Processor Clock Generator.

FIG. 145d

TIMING DIAGRAM - INSTRUCTION 03 (CAA) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				Wait for memory synchronizing signal.
	QTC0			DACU	Advance count in Accumulator Counter Register.
			$\overline{DLAO}\Delta WR0$		If not last accumulator operation, go to W08 block.
			DLAOΔ(WR3, WR0)		If last accumulator operation, go to W00 block.
	QTRK	RCK			Start Clock Generator on signal $\overline{DMBU}$ from Memory.

FIG. 146a
TIMING DIAGRAM - INSTRUCTION 02 (CDA)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					
W01 - If address development required (Fig. 121)					
W08	TL0			DABX	Transfer address of accumulator word to Memory. DABX is present during entire W08 block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
				$\overline{DAAC \triangle DDCD}$	Count D-Register down if not accumulator word A.
	QTCK	MSX			Transfer accumulator word to Adder when it is stored in M-Register.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for DMDA from Memory to start Clock Generator.
	QTRK	RKC			Start Clock Generator.
	QTCK	MWR			Initiate write operation in Memory to restore accumulator word.
	TL3				
	QTCK			QSEX	Transfer accumulator word to E-Register.

FIG. 146b
TIMING DIAGRAM - INSTRUCTION 02 (CDA) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4				
	QTCK			QCER	Complement E-Register to form 1's complement of accumulator word.
			MSX		
	TL5				
	QTCK		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0	WR0			Go to W09 block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W09	TL0			DDBX	Transfer operand address to Memory. DDBX is present during entire W09 block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1	DCE			DCE set to indicate decimal operation.
			DAACΔ(GRE,LES)		Reset comparison indicator flip-flops.
				DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	QTCK	ESX			Apply 1's complement of accumulator word in E-Register to Adder.
		CAY			Add one to 1's complement of accumulator word in Adder to form 2's complement.
		MSX			Apply operand in M-Register to Adder.
			RCK		Stop Program Processor Clock Generator.

FIG. 146c

TIMING DIAGRAM - INSTRUCTION 02 (CDA) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Clock Generator.
	QTCK	MWR			Initiate write operation in Memory to restore operand.
		DAAC·DMRM ΔSIN			Set SIN if operand 1 in M-Register is minus, if processing accumulator word A.
			DAAC($\overline{DMRM}$·FE04 ·$\overline{FE05}$+DMRM·$\overline{FE04}$ ·FE05)ΔLSN		Reset LSN if accumulator word A and operand 1 have unlike-signs.
	TL3				
	QTCK				
	TL4	($\overline{FLSN}$·FSIN +FLSN·$\overline{FSIN}$ ·$\overline{DDC0}$+FSIN ·$\overline{DSE4}$·DDC0) ΔGRE	($\overline{FLSN}$·FSIN+FLSN ·$\overline{FSIN}$·$\overline{DDC0}$ +FSIN·$\overline{DSE4}$ ·DDC0)ΔLES		Set GRE and reset LES if accumulator word positive and operand negative or if both positive and borrow from result or if operand negative, result ≠ 0 and carry from result.
		($\overline{FLSN}$·$\overline{FSIN}$ +FLSN·FSIN ·$\overline{DDC0}$+FSIN ·$\overline{DSE4}$·DDC0) ΔLES	($\overline{FLSN}$·$\overline{FSIN}$+FLSN ·FSIN·$\overline{DDC0}$ +FSIN·$\overline{DSE4}$ ·DDC0)ΔGRE		Set LES and reset GRE if operand positive and accumulator word negative or if both negative and borrow from result or if operand positive, carry from result, and result ≠ 0.
	QTCK				

FIG. 146d

TIMING DIAGRAM - INSTRUCTION 02 (CDA) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5		DCE		
	QTCK	$(\overline{FLSN}+DLAO)$ ΔPSX	CAY, MSX, ESX RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		$\overline{DLAO}\cdot FLSN\Delta WR0$	DACU	Advance count in Accumulator Counter Register. If accumulator and operand have like-signs, and if not last accumulator operation, go to W08 block.
	QTRK	RCK	$(DLAO+\overline{FLSN})$ Δ(WR3, WR0)		If last accumulator operation or if accumulator and operand have unlike-signs, go to W00 block. Start Clock Generator on signal $\overline{DMBU}$ from Memory.

FIG. 147a

TIMING DIAGRAM - INSTRUCTION 04 (CMM)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - (Fig. 119)					
W01 - If development of first address required (Fig. 121)					
W02	TL0			DDBX	Transfer address of first memory location to Memory.
	QTCK	MRD			Initiate read operation in Memory.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
			DSX		
	QTCK	MSX			Transfer operand 1 from M-Register to Adder.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for signal DMDA from Memory to start Clock Generator.
	QTRK	RCK			Start Clock Generator.
	QTCK	MWR			Initiate write operation in Memory to restore operand 1 to first memory location.
	TL3				
	QTCK				
	TL4				
	QTCK			QSEX	Transfer operand 1 to E-Register.

FIG. 147b

TIMING DIAGRAM - INSTRUCTION 04 (CMM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK	PSX			Transfer address of next P-sequence word to Adder.
			MSX		
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTCO	WRO	WR1		Go to W01 block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W01 - To develop second address (Fig. 121)					Develop address of second memory location and store operand 2 in M-Register.
W09	TL0				
	QTCK				
	TL1		GRE, LES		Reset comparison indicator flip-flops.
	QTCK			QCER	Complement E-Register to form 1's complement of operand 1.
		ESX			Apply 1's complement of operand 1 in E-Register to Adder.
		CAY			Add one to 1's complement of operand 1 in Adder to form 2's complement.
		MSX			Apply operand 2 in M-Register to Adder.
			RCK		Stop Program Processor Clock Generator.

FIG. 147c

TIMING DIAGRAM - INSTRUCTION 04 (CMM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Clock Generator.
	QTCK				
	TL3				
	QTCK				
	TL4	$\overline{MC23} \cdot \overline{DSE4}\Delta GRE$	$\overline{MC23} \cdot \overline{DSE4}\Delta LES$		Set GRE and reset LES if carry from result and result $\neq 0$. (Operand 2 greater than operand 1).
		MC23ΔLES	MC23ΔGRE		Set LES and reset GRE if no carry from result. (Operand 2 less than operand 1).
	QTCK				
	TL5				
	QTCK	PSX			Transfer address of next instruction word to Adder.
			CAY, MSX, ESX		
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		WR3, WR0		Go to W00 block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.

FIG. 148a

TIMING DIAGRAM - INSTRUCTION 01 (CMI)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					Read instruction word from Memory.
W01 - If development of first address required (Fig. 121)					Develop address field of instruction word and store developed address in E-Register.
W01 - To develop second address (Fig. 121)					Develop address of second memory location and store operand from second memory location in M-Register.
W09	TL0				
	QTCK				
	TL1		GRE, LES		Reset comparison indicator flip-flops.
	QTCK			QCER	Complement E-Register to form 1's complement of instruction address field.
		ESX			Apply 1's complement of instruction address field in E-Register to Adder.
		CAY			Add one to 1's complement of instruction address field to form 2's complement.
		MSX			Apply operand in M-Register to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Clock Generator.
	QTCK				

FIG. 148b

TIMING DIAGRAM - INSTRUCTION 01 (CMI) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL3				
	QTCK				
	TL4	$\overline{MC14 \cdot DSE5}$ ⧋GRE	$\overline{MC14 \cdot DSE5}$ ⧋LES		Set GRE and reset LES if carry from bit position 14 of result and if sum in bit positions 0-14 ≠ 0. (Address field of operand greater than instruction address field.)
		MC14⧋LES	MC14⧋GRE		Set LES and reset GRE if no carry from bit position 14 of result. (Address field of operand less than instruction address field.)
	QTCK				
	TL5				
	QTCK	PSX			Transfer address of next instruction word to Adder.
			CAY, MSX, ESX		
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		WR3, WR0		Go to W00 block.
	QTRK	RCK			Start Clock Generator on signal $\overline{DMBU}$ from Memory.

FIG. 149

TIMING DIAGRAM - INSTRUCTION 10 (BRU)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - (Fig. 119 except as indicated)	TL2				
	QTCK	MSX			Apply instruction word to Adder.
	TL4				
	QTCK			$\overline{\text{DAMS}}\Delta$QSPX	Transfer address field of instruction word through Adder to P-Register if development of address field not required.
	TL5		$\overline{\text{FPIT}}\Delta$PSA		
	QTCK	$\overline{\text{DAMS}}\Delta$PSX			Apply address field of instruction word in P-Register to Adder.
	TL0				
	QTC0	DAMSΔWR0			If address development required, go to W01 block.
W01 - If address development required. (Fig. 121 except as indicated)	TL5				
	QTCK			DXIMΔQSPX	If address development completed, transfer developed address field of instruction word through Adder to P-Register.
	TL0				
	QTC0		WR0		Go to W00 block.

FIG. 150a

TIMING DIAGRAM - INSTRUCTION 17 (SPB)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - (Fig. 119)					Read instruction word from Memory and store in E-Register.
W01 - If development of first address required (Fig. 121)					Store instruction word with developed address field in E-Register.
W01 - To develop second address (Fig. 121 except as indicated).	TL2				
	QTCK	$\overline{DVCA}\Delta MWR$			Inhibit restoration of contents of specified memory location when address of location obtained.
		$DIMM \cdot \overline{FNXF}\Delta ESX$			Inhibit transfer of instruction word in E-Register to Adder when address of second memory location developed.
	TL5				
	QTCK	DVCAΔPSX			When development of address of specified memory location completed, transfer contents of P-Register to Adder.
			MSX		Inhibit transfer of instruction word in M-Register to Adder.
	TL0				
	QTC0	WR3, WR2, WR1, TL2	TL0		Go to W15S block.

FIG. 150b

TIMING DIAGRAM - INSTRUCTION 17 (SPB) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W15S	TL2			$\overline{DFXW} \triangle DDBX$	If ACF of instruction word ≠ 1-6, transfer address of specified location from D-Register to Memory.
				DFXW△DQBX	If ACF of instruction word = 1-6, transfer fixed index location address from Q-Register to Memory.
	QTCK	MWR			Initiate write operation in Memory.
				QSMA, QSMB	Transfer contents of P-Register through Adder to bit positions 0-14 of M-Register.
	TL3				
	QTCK		ESX, MSX, PSX, CAY		
	TL4				
	QTCK				
	TL5				
	QTCK	ESX			Apply instruction word in E-Register to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		WR3, WR2, WR1, WR0		Go to W00 block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.

FIG. 150c

TIMING DIAGRAM - INSTRUCTION 17 (SPB) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - During initial routine of next instruction (Fig. 119 except as indicated).	TL0				
	QTCK			QSPX	Transfer address field of instruction word through Adder to P-Register.

FIG. 151

TIMING DIAGRAM - INSTRUCTIONS 12 (BRG), 13 (BRE) AND 14 (BRL)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - (Fig. 119 except as indicated).	TL2				
	QTCK	MSX			Apply instruction word to Adder.
	TL4				
	QTCK			DBRG·$\overline{\text{DAMS}}$·FGRE△QSPX DBRE·$\overline{\text{DAMS}}$·$\overline{\text{FGRE}}$·$\overline{\text{FLES}}$ △QSPX DBRL·$\overline{\text{DAMS}}$·FLES△QSPX	If address development not required and condition specified by instruction word exists in comparison indicator flip-flops, transfer address field of instruction word through Adder to P-Register.
	TL5				
	QTCK	$\overline{\text{DAMS}}$△PSX			Apply address field of instruction word in P-Register to Adder.
	TL0				
	QTC0	DAMS△WR0			If address development required, go to W01 block.
W01 - If address development required (Fig. 121 except as indicated).	TL5				
	QTCK	DXIM·DBCT△PSX		DBRG·DXIM·FGRE△QSPX DBRE·DXIM·$\overline{\text{FGRE}}$ ·$\overline{\text{FLES}}$△QSPX DBRL·DXIM·FLES △QSPX	If address development completed and condition specified by instruction word exists in comparison indicator flip-flops, transfer developed address field of instruction word through Adder to P-Register. If address development completed and specified condition satisfied, apply address field of instruction word in P-Register to Adder.
	TL0				
	QTC0		WR0		Go to W00 block.

FIG. 152a

TIMING DIAGRAM - INSTRUCTION 15 (BRZ)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					Read instruction word from Memory and store address field in D-Register.
W01 - If address development required (Fig. 121)					Develop address field and store in D-Register.
W02	TL0			DABX	Transfer address of accumulator word to B-Gates. DABX is present during entire W02 block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
			DSX		
	QTCK	MSX			Apply accumulator word in M-Register to Adder.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for signal DMDA from Memory to start Clock Generator.
	QTRK	RCK			Start Clock Generator.
	QTCK	MWR			Initiate write operation in Memory to restore accumulator word.
	TL3				
	QTCK	$\overline{\text{DSE4}}\Delta\text{RNZ}$			RNZ set if accumulator word $\neq 0$.
	TL4				
	QTCK				

FIG. 152b

TIMING DIAGRAM - INSTRUCTION 15 (BRZ) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5	FRNZ·DLAO∆DSX			If last accumulator operation and accumulator word = 0, apply branch address in D-Register to Adder.
	QTCK	FRNZ∆PSX			If accumulator word ≠ 0, apply address of next instruction in P-sequence to Adder.
			MSX		
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		(DLAO+FRNZ)∆WR1		If last accumulator operation or if accumulator word ≠ 0, go to W00 block. If not last accumulator operation (DLAO) and if accumulator = 0 (FRNZ), repeat W02 block.
	QTRK			DACU	Advance count in Accumulator Counter Register.
		RCK			Start Clock Generator on signal DMBU from Memory.
W00 - During initial routine of next instruction (Fig. 119 except as indicated).	TL0				
	QTCK			QSPX	Transfer address field of instruction word through Adder to P-Register.

FIG. 153a
TIMING DIAGRAM - INSTRUCTION 11 (BRM)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - (Fig. 119)					Read instruction word from Memory and store address field in D-Register.
W01 - If address development required (Fig. 121)					Develop address field and store in D-Register.
W02	TL0			DABX	Transfer address of accumulator word A to Memory. DABX is present during entire W02 block.
	QTCK	MRD			Initiate read operation in Memory.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
			DSX		
	QTCK		ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for signal DMDA from Memory to start Clock Generator.
	QTRK	RCK			Start Clock Generator.
	QTCK	MWR			Initiate write operation in Memory.
	TL3				
	QTCK				
	TL4				
	QTCK				

FIG. 153b

TIMING DIAGRAM - INSTRUCTION 11 (BRM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5	DMRM△DSX			If sign of accumulator word A is minus, apply branch address in D-Register to Adder.
	QTCK	$\overline{\text{DMRM}}$△PSX			If sign of accumulator word A is plus, apply address of next instruction in P-sequence to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		WR1		Go to W00 block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W00 - During initial routine of next instruction (Fig. 119 except as indicated).	TL0				
	QTCK			QSPX	Transfer address field of instruction word through Adder to P-Register.

FIG. 154a
TIMING DIAGRAM - INSTRUCTION 16 (BRC)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					Read instruction word from Memory and store in E-Register.
W01 - If development of instruction address field required (Fig. 121)					Store instruction word with developed address field in E-Register.
W01 - To develop second address (Fig. 121 except as indicated).	TL2				
	QTCK	MSX			Apply control word in M-Register to Adder.
		$\overline{\text{DVCA}}\Delta\text{MWR}$			Inhibit restoration of control word when address of second location developed.
		$\text{DIMM}\cdot\overline{\text{FNXF}}\ \Delta\text{ESX}$			Inhibit transfer of instruction word in E-Register to Adder when address of second location developed.
	TL3	DVCAΔIRE			Set IRE when address of second location developed to add one to work count field of control word.
	QTCK				
	TL4				
	QTCK			$\text{DVCA}\cdot\overline{\text{MC23}}\Delta\text{QSEX}$	If count in work count field goes to zero, transfer control word to E-Register.
		$\text{DVCA}\cdot\text{QSEX}\ \Delta\text{CRE}$			If work count field goes to zero, set CRE.

FIG. 154b

TIMING DIAGRAM - INSTRUCTION 16 (BRC) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK	DVCA·FCRE△ESX	FCRE△MSX	DVCA·FCRE△QEEX	If work count field goes to zero, replace work count field of control word with restore count field. If work count field goes to zero, apply control word to Adder. If work count field goes to zero, reset MSX.
	TL0				
	QTC0	DVCA△(WR3, WR2, WR1, TL2)	DVCA△TL0		If address of second location developed, go to W15S block.
W15S	TL2			$\overline{\text{DFXW}}$△DDBX DFXW△DQBX	If ACF of instruction word ≠ 1-6, transfer address of control word from D-Register to Memory; if ACF = 1-6, transfer fixed index location address from Q-Register to Memory.
	QTCK	MWR		QSMA, QSMB, QSMC	Initiate write operation in Memory. Transfer control word through Adder to M-Register.
	TL3	FCRE△MPL	IRE		If work count field goes to zero, set MPL.
	QTCK		ESX, MSX		
	TL4				
	QTCK				

FIG. 154c

TIMING DIAGRAM - INSTRUCTION 16 (BRC) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK	$\overline{\text{FCRE}}\Delta$ESX			If work count field does not go to zero, apply address field of instruction word to Adder.
		FCREΔPSX			If work count field goes to zero, apply address of next instruction in P-sequence to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTCO	WR3, WR2, WR1, WR0			Go to W00 block to branch, if $\overline{\text{FCRE}}$, or to fetch next instruction, if FCRE.
	QTRK	RCK			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.
W00 - During initial routine of next instruction (Fig. 119 except as indicated).	TL0				
	QTCK			QSPX	If work count field did not go to zero during last W01 block, transfer branch address to P-Register through Adder to P-Register.

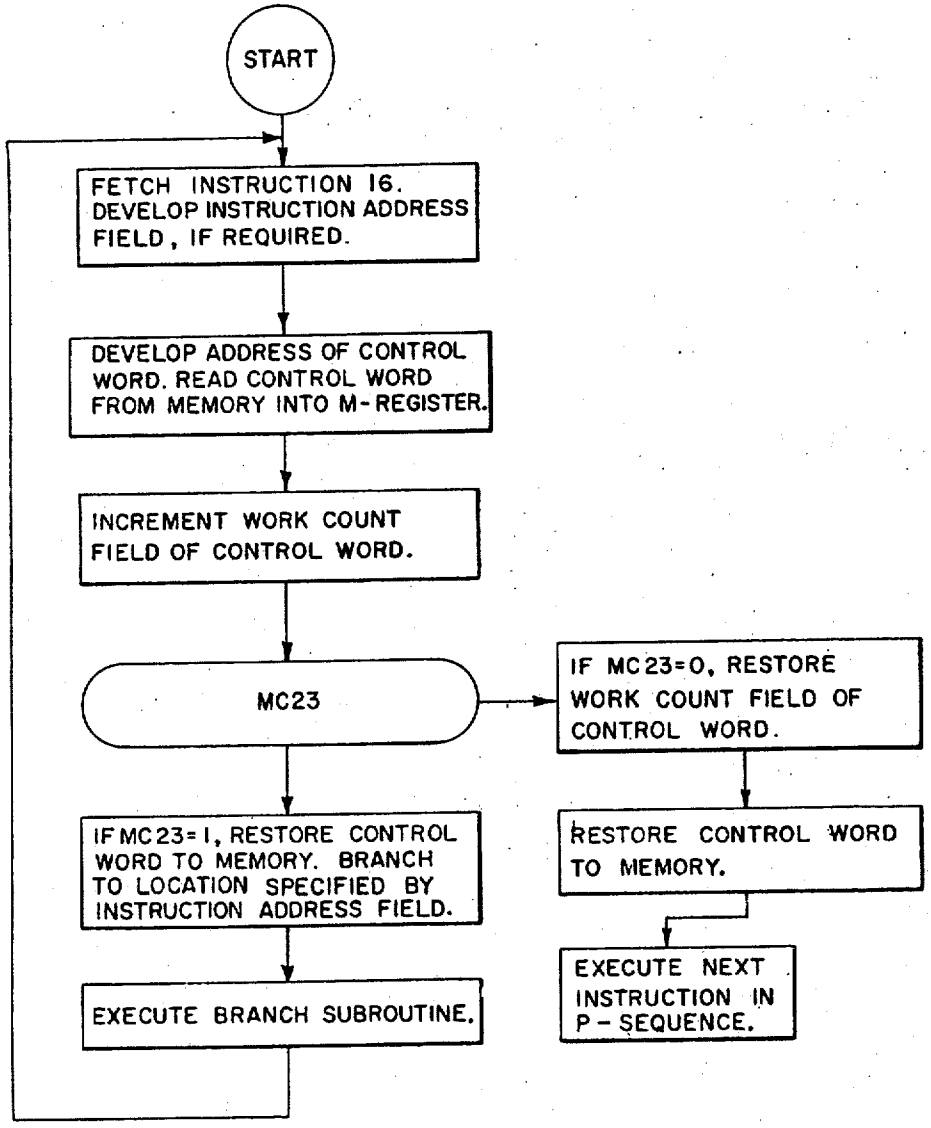


FIG. 155
FLOW DIAGRAM
INSTRUCTION 16 (BRC)

FIG. 156a

TIMING DIAGRAM - INSTRUCTIONS 24 (ANM), 23 (RIM) AND 25 (RXM)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - (Fig. 119)					Read instruction word from Memory and store address field in D-Register.
W01 - If development of first address required (Fig. 121)					Develop instruction address field and store developed address field in D-Register.
W02	TL0			DDBX	Transfer address of first memory location to Memory.
	QTCK	MRD			Initiate read operation in Memory.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
			DSX		
	QTCK	MSX			Transfer operand 1 from M-Register to Adder.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Clock Generator.
	QTCK	MWR			Initiate write operation in Memory to restore operand 1.
	TL3				
	QTCK				

FIG. 156b

TIMING DIAGRAM - INSTRUCTIONS 24 (ANM), 23 (RIM) AND 25 (RXM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4				
	QTCK			QSEX	Transfer operand 1 to E-Register.
	TL5				
	QTCK	PSX			Transfer address of next P-sequence word to Adder.
			MSX		
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0	WR0	WR1		Go to W01 block.
	QTRK	RCK			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.
W01 - To develop second address (Fig. 121)					Develop second address and store operand 2 in M-Register.
W11S	TL2				
	QTCK				
	TL3			DNSL	Preset N-Register flip-flops to count of 24.
			CRE, DCE		
	QTCK	SRI			Set SRI preparatory to shifting M- and E-Registers.
			MSX		

FIG. 156c

TIMING DIAGRAM - INSTRUCTIONS 24 (ANM), 23 (RIM) AND 25 (RXM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4				
	QTCK			QSHM	Shift operand 2 in M-Register right one bit position.
				QSHE	Shift operand 1 in E-Register right one bit position.
		QSHM·DANM ·FM00·FE00 ΔM23	QSHM·DANM ($\overline{FM00}$ +FE00)ΔM23		Instruction 24 (ANM): if both flip-flops M00 and E00 are set to the 1-state, set flip-flop M23 to the 1-state; if either M00 or E00 are reset to the 0-state, reset M23 to the 0-state. Simultaneously effect shift in M- and E-Registers.
		QSHM ·DRIM(FM00 +FE00)ΔM23	QSHM·DRIM·$\overline{FM00}$ ·FE00ΔM23		Instruction 23 (RIM): if either M00 or E00 are set to 1-state, set M23 to 1-state; if both M00 and E00 are reset to 0-state, reset M23 to 0-state. Simultaneously effect shift in M- and E-Registers.
		QSHM ·DRXM(FM00 ·FE00+FM00 ·FE00)ΔM23	QSHM·DRXM(FE00 ·FM00+FE00 ·FM00)ΔM23		Instruction 25 (RXM): if flip-flops M00 and E00 have opposite states, set M23 to 1-state; if M00 and E00 have the same state, reset M23 to the 0-state. Simultaneously effect shift in M- and E-Registers.
				QNCD	Count down N-Register by one for each shift of M- and E-Registers.

FIG. 156d

TIMING DIAGRAM - INSTRUCTIONS 24 (ANM), 23 (RIM) AND 25 (RXM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK			$\overline{\text{FSRI}}\Delta$QTLP	FSRI inhibits QTLP and advance of TL-Counter.
				QSHM, QSHE	QSHM and QSHE issue a total of 24 times, until count in N-Register = 1, to complete logical modification of operand 2 in M-Register.
			DNE1ΔSRI		Reset SRI when count in N-Register = 1.
	QTCK		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0	WR2, TL2	TL0		Go to W15S block.
	QTRK	RCK			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.
W15S	TL2			DDBX	Transfer address of second location to Memory.
	QTCK	MWR			Initiate write operation in Memory.
	TL3				
	QTCK				
	TL4				
	QTCK				

FIG. 156e

TIMING DIAGRAM - INSTRUCTIONS 24 (ANM), 23 (RIM) AND 25 (RXM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK	PSX			Transfer address of next instruction to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		WR3, WR2, WR1, WR0		Go to block W00.
	QTRK	RKC			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.

FIG. 157
TIMING DIAGRAM - INSTRUCTION 36 (LAL)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - (Fig. 119 except as indicated).					Read instruction word from Memory and store in E-Register.
	TL4			DCLA	Clear A-Register.
	QTCK				
	TL5				
	QTCK	$\overline{\text{DAMS}}\Delta\text{PSX}$			Apply address of next instruction to Adder. If no address development required, set PSX.
W01 - If address development required (Fig. 121 except as indicated).	TL4				Develop instruction address field.
	QTCK			DXIMΔQSEX	Transfer developed address field to E-Register.
	TL5				
	QTCK	DXIMΔPSX	DXIMΔESX		When address development completed, apply address of next instruction to Adder.
	TL0				
	QTC0		WR0		Go to W00 block.
W00 - During initial routine of next instruction (Fig. 119 except as indicated).	TL0			QEAX	Transfer accumulator location (bits 2-14) to A-Register and accumulator length (bits 0, 1) to flip-flops PL1, PL2.

FIG. 158a
TIMING DIAGRAM - INSTRUCTION 37 (SAL)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - (Fig. 119					Read instruction word from Memory and store address field in D-Register.
W01 - If address development required (Fig. 121)					Develop instruction address field and store developed address field in D-Register.
W03	TL0			DDBX	Transfer address of specified memory location to Memory.
	QTCK	MRD			Initiate read operation in Memory.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
				DAEX	Transfer accumulator location and length to E-Register.
	QTCK	ESX			Apply accumulator location and length to Adder.
			MSX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Clock Generator.
	QTCK	MWR			Initiate write operation in Memory.
				QSMA, QSMB	Transfer accumulator location and length (bits 0-14) through Adder to M-Register.
	TL3				
	QTCK				

FIG. 158b

TIMING DIAGRAM - INSTRUCTION 37 (SAL) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4				
	QTCK		ESX		
	TL5				
	QTCK	PSX			Apply address of next instruction to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		WR1, WR0		Go to W00 block.
	QTRK	RCK			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.

FIG. 159a

TIMING DIAGRAM - INSTRUCTION 27 (VLM)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					
W01 - If address development required (Fig. 121)					
W04R	TL0			DABX	Transfer address of accumulator word A to Memory.
	QTCK	MRD			Initiate read operation in Memory.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
			DSX		
	QTCK	MSX			Apply accumulator word A in M-Register to Adder.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for signal DMDA from Memory.
	QTC2			DCAD	Reduce count in Accumulator Counter Register by one to address accumulator word D.
		WR3, WR1, WR0	WR2		Go to W11S block.
	QTRK	RCK			Start Clock Generator.
W11S	TL2				
	QTCK	DMRM·DDAC ΔSIN			If accumulator word A is minus, set SIN.
				QSEX	Transfer accumulator word A to E-Register.
				DDACΔQSCX	Transfer character 3 of accumulator word A to CC-Register.

FIG. 159b
TIMING DIAGRAM - INSTRUCTION 27 (VLM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL3			DCLQ	Clear Q-Register.
				DCM0, DCM1, DCM2, DCM3	Clear M-Register.
			CRE		
	QTCK	SRI			Set SRI to enable contents of E-Register to be shifted.
				DDAC△QCNX	Transfer character 3 of accumulator word A to N-Register.
			MSX		
	TL4				
	QTCK			FSRI△QSHE	Shift accumulator word A in E-Register right one bit position.
		FSRI△Q00			Count in Q-Register advanced to one.
		ESX			Apply contents of E-Register to Adder.
	TL5				
	QTCK			$\overline{\text{FSRI}}$△QTLP	FSRI inhibits QTLP and advance of TL-Counter.
		FSRI△Q01	FSRI△Q00		Count in Q-Register advanced to two.
				FSRI△QSHE	Shift accumulator word A in E-Register right one bit position.
	$\overline{\text{QTCK}}$				QSHE issues and count in Q-Register advanced by one for each clock signal QTCK.
			DQE5△SRI		Reset SRI when count in Q-Register is five. Delay in flip-flop SRI permits one more shift, advancing count in Q-Register to zero.
			$\overline{\text{FSRI}}$△RCK		Stop Program Processor Clock Generator.

FIG. 159c
TIMING DIAGRAM - INSTRUCTION 27 (VLM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				Wait for memory synchronizing signal.
	$\overline{QTC0}$	TL2, WR2	WR3, WR1, WR0, TL0		Go to W04S block.
				DDAC△DACU	Advance count in Accumulator Counter Register by one to address accumulator word A.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W04S	TL2			DABX	Transfer address of accumulator word A to Memory.
	$\overline{QTCK}$	MWR			Initiate write operation in Memory.
				QSMA, QSMB, QSMC	Transfer adder outputs to M-Register.
	TL3				
	$\overline{QTCK}$		$\overline{SP2}$		
			ESX, CAY		
	TL4				
	$\overline{QTCK}$	MJE0△SP2			Set SP2 if MJE0 issues.
	TL5				
	$\overline{QTCK}$		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for synchronizing signal from Memory.
	$\overline{QTC0}$			DACU	Advance count in Accumulator Counter Register by one to address accumulator word B.
					Go to W04R block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.

FIG. 159d

TIMING DIAGRAM - INSTRUCTION 27 (VLM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W04R	TL0			DABX	Transfer address of accumulator word B to Memory.
	$\overline{QTCK}$	MRD			Initiate read operation in Memory.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	$\overline{QTCK}$	MSX			Apply accumulator word B in M-Register to Adder.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for signal DMDA from Memory.
	$\overline{QTC2}$			DCAD	Reduce count in Accumulator Counter Register by one to address accumulator word A location.
		WR3, WR1, WR0	WR2		Go to W11S block.
	$\overline{QTRK}$	RCK			Start Clock Generator.
W11S	TL2				
	$\overline{QTCK}$			QSEX	Transfer accumulator word B to E-Register.
	TL3			DCLQ	Clear Q-Register.
				DCM0, DCM1, DCM2, DCM3	Clear M-Register.
			CRE, DCE		
	$\overline{QTCK}$	SRI			Set SRI to enable contents of M- and E-Registers to be shifted.
			MSX		
	TL4				
	$\overline{QTCK}$			FSRIΔ(QSHM, QSHE)	Shift contents of M- and E-Registers right one bit position.
		FE00ΔM23			Shift E-Register into M-Register.
		FSRIΔQ00			Count in Q-Register advanced to one.
		ESX			Apply contents of E-Register to Adder.

FIG. 159c

TIMING DIAGRAM - INSTRUCTION 27 (VLD) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK			$\overline{\text{FSRI}}\Delta\text{QTLP}$	FSRI inhibits QTLP and advance of TL-Counter.
		FSRIΔQ01	FSRIΔQ00		Count in Q-Register advanced to two.
				FSRIΔ(QSHM, QSHE)	Shift contents of M- and E-Registers right one bit position.
		FE00ΔM23			Shift E-Register into M-Register.
	$\overline{\text{QTCK}}$				QSHM and QSHE issue and count in Q-Register advanced for each clock signal QTCK.
			DQE5ΔSRI		Reset SRI when count in Q-Register is five. Delay in flip-flop SRI permits one more shift, advancing count in Q-Register to zero.
			$\overline{\text{FSRI}}\Delta\text{RCK}$		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0	TL2	WR3, TL0		Go to W03S block.
	QTRK	RCK			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.
W03S	TL2			DABX	Transfer accumulator word A address to Memory.
	QTCK	MWR			Initiate write operation in Memory.
	TL3				
	$\overline{\text{QTCK}}$				
	TL4				
	$\overline{\text{QTCK}}$				
	TL5				
	QTCK		RCK		Stop Program Processor Clock Generator.

FIG. 159f

TIMING DIAGRAM - INSTRUCTION 27 (VLM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				Wait for memory synchronizing signal.
	QTC0	WR2, TL2	WR1, WR0, TL0		Go to W04S block.
				DACU	Advance count in Accumulator Counter Register to address accumulator word B location.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W04S	TL2			DABX	Transfer address of accumulator word B location to Memory.
	QTCK	MWR			Initiate write operation in Memory.
				QSMA, QSMB, QSMC	Transfer contents of E-Register through Adder to M-Register.
	TL3				
	QTCK		ESX, CAY		
	TL4				
	QTCK	MJE0ΔSP2			Set SP2 if MJE0 issues.
	TL5				
	QTCK		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for synchronizing signal from Memory.
	QTC0			DACU	Advance count in Accumulator Counter Register to address accumulator word C location.
					Go to W04R block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W04R	TL0			DABX	Transfer address of accumulator word C to Memory. DABX is present during entire W04 block.
	QTCK	MRD			Initiate read operation in Memory.

FIG. 159g

TIMING DIAGRAM - INSTRUCTION 27 (VLM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	QTCK	$\overline{\text{FMPL}}\Delta\text{MSX}$			Inhibit MSX if first multiplier character.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for signal DMDA from Memory.
	QTC2			DCAD	Reduce count in Accumulator Counter Register to address accumulator word B location.
		WR1, WR0, TL0	TL2		Go to W07 block.
	QTRK	RKC			Start Clock Generator.
W07	TL0			DDBX	Transfer address of least-significant multiplicand word to Memory. DDBX is present during entire W07 block.
	QTCK	MRD			Initiate read operation in Memory.
				QSEX	Clear E-Register.
	TL1	DCE			Enable decimal correction circuits of Adder.
		FN00ΔQ00	$\overline{\text{FN00}}\Delta\text{Q00}$		Store multiplier character in Q-Register.
		FN01ΔQ01	$\overline{\text{FN01}}\Delta\text{Q01}$		
		FN02ΔQ02	$\overline{\text{FN02}}\Delta\text{Q02}$		
		FN03ΔQ03	$\overline{\text{FN03}}\Delta\text{Q03}$		
			CS2		
				DCM0, DCM1, DCM2, DCM3	Clear accumulator word C from M-Register.
				DCLC	Clear CC-Register.

FIG. 159h

TIMING DIAGRAM - INSTRUCTION 27 (VLM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	QTCK	MSX			Apply least-significant multiplicand word in M-Register to Adder.
		ESX			
			TB0, TB1		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for signal DMDA from Memory.
	$\overline{QTRK}$	$\overline{RCK}$			Start Clock Generator.
	QTCK	$\overline{DNEZ}\Delta MOD$			If multiplier in N-Register ≠ 0, set MOD to initiate multiplication.
		MWR			Initiate write operation in Memory.
			FMPL(FSIN·$\overline{DMRM}$ +FSIN·DMRM) ΔLSN		Reset LSN if signs of multiplier and multiplicand are different.
				DBCU	Advance count in TB-Counter starting with next QTCK clock signal.
	TL3				
	$\overline{QTCK}$	$\overline{DBCU}\Delta TB0$			Advance count in TB-Counter to one.
	TL4		FMOD·$\overline{FTB1}$·FTB0 ΔCRE		Reset CRE when count in TB-Counter is one to prepare for storage of next carry.
	$\overline{QTCK}$	$\overline{DBCU}\Delta TB1$	$\overline{DBCU}\Delta TB0$		Advance count in TB-Counter to two.

FIG. 159i

TIMING DIAGRAM - INSTRUCTION 27 (VLM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK			$\overline{FMOD}\Delta QTLP$	FMOD inhibits QTLP and advance of TL-Counter.
				$FTB1 \cdot \overline{FTB0} \cdot FMOD\Delta QSEX$	When count in TB-Counter = 2, transfer adder output to E-Register.
				$FTB1 \cdot \overline{FTB0} \cdot FMOD\Delta QNCD$	When count in TB-Counter = 2, reduce multiplier count in N-Register by one.
		$DDC0 \cdot QSEX \Delta CRE$			Set CRE if carry from character 0 of adder output.
			TB1, TB0		Reset TB-Counter to zero.
				$\overline{FTB1} \cdot \overline{FTB0} \cdot FMOD \cdot FCRE\Delta DCCU$	At end of each addition, advance count in CC-Register by one if CRE is set.
	$\overline{QTCK}$				Count in TB-Counter advanced by one for each QTCK clock signal from zero to two and back to zero. QSEX issues to add least-significant multiplicand word to contents of E-Register each time count in TB-Counter = 2. QNCD also issues at this time to reduce count in N-Register by one. Above micro-operations repeated until MOD flip-flop is reset to 0-state.
			$FTB1 \cdot \overline{FTB0} \cdot DNE1 \Delta MOD$		Reset MOD when counts in TB-Counter and N-Register are two and one respectively.
			RCK		Stop Program Processor Clock Generator.

FIG. 159j

TIMING DIAGRAM - INSTRUCTION 27 (VLM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				Wait for memory synchronizing signal.
	$\overline{QTC0}$	$\overline{WR3}$, $\overline{TL2}$	$\overline{WR2}$, $\overline{TL0}$		Go to W11S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W11S	TL2				
	$\overline{QTCK}$			FCREΔDCCU	Advance count in CC-Register by one if CRE set to remember carry during last addition of W07 block.
				DBACΔQQNX	Transfer multiplier in Q-Register to N-Register.
	TL3	FMPL·DBAC ·FLSNΔE05			Set flip-flop E05 of E-Register if signs of multiplier and multiplicand are opposite.
				DCLQ	Clear Q-Register.
				DCM0, DCM1, DCM2, DCM3	Clear M-Register.
			CRE, DCE		
	$\overline{QTCK}$	$\overline{SRI}$			Set $\overline{SRI}$ to enable contents of M- and E-Registers to be shifted.
			MSX		
	TL4				
	$\overline{QTCK}$			FSRIΔ($\overline{QSHM}$, $\overline{QSHE}$)	Shift contents of M- and E-Registers right one bit position.
		FE00ΔM23			Shift E-Register into M-Register.
		FSRIΔQ00			Count in Q-Register advanced to one.
		ESX			Apply contents of E-Register to Adder.

FIG. 159k

TIMING DIAGRAM - INSTRUCTION 27 (VLM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5				
	QTCK			$\overline{\text{FSRI}}\Delta$QTLP	FSRI inhibits QTLP and advance of TL-Counter.
		FSRIΔQ01	FSRIΔQ00		Count in Q-Register advanced to two.
				FSRIΔ(QSHM, QSHE)	Shift contents of M- and E-Registers right one bit position.
		FE00ΔM23			Shift E-Register into M-Register.
	$\overline{\text{QTCK}}$				QSHM and QSHE issue and count in Q-Register advanced for each clock signal QTCK.
			DQE5ΔSRI		Reset SRI when count in Q-Register is five. Delay in flip-flop SRI permits one more shift, reducing count in Q-Register to zero.
			$\overline{\text{FSRI}}\Delta$RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	$\overline{\text{QTC0}}$	TL2	WR3, TL0		Go to W03S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W03S	TL2			DABX	Transfer accumulator word B address to Memory.
	$\overline{\text{QTCK}}$	MWR			Initiate write operation in Memory.
	TL3				
	$\overline{\text{QTCK}}$				
	TL4				
	$\overline{\text{QTCK}}$	ESX			Apply product in E-Register to Adder.
	TL5				
	$\overline{\text{QTCK}}$		RCK		Stop Program Processor Clock Generator.

FIG. 159L

TIMING DIAGRAM - INSTRUCTION 27 (VLM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				Wait for memory synchronizing signal.
	QTC0	WR2, TL2	WR1, WR0, TL0		Go to W04S block.
				DACU	Advance count in Accumulator Counter Register to address accumulator word C location.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W04S	TL2			DABX	Transfer address of accumulator word C to Memory.
	QTCK	MWR			Initiate write operation in Memory.
				QSMA, QSMB, QSMC	Transfer adder outputs to M-Register.
	TL3				
	QTCK		SP2		
			ESX, CAY		
	TL4				
	QTCK	MJE0ΔSP2			Set SP2 if MJE0 issues.
	TL5				
	QTCK		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for synchronizing signal from Memory.
	QTC0			DACU	Advance count in Accumulator Counter Register to address accumulator word D location.
					Go to W04R block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W04R	TL0			DABX	Transfer address of accumulator word D to Memory. DABX is present during entire W04 block.
	QTCK	MRD			Initiate read operation in Memory.

FIG. 159m
TIMING DIAGRAM - INSTRUCTION 27 (VLM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
			DSX	DDAC△DDCD	Form address of most-significant multiplicand word in D-Register.
	QTCK	FMPL△MSX			Inhibit MSX if first multiplier character.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for signal DMDA from Memory.
	$\overline{QTC2}$			DCAD	Reduce count in Accumulator Counter Register to address accumulator word C location.
		WR1			Go to W06S block.
	$\overline{QTRK}$	$\overline{RCK}$			Start Clock Generator.
W06S	TL2				
	$\overline{QTCK}$				
	TL3	CS2			Apply carry count in CC-Register to inputs of full adder circuits S00-S03.
		DCE			Enable decimal correction circuits of Adder.
	$\overline{QTCK}$				
	TL4		CRE		
	$\overline{QTCK}$				
	TL5				
	$\overline{QTCK}$		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for synchronizing signal from Memory.
	$\overline{QTC0}$	$\overline{WR0}$			Go to W07 block.
	QTRK	RCK			Start Clock Generator on signal $\overline{DMBU}$ from Memory.

FIG. 159n

TIMING DIAGRAM - INSTRUCTION 27 (VLM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W07	TL0			DDBX	Transfer address of most-significant multiplicand word to Memory.
	QTCK	MRD			Initiate read operation in Memory.
				QSEX	Transfer carries to E-Register.
		DDC0·QSEX ΔCRE			Set CRE if carry out of character 0 of adder output.
	TL1	FN00ΔQ00 FN01ΔQ01 FN02ΔQ02 FN03ΔQ03	$\overline{FN00}$ΔQ00 $\overline{FN01}$ΔQ01 $\overline{FN02}$ΔQ02 $\overline{FN03}$ΔQ03		Store multiplier character in Q-Register.
			CS2		
			DCACΔMPL		
				DCM0, DCM1, DCM2, DCM3	Clear accumulator word D from M-Register.
				DCLC	Clear CC-Register.
	QTCK	MSX			Apply most-significant multiplicand word in M-Register to Adder.
		ESX			Apply contents of E-Register to Adder.
			TB1, TB0		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Clock Generator.
	QTCK	$\overline{DNEZ}$ΔMOD			If multiplier in N-Register ≠ 0, set MOD to initiate multiplication.
		MWR			Initiate write operation in Memory.
				FMODΔDBCU	Advance count in TB-Counter starting with next QTCK clock signal.

FIG. 159a

TIMING DIAGRAM - INSTRCUTION 27 (VLM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL3				
	QTCK			$\overline{FTB1}\cdot\overline{FTB2}\cdot FMOD \cdot FCRE\Delta DCCU$	If CRE set, advance carry count in CC-Register by one.
		DBCUΔTB0			Advance count in TB-Counter to one.
	TL4		FMOD$\cdot\overline{FTB1}\cdot$FTB0 ΔCRE		Reset CRE when count in TB-Counter is one to prepare for storage of next carry.
	$\overline{QTCK}$	$\overline{DBCU\Delta TB1}$	$\overline{DBCU\Delta TB0}$		Advance count in TB-Counter to two.
	TL5				
	QTCK			$\overline{FMOD}\Delta$QTLP	FMOD inhibits QTLP and advance of TL-Counter.
				FTB1$\cdot\overline{FTB0}\cdot$FMOD ΔQSEX	Transfer adder output to E-Register when count in TB-Counter = 2.
				FTB1$\cdot\overline{FTB0}\cdot$FMOD ΔQNCD	Reduce multiplier count in N-Register by one when count in TB-Counter =2.
		DDC0$\cdot$QSEX ΔCRE			Set CRE if carry from character 0 of adder output.
			TB1, TB0		Reset TB-Counter to zero.
				$\overline{FTB1}\cdot\overline{FTB0}\cdot FMOD \cdot FCRE\Delta DCCU$	At end of each addition, advance count in CC-Register by one if CRE is set.

FIG. 159p

TIMING DIAGRAM - INSTRUCTION 27 (VLM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	QTCK				Count in TB-Counter advanced by one for each QTCK clock signal from zero to two and back to zero. QSEX issues to add most-significant multiplicand word to contents of E-Register each time count in E-Register = 2. QNCD also issues at this time to reduce multiplier count in N-Register by one. Above micro-operations repeated until multiplier count = 1.
			FTB1·$\overline{\text{FTB0}}$·DNE1 ΔMOD		Flip-flop MOD is reset to the 0-state when counts in N-Register and TB-Counter are one and two respectively.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	$\overline{\text{QTC0}}$	WR3, TL2	WR2, TL0		Go to W11S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W11S	TL2				
	$\overline{\text{QTCK}}$			FCRE·ΔDCCU	Advance count in CC-Register by one if CRE set to remember carry during last addition of W07 block.
	TL3			DCLQ	Clear Q-Register.
				DCM0, DCM1, DCM2, DCM3	Clear M-Register.
			CRE, DCE		
	$\overline{\text{QTCK}}$	SRI			Set SRI to enable contents of M- and E-Registers to be shifted.
			MSX		

FIG. 159q

TIMING DIAGRAM - INSTRUCTION 27 (VLM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4	DCAC△CS3			Apply carry count in CC-Register to inputs of full adders S18-S21.
	$\overline{\text{QTCK}}$			FSRI△(QSHM, QSHE)	Shift contents of M- and E-Registers right one bit position.
		FE00△M23			Shift E-Register into M-Register.
		FSRI△Q00			Count in Q-Register advanced to one.
		ESX			Apply contents of E-Register to Adder.
	TL5				
	QTCK			$\overline{\text{FSRI}}$△QTLP	FSRI inhibits QTLP and advance of TL-Counter.
		FSRI△Q01	FSRI△Q00		Count in Q-Register advanced to two.
				FSRI△(QSHM, QSHE)	Shift contents of M- and E-Registers right one bit position.
		FE00△M23			Shift E-Register into M-Register.
	QTCK				QSHM and QSHE issue and count in Q-Register advanced for each clock signal QTCK.
			DQE5△SRI		Reset SRI when count in Q-Register is five. Delay in flip-flop SRI permits one more shift, reducing count in Q-Register to zero.
			$\overline{\text{FSRI}}$△RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	$\overline{\text{QTC0}}$	TL2	WR3, TL0		Go to W03S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.

FIG. 159r

TIMING DIAGRAM - INSTRUCTION 27 (VLM) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W03S	TL2			DABX	Transfer accumulator word C address to Memory.
	QTCK	MWR			Initiate write operation in Memory.
	TL3				
	QTCK				
	TL4				
	QTCK				
	TL5				
	QTCK		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTCO	WR2, TL2	WR1, WR0, TL0		Go to W04S block.
				DACU	Advance count in Accumulator Counter Register to address accumulator word D location.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W04S	TL2			DABX	Transfer address of accumulator word D location to Memory.
	QTCK	MWR			Initiate write operation in Memory.
				QSMA, QSMB, QSMC	Transfer adder outputs to M-Register.
	TL3				
	QTCK		ESX, CAY		
	TL4				
	QTCK				
	TL5				
	QTCK	DDACAPSX			Apply address of next instruction to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for synchronizing signal from Memory.
	QTCO		WR2		Go to W00 block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.

FIG. 160a

TIMING DIAGRAM - INSTRUCTION 26 (VLD)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - (Fig. 119 except as indicated).	TL4				
	QTCK	COR			Set COR flip-flop to monitor shift.
W01 - If address development required (Fig. 121)					
W02R	TL0			$\overline{\text{FCOR}}\Delta$DDBX	Transfer address of divisor word to Memory. DDBX is present during entire W02R block.
				FCORΔDFBS	Transfer address of memory location 10 (octal) to B-Register. DFBS is present during entire W02R block.
	QTCK	MRD	FCORΔSP2		Initiate read operation in Memory.
				FCORΔQNCD	Reduce count in N-Register by one.
				FCOR·DDACΔQSCX	Transfer most-significant character of dividend to CC-Register.
	TL1			DCLQ	Clear Q-Register.
				DCM0, DCM1, DCM2, DCM3	Clear M-Register.
			DSX		
	QTCK	FCOR·FSP1 ΔSP2			Transfer partial parity from SP1 to SP2 during shift operation.
		MSX			Apply contents of M-Register to Adder.
				FCORΔQNCX	Transfer count in N-Register to CC-Register.
				$\overline{\text{FCOR}}\Delta$QCNX	Transfer contents of CC-Register to N-Register.
			RCK		Stop Program Processor Clock Generator.

FIG. 160b
TIMING DIAGRAM - INSTRUCTION 26 (VLD) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL2				Wait for signal DMDA from Memory.
	QTC2			FCOR△DCAU	Advance count in Accumulator Counter Register by one, addressing accumulator word A.
		FCOR△WR2	FCOR△WR1		If shift operation, go to W04S block; otherwise go to W02S block.
	QTRK	RCK			Start Clock Generator.
W02S	TL2			DDBX	Transfer address of divisor word to Memory. DDBX is present during entire W02S block.
	QTCK	MWR			Initiate write operation in Memory.
		$\overline{\text{FCOR}}$·DCAC ·DMRM△SIN			Set SIN if sign of divisor is minus.
	TL3				
	QTCK				
	TL4	$\overline{\text{FCOR}}$·DDAC ·DSE9·DSTZ △FVO			Set FVO if divisor is less than 100,000.
	QTCK			QSEX	Transfer divisor word to E-Register.
	TL5		$\overline{\text{FCOR}}$·FFVO△MPL		Reset MPL if Overflow flip-flop set.
	QTCK		MSX		
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for synchronizing signal from Memory.
	QTC0	WR2, WR0			Go to W07 block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.

FIG. 160c

TIMING DIAGRAM - INSTRUCTION 26 (VLD) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W03R	TL0			$\overline{FCOR}\Delta DABX$	Transfer accumulator word address to Memory. DABX is present during entire W03R block.
	QTCK	MRD			Initiate read operation in Memory.
				$\overline{FCOR} \cdot DAAC\Delta QNCX$	Transfer final quotient from N-Register to CC-Register.
	TL1	$\overline{FCOR}\Delta CS2$			Apply contents of CC-Register to Adder.
				$\overline{FCOR}\Delta(DCM0, DCM1, DCM2, DCM3)$	Clear M-Register.
	QTCK		$\overline{ESX}$		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for signal DMDA from Memory.
	QTC2	$\overline{DAAC}\Delta WR2$	$\overline{DAAC}\Delta WR0$		If accumulator word A not being addressed, go to W06S block. If accumulator word A being processed, go to W03S block.
W03S	TL2			$FCOR + \overline{FCOR}\Delta DABX$	Transfer accumulator word address to Memory. DABX is present during entire W03S block.
				$\overline{FCOR} \cdot DAAC \cdot FSIN \Delta \overline{DS55}$	If information being stored in accumulator word A location during divide operation and if sign of divisor is minus, $\overline{DS55}$ is binary 1 to insert minus sign in quotient.
	QTCK			$\overline{FCOR} \cdot DAAC\Delta QSMA$	Transfer least-significant character of adder output to M-Register.
			$\overline{FCOR} \cdot DDAC \Delta(ESX, MSX)$		
		MWR			Initiate write operation in Memory.

FIG. 160d

TIMING DIAGRAM - INSTRUCTION 26 (VLD) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL3	$\overline{FCOR \cdot FFVO}$ $\cdot$DDACΔDSX			If no overflow during divide operation, apply address of most-significant divisor word to Adder. FCAY adds one to address in Adder to form address of least-significant divisor word.
			$\overline{FCOR}\Delta$DCE		
	QTCK	FCORΔESX			Apply contents of E-Register to Adder during shift operation.
			FCORΔSP1		
				FCORΔQCNX	Transfer count in CC-Register to N-Register during shift operation.
	TL4				
	QTCK	FCOR$\cdot$MJEO ΔSP1			Set SP1 to store partial parity during shift operation.
	TL5				
	QTCK	($\overline{FCOR} \cdot$FFVO +FCOR$\cdot$DAAC) ΔPSX			Apply address in P-Register to Adder during shift operation if accumulator word A location being addressed or during divide operation if overflow.
				$\overline{FCOR} \cdot$DDACΔQSDX	Transfer address of least-significant divisor word from Adder to D-Register during divide operation.
			CAY		
			RCK		Stop Program Processor Clock Generator.

FIG. 160e
TIMING DIAGRAM - INSTRUCTION 26 (VLD) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				Wait for synchronizing signal from Memory.
	$\overline{QTC0}$	$\overline{FCOR}$ Δ(WR2, TL2)	$\overline{FCOR}$Δ($\overline{WR1}$, $\overline{WR0}$, TL0)		During shift operation, go to W04S block.
			FCOR·$\overline{FFVO}$·$\overline{DAAC}$ ΔWR0		Go to W02 block.
			FCOR(FFVO+DAAC) ΔWR1, WR0		Go to W00 block if either overflow or accumulator word A location being addressed.
	QTRK			FCORΔDACU	Advance count in Accumulator Counter Register during shift operation.
				$\overline{FCOR}$ΔDACD	Reduce count in Accumulator Counter Register during divide operation.
		RCK			Start Clock Generator on signal $\overline{DMBU}$ from Memory.
W04R	TL0			FCOR·FA0I·DAAC ΔDFBS	Enable gate B03 to address memory location 10 (octal) if A0I set and if accumulator word A location being addressed. DFBS is present during entire W04R block.
				$\overline{FCOR}$·$\overline{DAAC}$ΔDABX	Transfer address of accumulator word to Memory during divide operation, if accumulator word A location not being addressed. DABX is present during entire W04R block.
	$\overline{QTCK}$			$\overline{FCOR}$Δ$\overline{QCNX}$	Transfer count in CC-Register to N-Register.
		MRD			Initiate read operation in Memory.

FIG. 160f
TIMING DIAGRAM - INSTRUCTION 26 (VLD) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
				$\overline{FCOR}$·DDAC△DDCD	Reduce address in D-Register by one to form address of most-significant divisor word during divide operation if accumulator word D is being addressed.
			DSX		
	QTCK	MSX			Apply contents of M-Register to Adder.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	$\overline{TL2}$				Wait for synchronizing signal DMDA from Memory.
	QTC2	FCOR·$\overline{DNEZ}$ △AOI			Set AOI during shift operation.
		$\overline{FCOR}$ △(WR0, TL0)	$\overline{FCOR}$△TL2		Go to W05 block during divide operation; during shift operation, go to W04S block.
	$\overline{QTRK}$	$\overline{RCK}$			Start Clock Generator.
W04S	TL2			FCOR·FAOI·DAAC △DFBS	Enable gate B03 to address memory location 10 (octal) if AOI set and if accumulator word A location being addressed. DFBS is present during entire W04S block.
				$\overline{FCOR}$△DABX	Transfer address of accumulator word location to Memory. DABX is present during entire W04S block.
	QTCK	MWR			Initiate write operation in Memory.
				$\overline{DCAC}$△QSMA, QSMB, QSMC	Transfer most-significant corrected remainder word to M-Register.

FIG. 160g

TIMING DIAGRAM - INSTRUCTION 26 (VLD) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL3		DCE		
	QTCK		$\overline{\text{FAOI}}$·DAAC△COR ESX, CAY, MSX		Reset COR to indicate end of shift operation.
	TL4				
	QTCK				
	TL5				
	QTCK		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0			$\overline{\text{FCOR}}$·$\overline{\text{DAAC}}$△DACU	Advance count in Accumulator Counter Register during divide operation, if accumulator word A location not being addressed.
				$\overline{\text{FCOR}}$·DAAC△DACD	Reduce count in Accumulator Counter Register during divide operation, if accumulator word A location being addressed.
		FCOR△WR2	FCOR△WR1		During shift operation, go to W02R block.
		$\overline{\text{FCOR}}$△(WR1, WR0)	$\overline{\text{FCOR}}$△WR2		During divide operation, go to W03R block.
	QTRK	RCK			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.
W05	TL0			DDBX	Transfer address of divisor word to Memory. DDBX is present during entire W05 block.
	QTCK	MRD			Initiate read operation in Memory.
				QSEX	Transfer remainder word to E-Register.

FIG. 160h

TIMING DIAGRAM - INSTRUCTION 26 (VLD) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL1	DCE			Enable decimal correction circuits of Adder.
				DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	$\overline{QTCK}$	$\overline{LSN}$			Set LSN to force like-signs addition in Adder.
		FCREΔCAY			Set CAY if CRE is set.
		ESX			Apply remainder in E-Register to Adder to add to divisor word.
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	$\overline{QTRK}$	$\overline{RCK}$			Start Clock Generator.
	$\overline{QTCK}$	$\overline{MWR}$			Initiate write operation in Memory to restore divisor word.
	TL3		CRE		
	$\overline{QTCK}$				
	TL4				
	$\overline{QTCK}$				
	TL5				
	$\overline{QTCK}$		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0	$\overline{FCOR}\cdot DCAC \Delta(WR3, WR1, TL2)$	$\overline{FCOR}\cdot DCAC\Delta TL0$		If accumulator word C location being addressed, go to W15S block.
		$\overline{FCOR}\cdot\overline{DCAC} \Delta TL2$	$\overline{FCOR}\cdot\overline{DCAC} \Delta WR0, TL0$		If accumulator word C location not being addressed, go to W04S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.

FIG. 160i
TIMING DIAGRAM - INSTRUCTION 26 (VLD) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W06S	TL2				
	QTCK			QSEX	Transfer borrow count to E-Register.
	TL3	DCE			Enable decimal correction circuits of Adder.
			CS2		
	QTCK			QCER	Complement borrow count in E-Register. Add one to complement of borrow count in Adder.
		ESX			Apply complement of borrow count in E-Register to Adder.
		MSX			Apply most-significant remainder in M-Register to Adder.
	TL4		CRE		
	QTCK				
	TL5				
	QTCK		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0	WR3, WR0, TL2	TL0		Go to W15S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W07	TL0			DABX	Transfer address of dividend word to Memory.
	QTCK	MRD			Initiate read operation in Memory.
				QCER	Form complement of divisor word in E-Register.

FIG. 160j
TIMING DIAGRAM - INSTRUCTION 26 (VLD) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL1	DCE			Enable decimal correction circuits of Adder.
		$\overline{\text{FFVO}}\Delta\text{MPL}$			Set MPL if no overflow.
		FN00ΔQ00 FN01ΔQ01 FN02ΔQ02 FN03ΔQ03	$\overline{\text{FN00}}\Delta\text{Q00}$ $\overline{\text{FN01}}\Delta\text{Q01}$ $\overline{\text{FN02}}\Delta\text{Q02}$ $\overline{\text{FN03}}\Delta\text{Q03}$		Store most-significant dividend character in Q-Register.
				DCM0, DCM1, DCM2, DCM3	Clear M-Register.
				DCLC	Clear CC-Register.
			CS2		
	QTCK	MSX			Apply dividend word to Adder.
		ESX			Apply complement of divisor word to Adder.
		$\overline{\text{FFVO}}\Delta\text{CAY}$			Add one to complement of divisor in Adder.
			TL1, TB0		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
				DCAC·FMOD·DMRM $\Delta\overline{\text{DS55}}$	$\overline{\text{DS55}}$ issues to insert minus sign in quotient if sign of dividend is minus.
	QTRK	RCK			Start Clock Generator.
	QTCK	$\overline{\text{FFVO}}$·FMPL ΔMOD			Set MOD to initiate division if no overflow and MPL set.
			LSN		
				DBCU	Advance count in TB-Counter starting with next QTCK clock signal.

FIG. 160k
TIMING DIAGRAM - INSTRUCTION 26 (VLD) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL3				
	QTCK	DCAC·DMRM ·FSIN△SIN	DCAC·DMRM·FSIN △SIN		Set SIN if signs of dividend word divisor are minus and plus respectively. State of SIN significant only when final quotient character is being developed.
		DBCU△TB0			Advance count in TB-Counter to one.
	TL4		$\overline{FTB1}\cdot FTB0\cdot FMOD$ △MPL		Reset MPL when count in TB-Counter = 1.
			$\overline{FTB1}\cdot FTB0\cdot FMOD$ △CRE		Reset CRE when count in TB-Counter = 1 to prepare for storage of next carry.
	QTCK	DBCU△TB1	DBCU△TB0		Advance count in TB-Counter to two.
	TL5				
	QTCK			$\overline{FMOD}$△QTLP	FMOD inhibits QTLP and advance of TL-Counter.
				$FTB1\cdot\overline{FTB0}\cdot FMOD\cdot(\overline{DNEZ}+DDAC\cdot DDC0)$ △(QSMA, QSMB, QSMC)	Transfer remainder to M-Register.
		DDC0·QSMC △CRE			Set CRE if carry from character 0 of adder output.
			TB1, TB0		Reset TB-Counter to zero.
				$FMOD\cdot\overline{FMPL}\cdot\overline{FTB1}\cdot\overline{FTB0}(DCAC\cdot\overline{DNEZ}\cdot FCRE+DNEZ\cdot DDAC+DDAC\cdot FCRE)$△DCCU	At end of each subtraction, advance count in CC-Register by one to form trial quotient or to count borrows to be subtracted from remainder.
				$FMOD\cdot\overline{FMPL}\cdot\overline{FTB1}\cdot FTB0(DCAC+FCRE)$ △QNCD	Reduce count in N-Register by one when count in TB-Counter = 0 for each borrow when processing most-significant dividend word or for each subtraction when processing least-significant dividend word.

FIG. 160L

TIMING DIAGRAM - INSTRUCTION 26 (VLD) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	QTCK	DBCU△TB0			Count in TB-Counter advanced by one for each QTCK clock signal from zero to two and back to zero. QSMA, QSMB and QSMC issue to subtract divisor from dividend each time count in TB-Counter = 2. When most-significant divisor word is being subtracted from most-significant dividend word, count in CC-Counter advanced by one for each subtraction to form trial quotient. When least-significant divisor word is being subtracted from least-significant dividend word, count in CC-Counter advanced by one for each borrow. Borrow count is later subtracted from most-significant remainder word.
			[$\overline{FTB1}$·FTB0·DDAC ·DCE1·FMOD +$\overline{FMPL}$·FTB1 ·$\overline{FTB0}$(DCAC ·DNE1+DCAC ·DNEZ+DDAC ·DNEZ·$\overline{FCRE}$)] △MOD		Reset MOD if error condition (DCE1) or if division completed.
			RCK		Stop Program Processor Clock Generator.

FIG. 160m
TIMING DIAGRAM - INSTRUCTION 26 (VLD) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				Wait for memory synchronizing signal.
	$\overline{QTC0}$	DCAC△TL2	DCAC△WR1, WR0, TL0		If addressing accumulator word C, go to W04S block.
		$\overline{DCAC}$△TL2	$\overline{DCAC}$△WR2, TL0		If addressing accumulator word D, go to W03S block.
		DSGP·DCE1 △FVO			Set FVO if trial quotient = 11.
	QTRK	RCK			Start Clock Generator on signal $\overline{DMBU}$ from Memory.
W11S	TL2			FCOR·$\overline{FMPL}$·DAAC ·FLSN△DRE4	Force adder output signals $\overline{DS55}$ and $\overline{DS54}$ to binary 0 during shift operation if accumulator word A location being addressed and dividend and divisor have like-signs.
	$\overline{QTCK}$			$\overline{QSEX}$	Transfer accumulator word to E-Register.
	TL3			DCLQ	Clear Q-Register.
				DCM0, DCM1, DCM2, DCM3	Clear M-Register.
			CRE, DCE		
	QTCK	$\overline{DNEZ}$△SRI			If count in N-Register ≠ 0, set SRI to enable contents of M- and E-Registers to be shifted.
			MSX, SP2		
			$\overline{FMPL}$·DAAC△LSN		Reset LSN if accumulator word A location being addressed.

FIG. 160n

TIMING DIAGRAM - INSTRUCTION 26 (VLD) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4				
	$\overline{QTCK}$			$\overline{FSRI}\Delta(\overline{QSHM}, \overline{QSHE})$	Shift accumulator word in E-Register right one bit position.
		FSP1ΔSP2			Transfer partial parity from SP2 to SP1.
		FE00ΔM23			Shift E-Register into M-Register.
		FSRIΔQ00			Count in Q-Register advanced to one.
	TL5				
	QTCK			$\overline{FSRI}\Delta QTLP$	FSRI inhibits QTLP and advance of TL-Counter.
		FSRIΔQ01	FSRIΔQ00		Count in Q-Register advanced to two.
				FSRIΔ(QSHM, QSHE)	Shift accumulator word in E-Register right one bit position.
		FE00ΔM23			Shift E-Register into M-Register.
	$\overline{QTCK}$				$\overline{QSHM}$ and $\overline{QSHE}$ issue and count in Q-Register advanced by one for each clock signal QTCK.
				FSRI·DQE5ΔQNCD	Count in N-Register reduced by one for each six bit position shifts.
			DNE1·DQE5ΔSRI		Reset SRI when counts in N- and Q-Registers are one and five respectively. Delay in flip-flop SRI permits one more shift, reducing count in N- and Q-Registers to zero.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	$\overline{QTC0}$	TL2	WR3, $\overline{TL0}$		Go to W03S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.

FIG. 160o

TIMING DIAGRAM - INSTRUCTION 26 (VLD) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W15R	TL0				
	$\overline{QTCK}$				
	TL1	CC1, CC0			Preset CC-Register to count of three. Right shift through three character positions is equivalent to left shift through one character position.
			FMPLΔFV0		Reset FV0 during first execution of instruction 26.
				DDCD	Reduce count in D-Register by one to address most-significant divisor word.
	$\overline{QTCK}$		$\overline{ESX}$		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	$\overline{QTC2}$			FC0RΔDCAD	Reduce count in Accumulator Counter Register by one to address accumulator word D.
		TL0	WR3, WR1, WR0, TL2		Go to W04R block.
	$\overline{QTRK}$	RCK			Start Program Processor Clock Generator.
W15S	TL2			DABX	Transfer address of accumulator word to Memory. DABX is present during entire W15S block.
				DCAC·DERM$\Delta\overline{DS55}$	$\overline{DS55}$ is binary 1 to insert proper sign in remainder if sign of dividend is minus.

FIG. 160p

TIMING DIAGRAM - INSTRUCTION 26 (VLD) (Cont.)

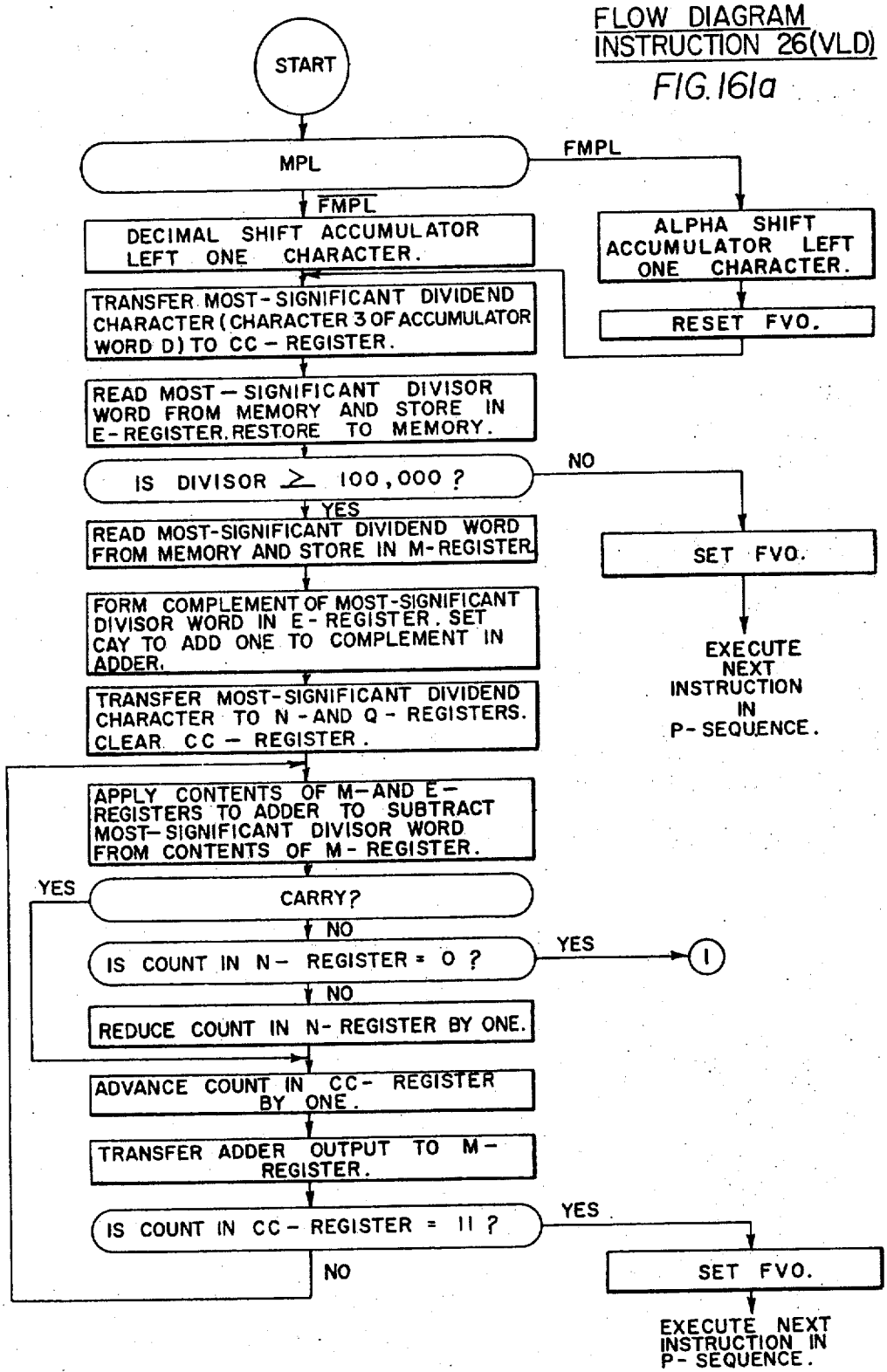
BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	QTCK	MWR			Initiate write operation in Memory to restore part of corrected remainder to Memory.
				QSMA, QSMB, QSMC	Transfer corrected remainder to M-Register.
		DDC0·QSMC ΔCRE			Set CRE if carry out of bit position 21 of Adder.
				DDACΔQQNX	Transfer trial quotient from Q-Register to N-Register.
	TL3		DCE		
	QTCK			DDAC·$\overline{\text{FCRE}}$ΔQNCD	Reduce count in N-Register by one to correct trial quotient.
			ESX, MSX, PSX, CAY		
	TL4				
	$\overline{\text{QTCK}}$				
	TL5	$\overline{\text{FN00}}$·FN01 ·$\overline{\text{FN02}}$·FN03 ΔFVO			Set FVO is trial quotient = 10.
	$\overline{\text{QTCK}}$		$\overline{\text{RCK}}$		Stop Program Processor Clock Generator.

FIG. 160q
TIMING DIAGRAM - INSTRUCTION 26 (VLD) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				Wait for synchronizing signal from Memory.
	QTC0			(DDAC+FCRE)ΔDACU	Advance count in Accumulator Counter Register by one to address most-significant half of remainder for correction (DDAC) or to address accumulator word A location to store quotient (FCRE).
				DDAC·$\overline{\text{FCRE}}$ΔDACD	Reduce count in Accumulator Counter Register by one to address least-significant half of remainder for correction.
			DDAC·FCREΔ(WR3, WR2)		Go to W03 block, if correction of remainder not required.
			$\overline{\text{DDAC}}$+$\overline{\text{FCRE}}$Δ(WR3, WR1, WR0)		Go to W04 block, if remainder correction required.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.

FLOW DIAGRAM
INSTRUCTION 26(VLD)

FIG. 161a



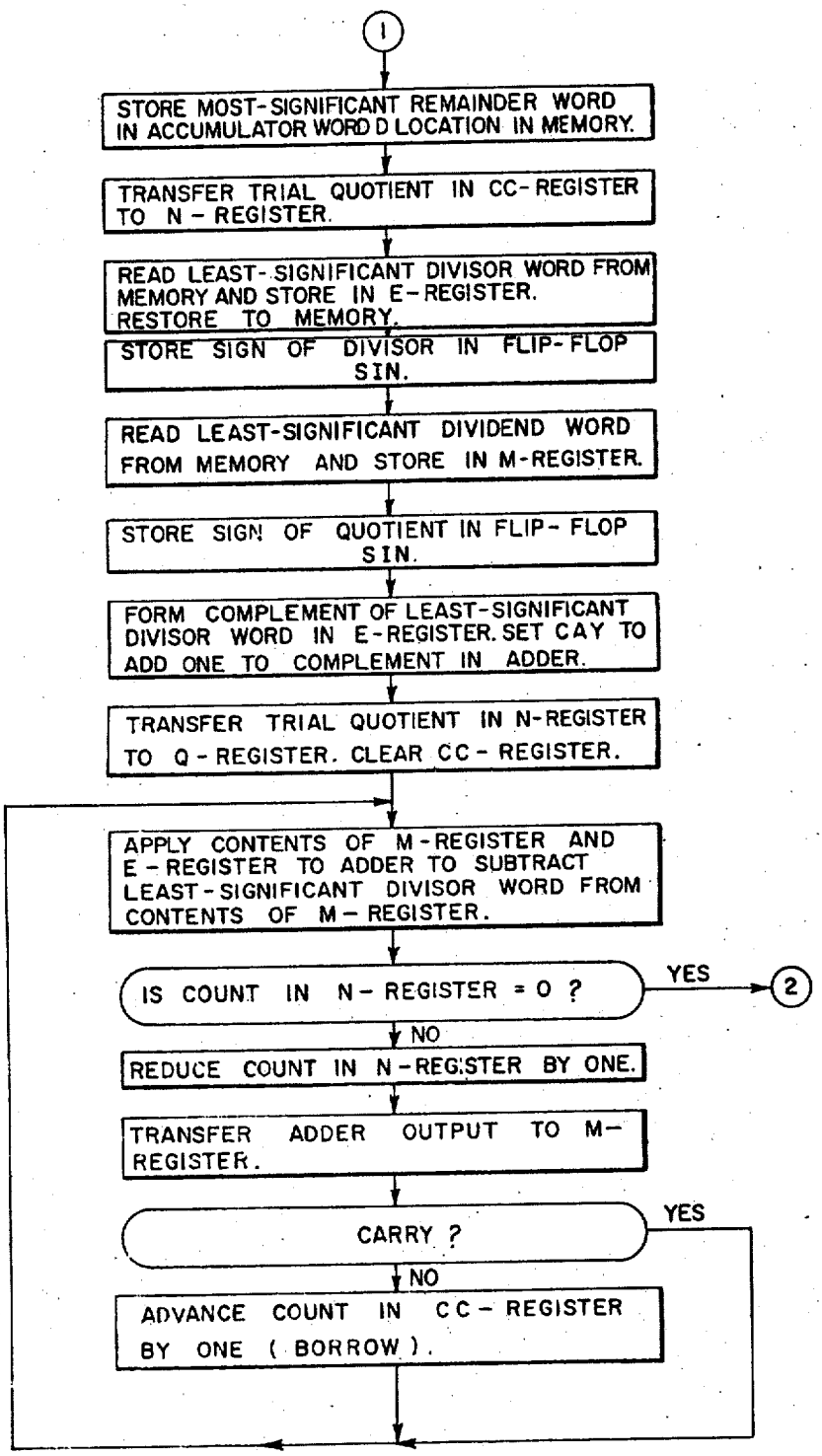

1
STORE MOST-SIGNIFICANT REMAINDER WORD IN ACCUMULATOR WORD D LOCATION IN MEMORY.
TRANSFER TRIAL QUOTIENT IN CC-REGISTER TO N-REGISTER.
READ LEAST-SIGNIFICANT DIVISOR WORD FROM MEMORY AND STORE IN E-REGISTER. RESTORE TO MEMORY.
STORE SIGN OF DIVISOR IN FLIP-FLOP SIN.
READ LEAST-SIGNIFICANT DIVIDEND WORD FROM MEMORY AND STORE IN M-REGISTER.
STORE SIGN OF QUOTIENT IN FLIP-FLOP SIN.
FORM COMPLEMENT OF LEAST-SIGNIFICANT DIVISOR WORD IN E-REGISTER. SET CAY TO ADD ONE TO COMPLEMENT IN ADDER.
TRANSFER TRIAL QUOTIENT IN N-REGISTER TO Q-REGISTER. CLEAR CC-REGISTER.
APPLY CONTENTS OF M-REGISTER AND E-REGISTER TO ADDER TO SUBTRACT LEAST-SIGNIFICANT DIVISOR WORD FROM CONTENTS OF M-REGISTER.
IS COUNT IN N-REGISTER = 0 ?
YES
2
NO
REDUCE COUNT IN N-REGISTER BY ONE.
TRANSFER ADDER OUTPUT TO M-REGISTER.
CARRY ?
YES
NO
ADVANCE COUNT IN CC-REGISTER BY ONE (BORROW).

FIG. 161b

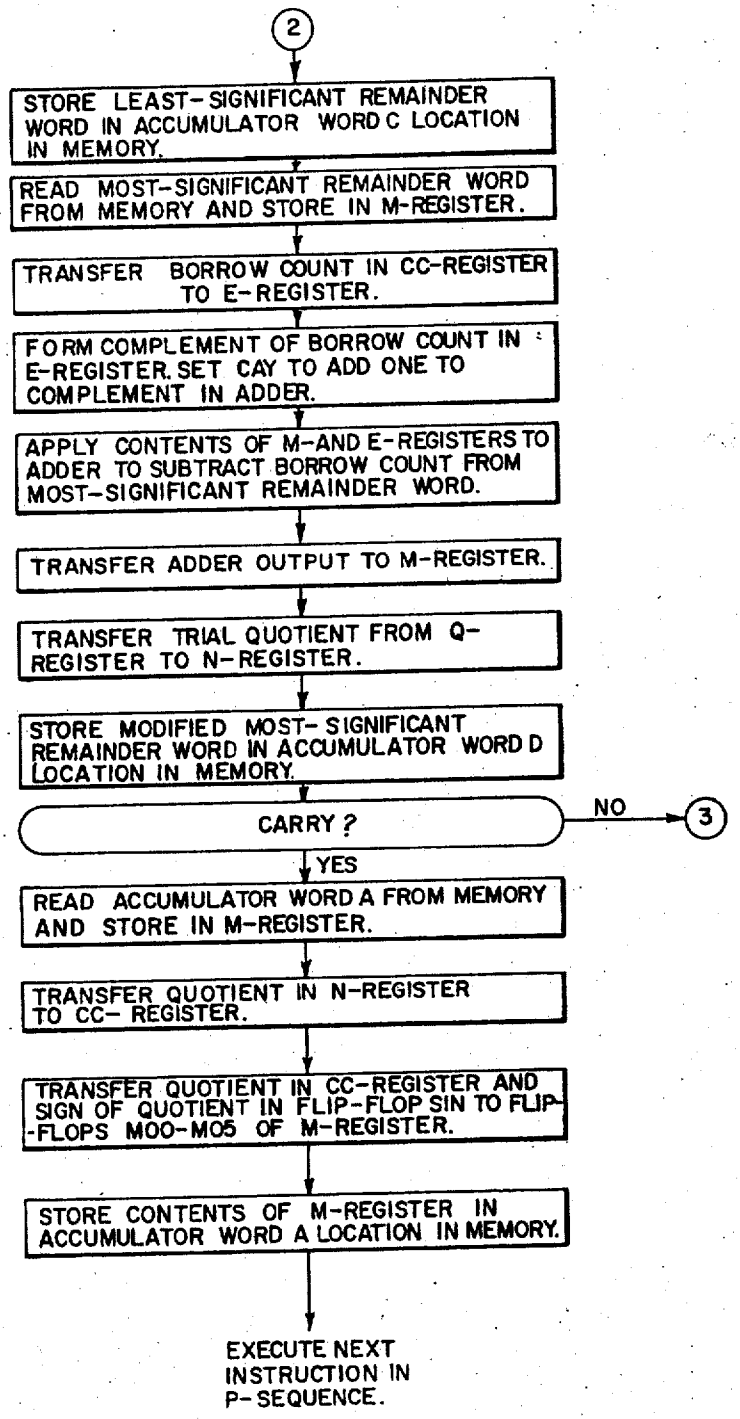

2
STORE LEAST-SIGNIFICANT REMAINDER WORD IN ACCUMULATOR WORD C LOCATION IN MEMORY.
READ MOST-SIGNIFICANT REMAINDER WORD FROM MEMORY AND STORE IN M-REGISTER.
TRANSFER BORROW COUNT IN CC-REGISTER TO E-REGISTER.
FORM COMPLEMENT OF BORROW COUNT IN E-REGISTER. SET CAY TO ADD ONE TO COMPLEMENT IN ADDER.
APPLY CONTENTS OF M-AND E-REGISTERS TO ADDER TO SUBTRACT BORROW COUNT FROM MOST-SIGNIFICANT REMAINDER WORD.
TRANSFER ADDER OUTPUT TO M-REGISTER.
TRANSFER TRIAL QUOTIENT FROM Q-REGISTER TO N-REGISTER.
STORE MODIFIED MOST-SIGNIFICANT REMAINDER WORD IN ACCUMULATOR WORD D LOCATION IN MEMORY.
CARRY ?
NO
3
YES
READ ACCUMULATOR WORD A FROM MEMORY AND STORE IN M-REGISTER.
TRANSFER QUOTIENT IN N-REGISTER TO CC- REGISTER.
TRANSFER QUOTIENT IN CC-REGISTER AND SIGN OF QUOTIENT IN FLIP-FLOP SIN TO FLIP-FLOPS M00-M05 OF M-REGISTER.
STORE CONTENTS OF M-REGISTER IN ACCUMULATOR WORD A LOCATION IN MEMORY.
EXECUTE NEXT INSTRUCTION IN P-SEQUENCE.

FIG. 161c

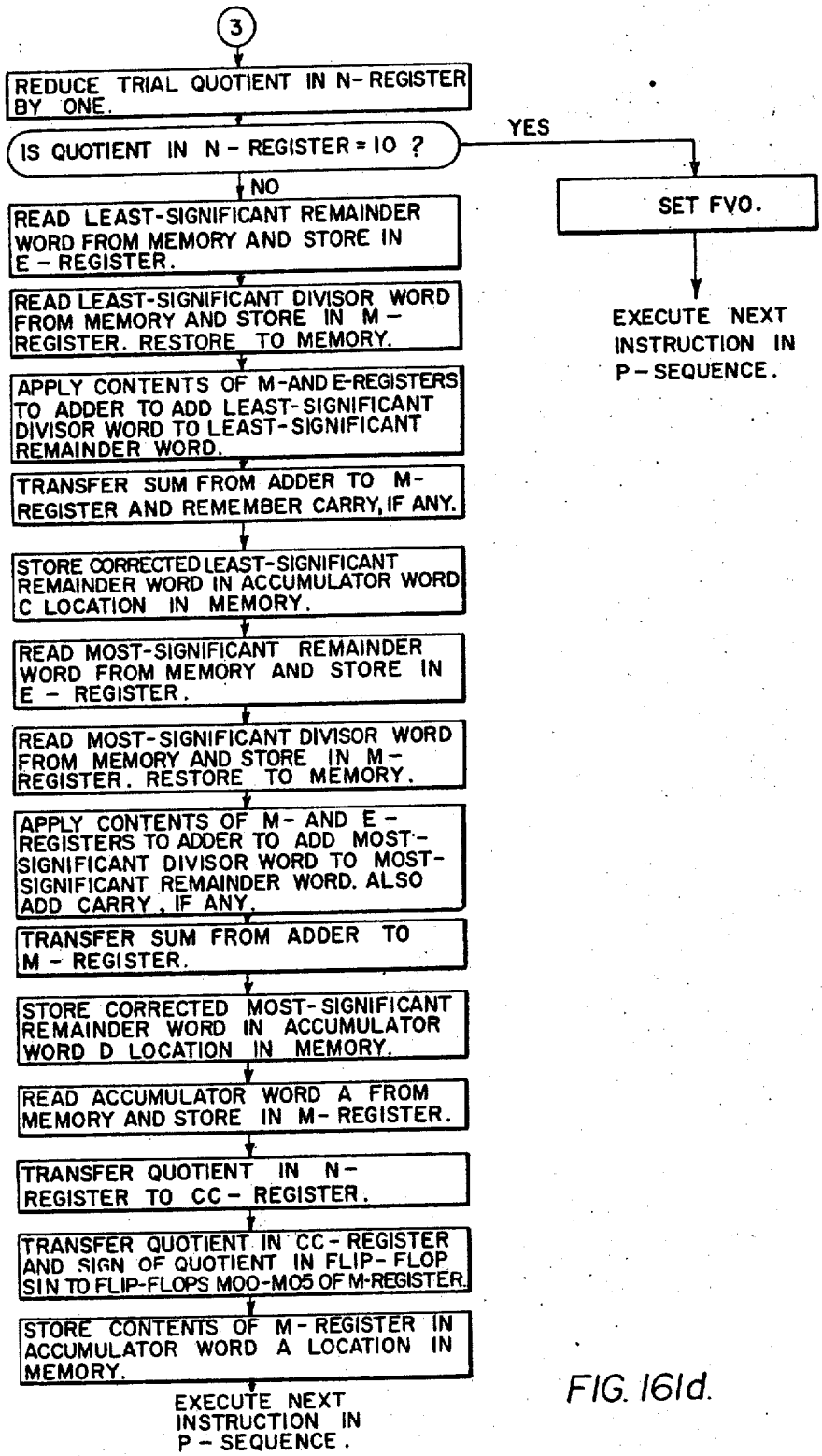

3
REDUCE TRIAL QUOTIENT IN N-REGISTER BY ONE.
IS QUOTIENT IN N-REGISTER = 10 ?
YES
SET FVO.
EXECUTE NEXT INSTRUCTION IN P-SEQUENCE.
NO
READ LEAST-SIGNIFICANT REMAINDER WORD FROM MEMORY AND STORE IN E-REGISTER.
READ LEAST-SIGNIFICANT DIVISOR WORD FROM MEMORY AND STORE IN M-REGISTER. RESTORE TO MEMORY.
APPLY CONTENTS OF M-AND E-REGISTERS TO ADDER TO ADD LEAST-SIGNIFICANT DIVISOR WORD TO LEAST-SIGNIFICANT REMAINDER WORD.
TRANSFER SUM FROM ADDER TO M-REGISTER AND REMEMBER CARRY, IF ANY.
STORE CORRECTED LEAST-SIGNIFICANT REMAINDER WORD IN ACCUMULATOR WORD C LOCATION IN MEMORY.
READ MOST-SIGNIFICANT REMAINDER WORD FROM MEMORY AND STORE IN E-REGISTER.
READ MOST-SIGNIFICANT DIVISOR WORD FROM MEMORY AND STORE IN M-REGISTER. RESTORE TO MEMORY.
APPLY CONTENTS OF M- AND E-REGISTERS TO ADDER TO ADD MOST-SIGNIFICANT DIVISOR WORD TO MOST-SIGNIFICANT REMAINDER WORD. ALSO ADD CARRY, IF ANY.
TRANSFER SUM FROM ADDER TO M-REGISTER.
STORE CORRECTED MOST-SIGNIFICANT REMAINDER WORD IN ACCUMULATOR WORD D LOCATION IN MEMORY.
READ ACCUMULATOR WORD A FROM MEMORY AND STORE IN M-REGISTER.
TRANSFER QUOTIENT IN N-REGISTER TO CC-REGISTER.
TRANSFER QUOTIENT IN CC-REGISTER AND SIGN OF QUOTIENT IN FLIP-FLOP SIN TO FLIP-FLOPS M00-M05 OF M-REGISTER.
STORE CONTENTS OF M-REGISTER IN ACCUMULATOR WORD A LOCATION IN MEMORY.
EXECUTE NEXT INSTRUCTION IN P-SEQUENCE.

FIG. 161d.

FIG. 162
EDIT MODE

CONDITIONS								EFFECTS				
Type of Format	Format Code		Is First Data Character Still Being Considered?	Sign of Data	Suppression Established Previously?	D = 0	Tentative Result	Advance to Next Character?		Effect on Suppression Counter	Other Actions	Note
	Char.	Octal						Format	Data			
Simple Suppress	~	12				No Yes	D D	Yes Yes	Yes Yes	Clear to 0 Add 1	Set simple suppress Set simple suppress	III III
Float $	#	13				No Yes	D D	Yes Yes	Yes Yes	Clear to 0 Add 1	Set float $ Set float $	III III
Asterisk Protect	>	16				No Yes	D D	Yes Yes	Yes Yes	Clear to 0 Add 1	Set * protect Set * protect	III III
Ignore	I	17			No Yes		F F	Yes Yes	Yes Yes	Add 1		II II
Sign	—	52	No No Yes Yes	Plus Minus	No Yes		F F Space F	Yes Yes Yes Yes	No No No No	Add 1	Set zone bits of D to 00 in processor Set zone bits of D to 00 in processor	I I IV I
Use	;	56			No Yes Yes	No Yes	D D D	Yes Yes Yes	Yes Yes Yes	Clear to 0 Add 1		III III III
Insert	Others	Others			No Yes		F F	Yes Yes	No No	Add 1		I I

F = Format character under consideration D = Data character under consideration

NOTE:
I. The format character is retained in the tentative result and the data character is retained to be processed by the next format character.
II. The format character is retained in the tentative result and the data character is discarded, so that the next format character will process the next data character.
III. The format character is discarded and the data character is placed in the tentative result so that the next format character processes the next data character.
IV. A blank space is placed in the tentative result. The format character is discarded and the data character is retained to be processed by the next format character.

FIG. 163a

TIMING DIAGRAM - INSTRUCTION 05 (EDT)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					
W01 - If address development required (Fig. 121)					
W02	TL0			DDBX	Transfer address of data word to Memory.
	QTCK	MRD			Initiate read operation in Memory.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
	QTCK	MSX			Apply data word to Adder.
			ESX		
			RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Clock Generator.
	QTCK	MWR			Initiate write operation in Memory to restore data word.
	TL3				
	QTCK				
	TL4				
	QTCK			QSEX	Transfer data word to E-Register.
	TL5				
	QTCK		MSX		
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0	WR3, WR2			Go to W14 block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.

FIG. 163b

TIMING DIAGRAM - INSTRUCTION 05 (EDT) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W14	TL0			DABX	Transfer address of accumulator (format) word to Memory.
	$\overline{QTCK}$	MRD			Initiate read operation in Memory.
	TL1			DCM0, DCM1, DCM2, DCM3	Clear M-Register.
				DSGO△DNS4	If working length of Accumulator is single, preset count of four in N-Register.
				DDBO·DAAC△DNS8	If working length of Accumulator is double and if addressing accumulator word A, preset count of eight in N-Register.
				DTPO·DAAC△DNS2	If working length of Accumulator is triple and if addressing accumulator word A, preset count of twelve in N-Register.
				DQDO·DAAC△DNS6	If working length of Accumulator is quadruple and if addressing accumulator word D, preset count of sixteen in N-Register.
				DCLQ	Clear Q-Register.
	$\overline{QTCK}$		$\overline{RCK}$		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
	QTRK	RCK			Start Clock Generator.

FIG. 163c

TIMING DIAGRAM - INSTRUCTION 05 (EDT) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	QTCK	$\overline{DVRA}\Delta CL1$ $DVRB\Delta CL2$	$DVRA\Delta CL1$ $\overline{DVRB}\Delta CL2$		Preset class flip-flops according to format character in flip-flops M05-M00 of M-Register as follows: Octal / Class / FCL2 / FCL1 12, 13, 16, 56 / Use / 0 / 0 52 / Sign / 1 / 1 17 / Ignore / 1 / 0 All others / Insert / 0 / 1
		$\overline{FCOR}\cdot\overline{FM04}\cdot FM03\cdot FM01\cdot(FM02\cdot\overline{FM00}+FM05\cdot\overline{FM02})\Delta ESX$			Apply data character in E-Register to Adder if format character in flip-flops M05-M00 of M-Register is octal 12, 13, 16 or 56.
		$(DVRC+DVRE)\Delta SP1$ $(DVRC+DVRD)\Delta SP2$	$DVRD\Delta SP1$ $(DVRE+DGE4)\Delta SP2$		If character in flip-flops M05-M00 of M-Register is octal 56, flip-flops SP1 and SP2 remain in 0-state; otherwise set to suppress code as follows: Octal / FSP2 / FSP1 12 / 0 / 1 13 / 1 / 1 16 / 1 / 0 If suppress mode, type of suppression is suppress and float $, count in CC-Register is equal to accumulator working length and character in flip-flops M05-M00 of M-Register is a zero, comma or period, reset flip-flop SP2 to change to single suppress.
		$FCOR\Delta SRI$			Initiate shift operation during suppress mode.

FIG. 163d
TIMING DIAGRAM - INSTRUCTION 05 (EDT) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	QTCK (Cont.)			FCOR·DZCP·DCEL ΔQSMA	Transfer least-significant character of adder output (space if $\overline{FSP2}$·FSP1, * if FSP2·FSP1 or $ if FSP2·FSP1) to M-Register if suppress mode, count in CC-Register is equal to accumulator working length and character in flip-flops M05-M00 of M-Register is a zero, comma or period.
	TL3			$\overline{FCL1}$·$\overline{FCL2}$·$\overline{FCOR}$ ·$\overline{DEEZ}$·(FSP1+FSP2) ΔDCLC	Clear CC-Register during edit mode if format character is of suppress type and character 3 of E-Register ≠ 0.
	QTCK	FSRIΔQ00			Count in Q-Register advanced to one (suppress mode).
				$\overline{FCOR}$·DVFB(FCL2 +FCL1+DEEZ)ΔDCCU	If edit mode and suppression format character previously used and if present format character is ignore, insert or sign or if data character is zero, advance count in CC-Register by one.
				FSRIΔQSHM	Shift format word in M-Register right one bit position, if suppress mode.
		QSHM·FM00 ΔM23	QSHM·$\overline{FM00}$ΔM23		Circular shift M-Register.

FIG. 163e

TIMING DIAGRAM - INSTRUCTION 05 (EDT) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL4				
	$\overline{QTCK}$	FSRIΔQ01	FSRIΔQ00		Count in Q-Register advanced to two during suppress mode.
		$\overline{FCOR}$ΔSRI			Initiate shift operation during edit mode.
			ESX		
				$(\overline{FCOR}\cdot\overline{FCL1}\cdot\overline{FCL2}$ $+FCOR\cdot FRNZ\cdot DERM$ $\cdot FCL1\cdot FCL2)\Delta QSMA$	During edit mode, transfer data character from flip-flops E05-E00 of the E-Register to flip-flops M05-M00 of the M-Register if the format character is of the use class. If the format character is of the sign class, the data character is the first and the sign of the data word in the E-Register is plus, transfer a space to flip-flops M05-M00 of the M-Register; if the data character is the first and the sign of the data word in the E-Register is minus or if the data character is not the first, the format character (−) is used and no transfer is necessary.
				FSRIΔQSHM	Shift format word in M-Register right one bit position.
		QSHM·FM00 ΔM23	QSHM·$\overline{FM00}$ΔM23		Circular shift M-Register.

FIG. 163f

TIMING DIAGRAM - INSTRUCTION 05 (EDT) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5			$\overline{FCOR}\cdot\overline{FRNZ}\cdot FCL1\cdot FCL2\Delta DRE5$	During edit mode and sign class, if first data character, reset flip-flops E04 and E05.
	QTCK			$\overline{FSRI}\Delta QTLP$	FSRI inhibits QTLP and advance of TL-Counter.
		FSRIΔQ00			Count in Q-Register advanced to one (edit mode) or three (suppress mode).
				FSRIΔQSHM	Shift format word in M-Register right one bit position.
				FSRI$\cdot\overline{FCL1}\Delta$QSHE	Shift data word in E-Register right one bit position if use or ignore class.
		QSHM$\cdot$FM00 ΔM23	QSHM$\cdot\overline{FM00}\Delta$M23		Circular shift M-Register.
	$\overline{QTCK}$				$\overline{QSHM}$ and $\overline{QSHE}$ continue to issue as described above and count in Q-Register advanced by one for each clock signal QTCK.
				FSRI$\cdot$DQE5ΔQNCD	Count in N-Register reduced by one for each shift through six bit positions (format character count).
				$\overline{FCOR}\cdot$DQE5$\cdot\overline{FCL1}\cdot$FSRI$\Delta$DBCU	Count in TB-Counter advanced by one for each shift of the data word through six bit positions (data character count), during edit mode.
				FCOR$\cdot$DQE5$\cdot\overline{DCEL}\cdot$FSRI$\Delta$DCCU	Advance count in CC-Register by one for each shift through six bit positions during suppress mode if character count in CC-Register is not equal to accumulator working length.

FIG. 163g

TIMING DIAGRAM - INSTRUCTION 05 (EDT) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	QTCK (Cont.)	$\overline{FCOR}\cdot DQE5 \cdot \overline{FCL1}\cdot FSRI \Delta RNZ$			Set RNZ after processing least-significant data character; during subsequent sign class, insert (−) and retain data character for consideration by next format character.
			DQE5(DCEL+$\overline{FCOR}$) ΔSRI		Reset SRI when count in Q-Register = 5.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0	$\overline{DFDE}\Delta(\overline{WR0}, TL2)$	$\overline{DFDE}\Delta(\overline{WR3}, \overline{WR2}, TL0)$		If editing of data word in E-Register completed, go to W03S block.
		(DNE4+DNZE) ΔWR0, TL2	(DNE4+DNZE)ΔTL0		If count in N-Register = 0, 4, 8 or 12 (end of format word in M-Register), go to W15S block.
		$\overline{DFDE}\cdot\overline{DNE4}$ $\cdot\overline{DNZE}\Delta TL2$	$\overline{DFDE}\cdot\overline{DNE4}\cdot\overline{DNZE}$ ΔTL0		If none of above conditions exist, go to W14S block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.
W03S	TL2			DABX	Transfer address of accumulator word to Memory.
	QTCK	MWR			Initiate write operation to write result into addressed accumulator word location.
	TL3				
	QTCK				
	TL4				
	QTCK				

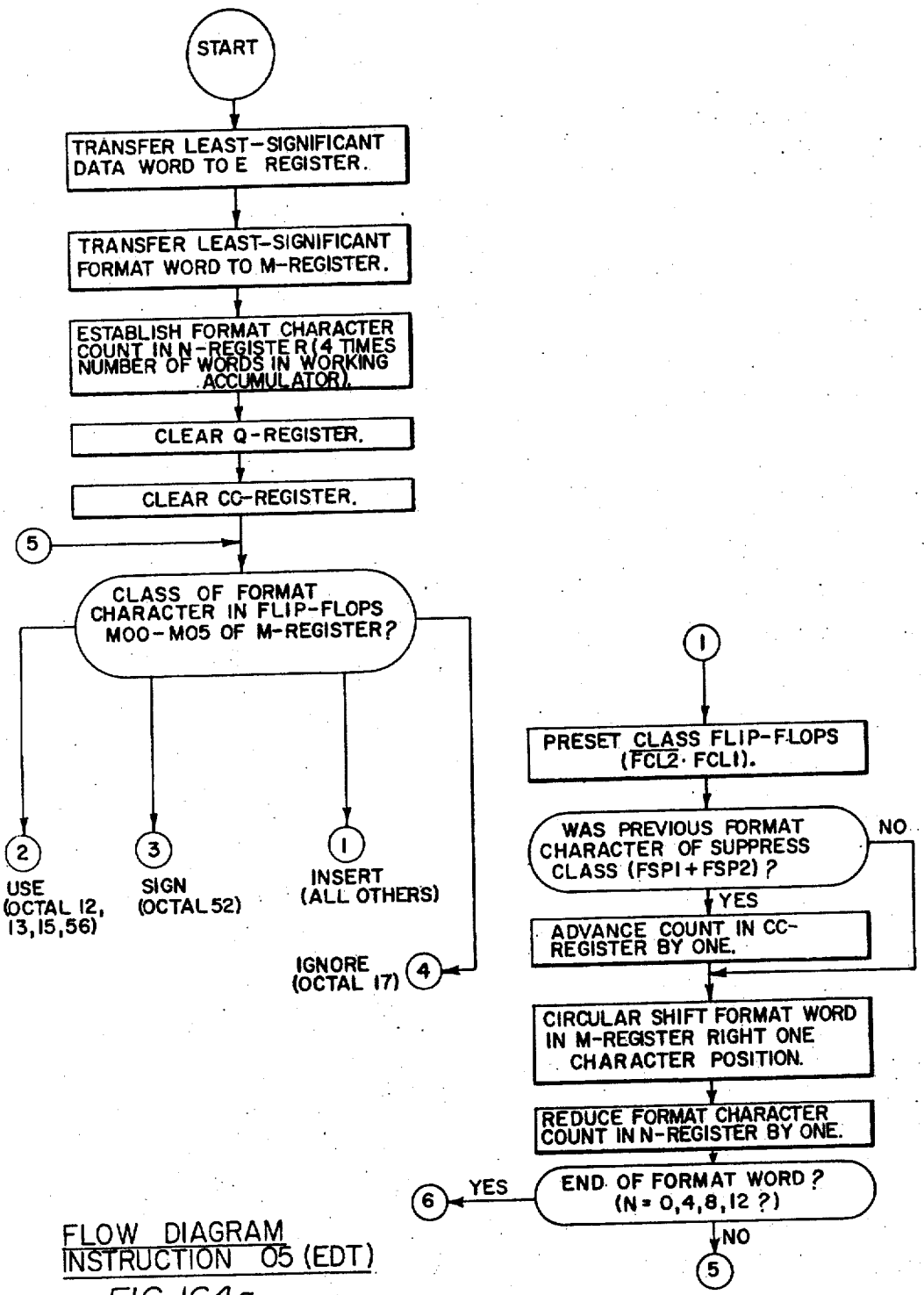
FIG. 163h

TIMING DIAGRAM - INSTRUCTION 05 (EDT) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5			DDC0	Reduce count in D-Register (data word address) by one.
	$\overline{QTCK}$		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for synchronizing signal DMDA from Memory.
	$\overline{QTC0}$		WR0		Go to W02 block.
	QTRK	RCK			Start Clock Generator.
W15S	TL2			DABX	Transfer accumulator (format) word address to Memory.
	$\overline{QTCK}$	MWR			Initiate write operation to write result into addressed accumulator word location.
	TL3				
	QTCK	DNE2·$\overline{DCCZ}$ ·FCOR△COR			If count in CC-Register is not zero and last accumulator word has been read, set COR to establish suppress mode.
			DNE2·FCOR△COR		Reset COR after suppress mode completed.
			ESX, MSX, PSX, CAY		
	TL4				
	$\overline{QTCK}$				
	TL5				
	QTCK	$\overline{FCOR}$△PSX			If no suppress mode or if suppress mode completed, apply address of next instruction to Adder.
			RCK		Stop Program Processor Clock Generator.

FIG. 163i
TIMING DIAGRAN - INSTRUCTION 05 (EDT) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL0				Wait for memory synchronizing signal.
	QTC0	(FCOR+$\overline{DLA0}$) ΔTL2	(FCOR+$\overline{DLA0}$) Δ(WR0, TL2) $\overline{FCOR}$·DLA0Δ(WR3, WR2, WR1, WR0)		If suppress mode or if edit mode not completed, go to W14S block. If no suppress mode or end of suppress mode, go to W00 block.
	QTRK	RCK		(DNZE+DNE4)ΔDACU	Form address of next higher-order accumulator word if count in N-Register = 0, 4, 8 or 12. Start Clock Generator on signal DMBU from Memory.



FLOW DIAGRAM
INSTRUCTION 05 (EDT)

FIG. 164a

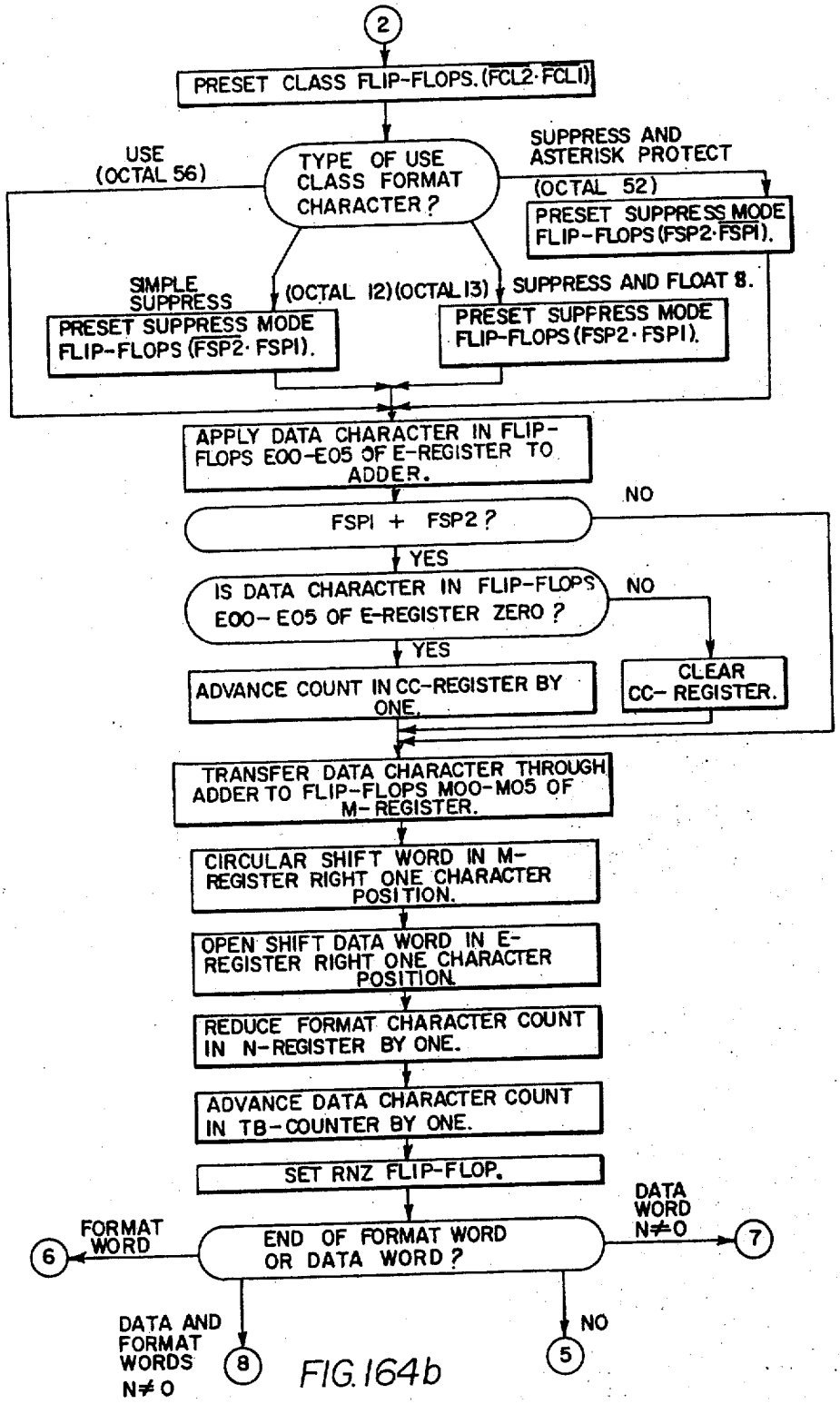

2
PRESET CLASS FLIP-FLOPS. (FCL2·FCL1)
USE (OCTAL 56)
TYPE OF USE CLASS FORMAT CHARACTER?
SUPPRESS AND ASTERISK PROTECT (OCTAL 52)
PRESET SUPPRESS MODE FLIP-FLOPS (FSP2·FSP1).
SIMPLE SUPPRESS (OCTAL 12)
(OCTAL 13) SUPPRESS AND FLOAT $.
PRESET SUPPRESS MODE FLIP-FLOPS (FSP2·FSP1).
PRESET SUPPRESS MODE FLIP-FLOPS (FSP2·FSP1).
APPLY DATA CHARACTER IN FLIP-FLOPS E00-E05 OF E-REGISTER TO ADDER.
FSP1 + FSP2?
NO
YES
IS DATA CHARACTER IN FLIP-FLOPS E00-E05 OF E-REGISTER ZERO?
NO
YES
ADVANCE COUNT IN CC-REGISTER BY ONE.
CLEAR CC-REGISTER.
TRANSFER DATA CHARACTER THROUGH ADDER TO FLIP-FLOPS M00-M05 OF M-REGISTER.
CIRCULAR SHIFT WORD IN M-REGISTER RIGHT ONE CHARACTER POSITION.
OPEN SHIFT DATA WORD IN E-REGISTER RIGHT ONE CHARACTER POSITION.
REDUCE FORMAT CHARACTER COUNT IN N-REGISTER BY ONE.
ADVANCE DATA CHARACTER COUNT IN TB-COUNTER BY ONE.
SET RNZ FLIP-FLOP.
DATA WORD N≠0
7
FORMAT WORD
6
END OF FORMAT WORD OR DATA WORD?
DATA AND FORMAT WORDS N≠0
8
NO
5

FIG. 164b

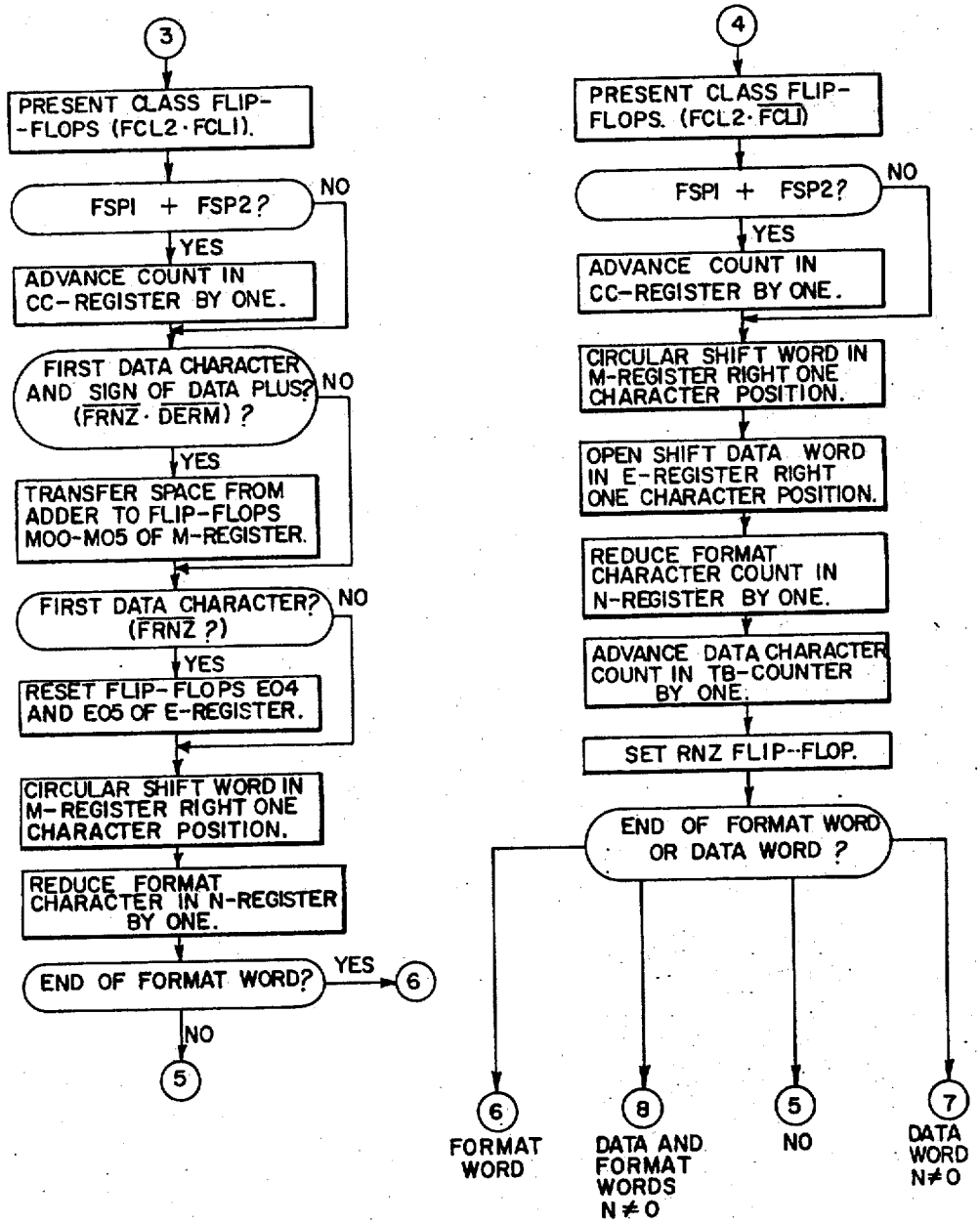

3
PRESENT CLASS FLIP-FLOPS (FCL2·FCL1).
FSP1 + FSP2?
NO
YES
ADVANCE COUNT IN CC-REGISTER BY ONE.
FIRST DATA CHARACTER AND SIGN OF DATA PLUS? (FRNZ·DERM)?
NO
YES
TRANSFER SPACE FROM ADDER TO FLIP-FLOPS M00-M05 OF M-REGISTER.
FIRST DATA CHARACTER? (FRNZ?)
NO
YES
RESET FLIP-FLOPS E04 AND E05 OF E-REGISTER.
CIRCULAR SHIFT WORD IN M-REGISTER RIGHT ONE CHARACTER POSITION.
REDUCE FORMAT CHARACTER IN N-REGISTER BY ONE.
END OF FORMAT WORD?
YES
6
NO
5
4
PRESENT CLASS FLIP-FLOPS. (FCL2·FCL1)
FSP1 + FSP2?
NO
YES
ADVANCE COUNT IN CC-REGISTER BY ONE.
CIRCULAR SHIFT WORD IN M-REGISTER RIGHT ONE CHARACTER POSITION.
OPEN SHIFT DATA WORD IN E-REGISTER RIGHT ONE CHARACTER POSITION.
REDUCE FORMAT CHARACTER COUNT IN N-REGISTER BY ONE.
ADVANCE DATA CHARACTER COUNT IN TB-COUNTER BY ONE.
SET RNZ FLIP-FLOP.
END OF FORMAT WORD OR DATA WORD ?
6
FORMAT WORD
8
DATA AND FORMAT WORDS N≠0
5
NO
7
DATA WORD N≠0

FIG. 164c

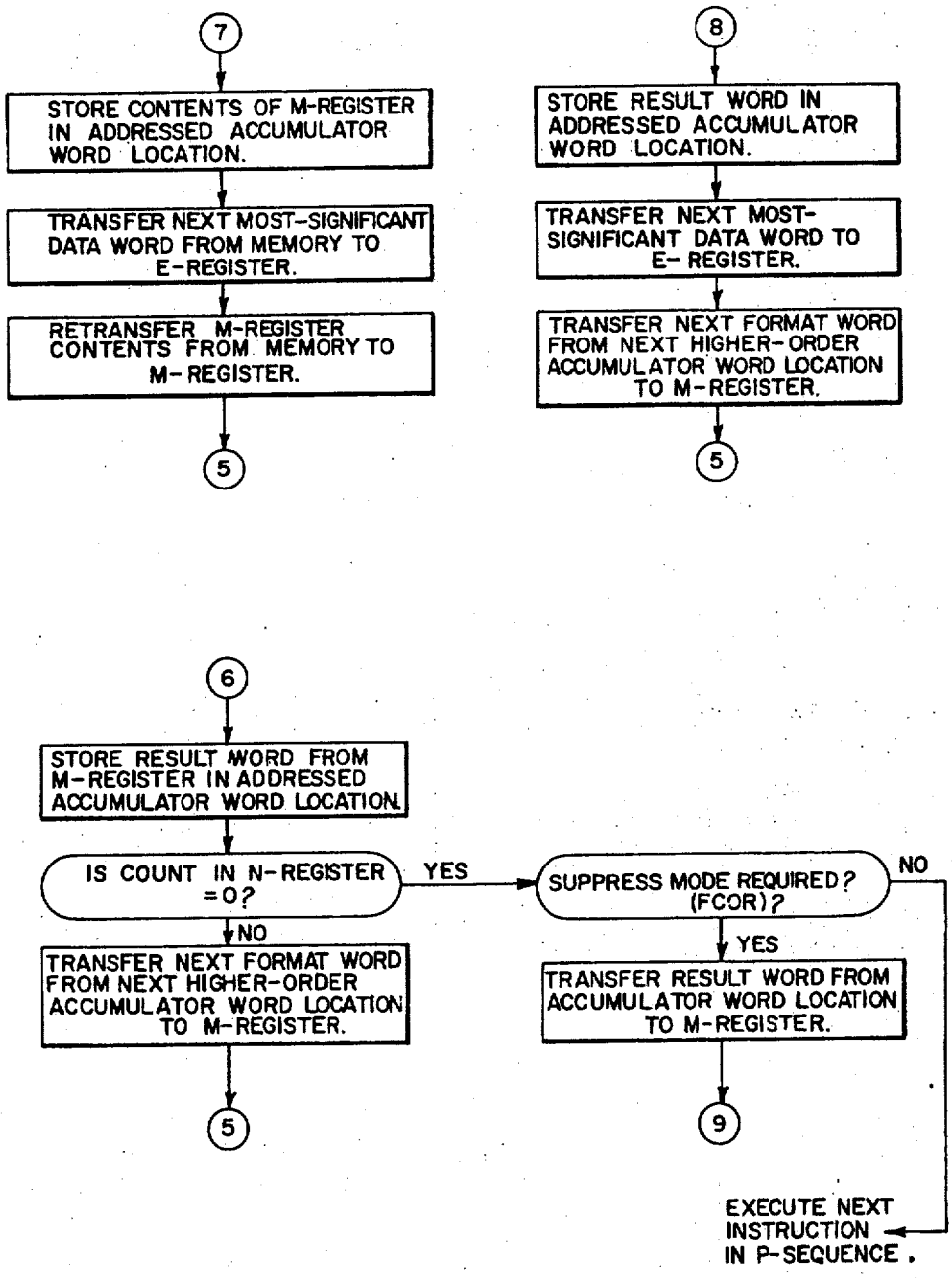

7
STORE CONTENTS OF M-REGISTER IN ADDRESSED ACCUMULATOR WORD LOCATION.
TRANSFER NEXT MOST-SIGNIFICANT DATA WORD FROM MEMORY TO E-REGISTER.
RETRANSFER M-REGISTER CONTENTS FROM MEMORY TO M-REGISTER.
5
8
STORE RESULT WORD IN ADDRESSED ACCUMULATOR WORD LOCATION.
TRANSFER NEXT MOST-SIGNIFICANT DATA WORD TO E-REGISTER.
TRANSFER NEXT FORMAT WORD FROM NEXT HIGHER-ORDER ACCUMULATOR WORD LOCATION TO M-REGISTER.
5
6
STORE RESULT WORD FROM M-REGISTER IN ADDRESSED ACCUMULATOR WORD LOCATION.
IS COUNT IN N-REGISTER =0?
YES
NO
TRANSFER NEXT FORMAT WORD FROM NEXT HIGHER-ORDER ACCUMULATOR WORD LOCATION TO M-REGISTER.
5
SUPPRESS MODE REQUIRED? (FCOR)?
NO
YES
TRANSFER RESULT WORD FROM ACCUMULATOR WORD LOCATION TO M-REGISTER.
9
EXECUTE NEXT INSTRUCTION IN P-SEQUENCE.

FIG. 164d

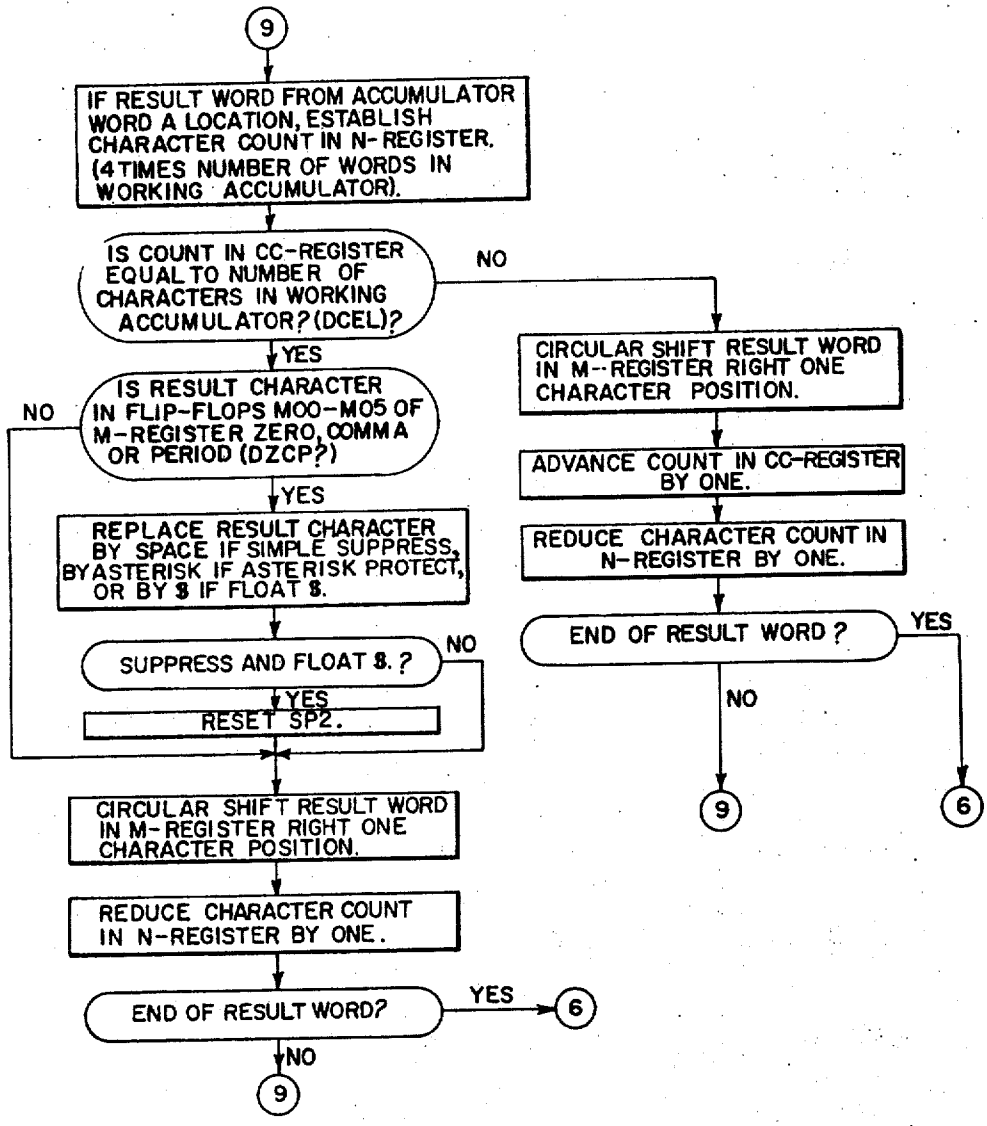

9
IF RESULT WORD FROM ACCUMULATOR WORD A LOCATION, ESTABLISH CHARACTER COUNT IN N-REGISTER. (4 TIMES NUMBER OF WORDS IN WORKING ACCUMULATOR).
IS COUNT IN CC-REGISTER EQUAL TO NUMBER OF CHARACTERS IN WORKING ACCUMULATOR? (DCEL)?
NO
YES
IS RESULT CHARACTER IN FLIP-FLOPS M00-M05 OF M-REGISTER ZERO, COMMA OR PERIOD (DZCP?)
NO
YES
REPLACE RESULT CHARACTER BY SPACE IF SIMPLE SUPPRESS, BY ASTERISK IF ASTERISK PROTECT, OR BY $ IF FLOAT $.
SUPPRESS AND FLOAT $.?
NO
YES
RESET SP2.
CIRCULAR SHIFT RESULT WORD IN M-REGISTER RIGHT ONE CHARACTER POSITION.
REDUCE CHARACTER COUNT IN N-REGISTER BY ONE.
END OF RESULT WORD?
YES
6
NO
9
CIRCULAR SHIFT RESULT WORD IN M--REGISTER RIGHT ONE CHARACTER POSITION.
ADVANCE COUNT IN CC-REGISTER BY ONE.
REDUCE CHARACTER COUNT IN N-REGISTER BY ONE.
END OF RESULT WORD ?
YES
NO
9
6

FIG. 164e

FIG. 165a

TIMING DIAGRAM - INSTRUCTION 67 (CPO)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 (Fig. 119)					Transfer instruction word from Memory to E-Register.
W01 - If development of first address required (Fig. 121).					
W01 - To develop second address (Fig. 121 except as indicated).	TL2 QTCK	$\overline{DCP0}\Delta MWR$			MWR remains reset to 0-state to inhibit restoration to Memory of word from location corresponding to second address.
		$\overline{DCP0}\Delta ESX$			ESX remains reset to 0-state to inhibit application of instruction word in E-Register to Adder.
	TL3			$DVCA \cdot \overline{FE00} \cdot \overline{FE01} \Delta(DCM0, DCM1, DCM2, DCM3)$	If "request status of processor" operation, word read from memory location corresponding to second address cleared from M-Register.
	TL5			$DVCA \cdot FE00 \Delta DSCA$	If second address development complete and flip-flop E00 set to 1-state, signal DSCA issues.
				$DVCA \cdot FE01 \Delta DSCZ$	If second address development complete and flip-flop E01 set to 1-state, signal DSCZ issues.
		$DSCA \cdot FM05 \Delta TVO$ $DSCA \cdot FM02 \Delta PIT$ $DSCA \cdot FM00 \Delta LES$ $DSCA \cdot FM01 \Delta GRE$ $DSCA \cdot FM03 \Delta MPL$ $DSCA \cdot FM04 \Delta \overline{FM05}(FTVO + FE01) \Delta FVO$			If "set status by ORing" operation, set indicator flip-flop to 1-state, if reset to 0-state, and if corresponding flip-flop of M-Register is set to 1-state. Set FVO only if TVO reset.

FIG. 165b
TIMING DIAGRAM - INSTRUCTION 67 (CPO) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5 (cont.)		DSCZ·$\overline{FM05}$ΔTVO DSCZ·$\overline{FM04}$ΔFVO DSCZ·$\overline{FM03}$ΔMPL DSCZ·$\overline{FM02}$ΔPIT DSCZ·$\overline{FM01}$ΔGRE DSCZ·FM00ΔLES		If "set status by ANDing" operation, reset indicator flip-flop to 0-state, if set to 1-state, and if corresponding flip-flop of M-Register is reset to 0-state. If "set status by loading" operation, both DSCA and DSCZ issue and states of indicator flip-flops are controlled by status of corresponding flip-flop of M-Register. FVO is set only if TVO reset.
	$\overline{QTCK}$		$\overline{MSX}$		
W03S	TL2			$\overline{DFXW}$ΔDDBX	If ACF of instruction word ≠ 1-6, transfer address of second memory location from D-Register to Memory. DDBX is present during entire W03S block.
				DFXWΔDQBX	If ACF of instruction word = 1-6, transfer address of second memory location from Q-Register to Memory. DQBX is present during entire W03S block.
				$\overline{FE00}$·$\overline{FE01}$ ΔDSSC	If "request status of processor" operation, DSSC issues. DSSC is present during entire W03S block.
				DGMM	DGMM issues.

FIG. 165c

TIMING DIAGRAM - INSTRUCTION 67 (CPO) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL2 (Cont.)	DSSC·DGMM ·FLES△M00 DSSC·DGMM ·FGRE△M01 DSSC·DGMM ·FMPL△M03 DSSC·DGMM ·FFV0△M04 DSSC·DGMM ·FTV0△M05			During "request status of processor" operation, transfer states of indicator flip-flops to corresponding status flip-flop of M-Register.
	$\overline{QTCK}$	MWR			Initiate write operation in Memory.
	TL3				
	$\overline{QTCK}$				
	TL4		DSSC△(FV0,PEF) DSSC·FPIT△TV0 DSSC·FPIT·FPAD △(ICW,MPI)		Reset FV0 and PEF during "request status of processor" operation; reset TV0 if PIT set; also reset ICW and MPI if PIT and PAD set.
	$\overline{QTCK}$				
	TL5				
	$\overline{QTCK}$	PSX			Apply address of next instruction in P-sequence to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	$\overline{QTC0}$		WR1, WR0		Go to W00 block.
	QTRK	RCK			Start Clock Generator on signal DMBU from Memory.

FIG. 166
TIMING DIAGRAM - INSTRUCTION 00 (HLT)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - (Fig. 119 except as indicated).	TL4				
	QTCK	$\overline{\text{DAMS}}\Delta\text{MAN}$			If address development not required, set MAN.
W01 - If address development required (Fig. 121 except as indicated).	TL4				
	QTCK	DXIMΔMAN			When address development completed, set MAN.
W10S	TL2				
	QTCK				
	TL3				
	QTCK				
	TL4				
	QTCK				
	TL5		DSX, IRE, CS2, CS3, DCE, CSF		
	QTCK	PSX	MSX, MOD, SRI, ESX, CAY, RCA		Apply address of next instruction to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0		$\overline{\text{FMAN}}\Delta$(WR3, WR1)		If manual flip-flop reset, go to W00 block. If manual flip-flop set, repeat W10S block.
	QTRK	RCK			Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.

FIG. 167a
TIMING DIAGRAM - INSTRUCTION 07 (GEN)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W00 - [Fig. 119 and Fig. 175, if Instruction 07 (GEN) in PIW location during program interrupt.]					
W01 - If development of instruction address field required (Fig. 121)					
W01 - To develop second address [Fig. 121, except as indicated, and Fig. 176 if Instruction 07 (GEN) in PIW location during program interrupt].	TL1	DBRS (DCBC +$\overline{\text{DNBS}}$+DPAN +$\overline{\text{FPRI}}$·DPRC) ΔBSY			Set BSY if input/output operation cannot be initiated. For example, if selected channel is busy, non-existent channel selected, no power on selected channel, etc.
	TL2				
	QTCK	$\overline{\text{DGEN}}$ΔMWR			MWR remains reset to prevent restoration to Memory of word from location specified by second address.
		$\overline{\text{DGEN}}$ΔESX			ESX remains reset to prevent application to Adder of instruction word in E-Register.

FIG. 167b
TIMING DIAGRAM - INSTRUCTION 07 (GEN) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL3			DVCAΔ(DCM0, DCM1, DCM2, DCM3)	If second address development complete, clear word read from location specified by second address from M-Register.
				DVCA·$\overline{\text{FBSY}}$ΔDIOD	DIOD issues if channel and peripheral are not busy.
		DIODΔDFE			If single- or double-word channel, transfer device and function code through input/output matrix to channel.
	QTCK				
	TL4				
	QTCK				
	TL5				
	QTCK		MSX		
	TL0				
	QTC0	DVCA·FBSY Δ(WR1, TL2)	DVCA·FBSYΔTL0		If BSY set, go to W03S block.
		DVCA·$\overline{\text{FBSY}}$ Δ(WR3, WR1)			If BSY reset, go to W11R block.

FIG. 167c
TIMING DIAGRAM - INSTRUCTION 07 (GEN) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
W03S	TL2			$\overline{DFXW}\Delta DDBX$	If ACF of instruction word ≠ 1-6, transfer address of second memory location from D-Register to Memory. DDBX is present during entire W03S block.
				DFXWΔDQBX	If ACF of instruction word = 1-6, transfer address of second memory location from D-Register to Memory. DQBX is present during entire W03S block.
				FPIAΔDPIB	If channel PSW location is second memory location, transfer address of PSW location to Memory.
				DGMM	Transfer major status of peripheral subsystem to bit positions 0-4 of M-Register and substatus to bit positions 18-23.
	QTCK	MWR			Initiate write operation in Memory.
	TL3	$\overline{FBSY}\cdot$DBYC ΔIOR			IOR set to initiate R-sequence, if channel will transfer data.
	QTCK				
	TL4				
	QTCK				

FIG. 167d
TIMING DIAGRAM - INSTRUCTION 07 (GEN) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL5			$\overline{FBSY}\cdot\overline{DBYC}\Delta DREL$	Release pulse generated.
				$\overline{FBSY}\Delta DIOI$	Selected channel parity flip-flop reset, if not busy condition.
			$\overline{FBSY}\cdot ZCN0\Delta PER$		
				DRELΔDWST	Transmit release pulse to channel, if channel does not go busy.
			DRELΔGST		Reset GST.
			PRI		
	QTCK	PSX			Apply address of next instruction in P-sequence to Adder.
			RCK		Stop Program Processor Clock Generator.
	TL0				Wait for synchronizing signal from Memory.
	QTC0		WR1, WR0		Go to W00 block.
	QTRK	RCK			Start Clock Generator on signal $\overline{DMBU}$ from Memory.
W11R	TL0				
	QTCK	DRITΔWRS			Set WRS when "request device" pulse from peripheral causes DRIT to issue.
	TL1			FWRSΔDIDS	Transmit device code to peripheral if WRS set.
				DSWD$\cdot$DME0ΔDWD0	
	QTCK		RCK		Stop Program Processor Clock Generator.

FIG. 167e

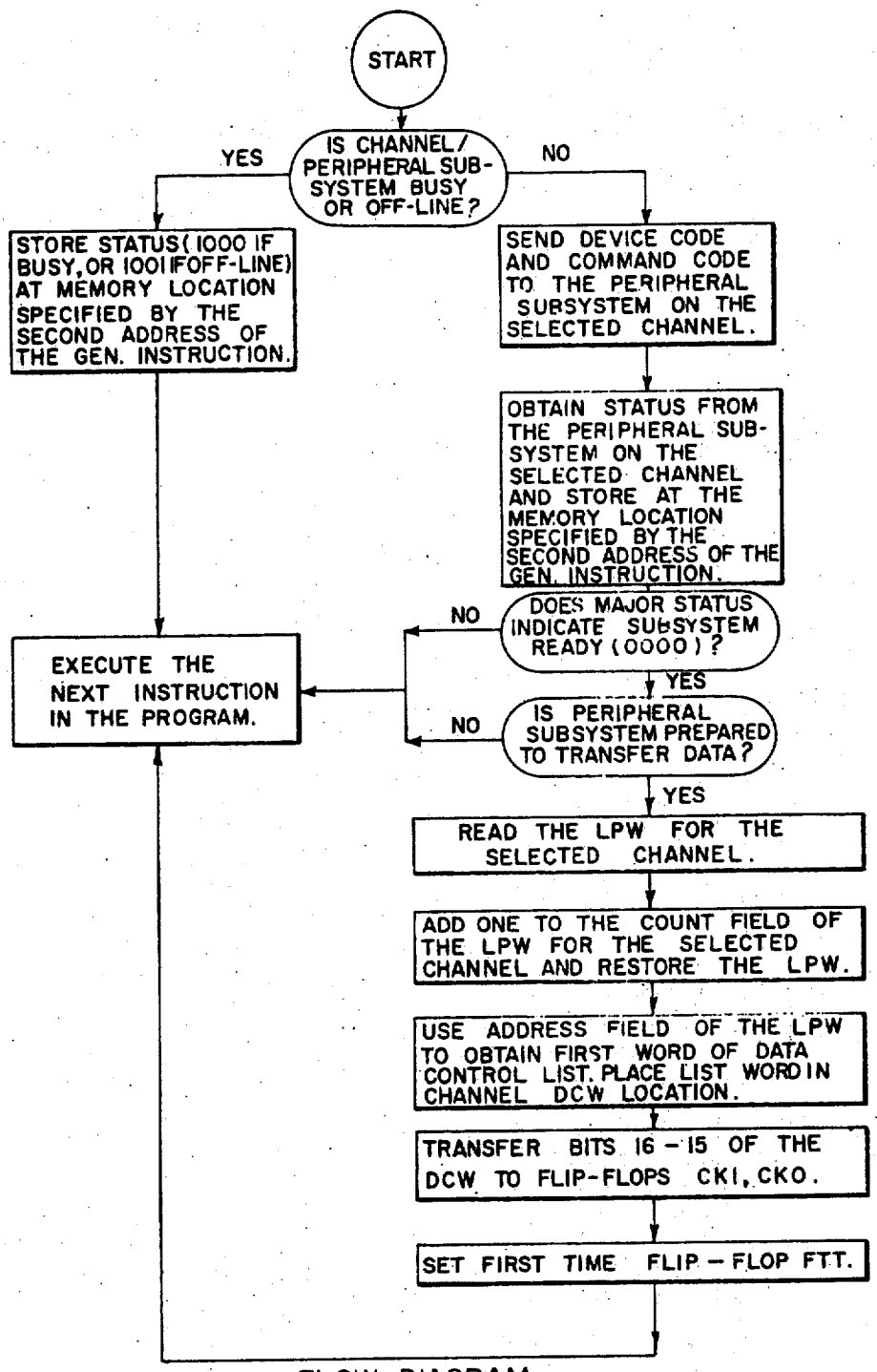
TIMING DIAGRAM - INSTRUCTION 07 (GEN) (Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	TL2			FWRS∆DWST	Notify peripheral that device code was transmitted.
				FWRS∆DRWR	Reset write receiver.
					Wait for synchronizing signal DMDA from Memory.
	QTRK		WRS		
		RCK			Start Clock Generator
	QTC2	$\overline{FWRS}$∆WR2,TL0 $\overline{FWRS}$∆TL0	FWRS ∆WR1, WR0,TL2 $\overline{FWRS}$∆TL2		If device code transmitted to peripheral subsystem, go to W12R block; if not, repeat W11R block.
W12R	TL0			DCCS·$\overline{FGST}$·$\overline{DDIB}$ ∆DSH3	If character channel, transfer function code through input/output matrix to channel.
	QTCK	DRIT∆WRS			Set WRS when "request function" pulse from peripheral causes DRIT to issue.
	TL1			FWRS∆DIDS	Transfer function code to peripheral if WRS set.
	QTCK		RCK		Stop Program Processor Clock Generator.
	TL2				Wait for synchronizing signal DMDA from Memory.
				FWRS∆DWST	Notify peripheral that function code has been transmitted.
				FWRS∆DRWR	Terminate function code request in channel.

FIG. 167f

TIMING DIAGRAM - INSTRUCTION 07 (GEN)(Cont.)

BLOCK	TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
	QTRK	RCK			Start Clock Generator.
			WRS		
	QTC2	$\overline{\text{FWRS}}\Delta$TL0	$\overline{\text{FWRS}}\Delta$TL2		If function code not transmitted to peripheral subsystem, repeat W12R block; otherwise go to W12S block.
W12S	TL2				
	QTCK				
	TL3	FWRSΔGST	FGSTΔDFE	FGST·DCCSΔDSH0 FGST·DWCSΔDSH3 DCCSΔDME0 DCCSΔDME3	Set GST. Transfer major status and substatus from channel through input/output matrix to M-Register.
	QTCK	DRITΔWRS			Set WRS when "request release" pulse from peripheral causes DRIT to issue.
	TL4			FWRSΔDRWR	
	QTCK				
	TL5				
	QTCK		RCK		Stop Program Processor Clock Generator.
	TL0				Wait for memory synchronizing signal.
	QTC0	FWRS ΔWR1,WR0, TL2	FWRS ΔWR3, WR2, TL0		If release pulse sent to peripheral subsystem, go to W03S block; otherwise repeat W12S block.
	QTRK	RCK	WRS		Start Clock Generator on signal $\overline{\text{DMBU}}$ from Memory.



FLOW DIAGRAM
INSTRUCTION 07 (GEN)

FIG. 168

FIG. 169
CENTRAL PROCESSOR ALERTS

CONDITIONS					RESULTS				
TYPE OF ERROR	PIT	PEF SET	ICW SET	WJE SET	PROCESSOR CHANNEL INTERRUPT		CENTRAL PROCESSOR HALTED		SEND END DATA TRANSFER SIGNAL DEDX TO CHANNEL
					IMMED.	END OF INSTR.	IMMED.	END OF INSTR.	
INVALID ADDRESS IN LPW DURING GEN INSTRUCTION	RESET		X			X			X
	SET		X					X	X
INVALID ADDRESS IN LPW DURING DATA TRANSFER SEQUENCE	RESET		X			X			X
	SET		X					X	X
INVALID ADDRESS IN DCW DURING DATA TRANSFER SEQUENCE	RESET		X			X			X
	SET		X					X	X
NO SPB OR GEN IN PIW	RESET		X				X		
INVALID OPERATION CODE IN PIW	RESET	X	X				X		
INVALID ADDRESS IN PSW	RESET	X			X				
	SET	X					X		
INVALID OPERATION CODE, OTHER THAN IN PIW	RESET	X			X				
	SET	X					X		
INVALID ADDRESS IN INDEX LINK OR INDEX POINTER	RESET	X			X				
	SET	X					X		
INVALID ADDRESS IN OPERAND LINK OR OPERAND POINTER	RESET	X			X				
	SET	X					X		
INVALID ADDRESSING OF INDIRECT WORD	RESET	X			X				
	SET	X					X		
INVALID ADDRESSING OF DATA	RESET	X			X				
	SET						X		
INVALID ADDRESS TAKEN FROM P-REGISTER	RESET	X			X				
	SET	X					X		
INCORRECT PARITY IN INSTRUCTION WORD				X			X		
INCORRECT PARITY IN OTHER THAN INSTRUCTION WORD				X				X	

CONTROL APPARATUS IN A DATA PROCESSING SYSTEM

Filed April 14, 1965 363 Sheets-Sheet 336

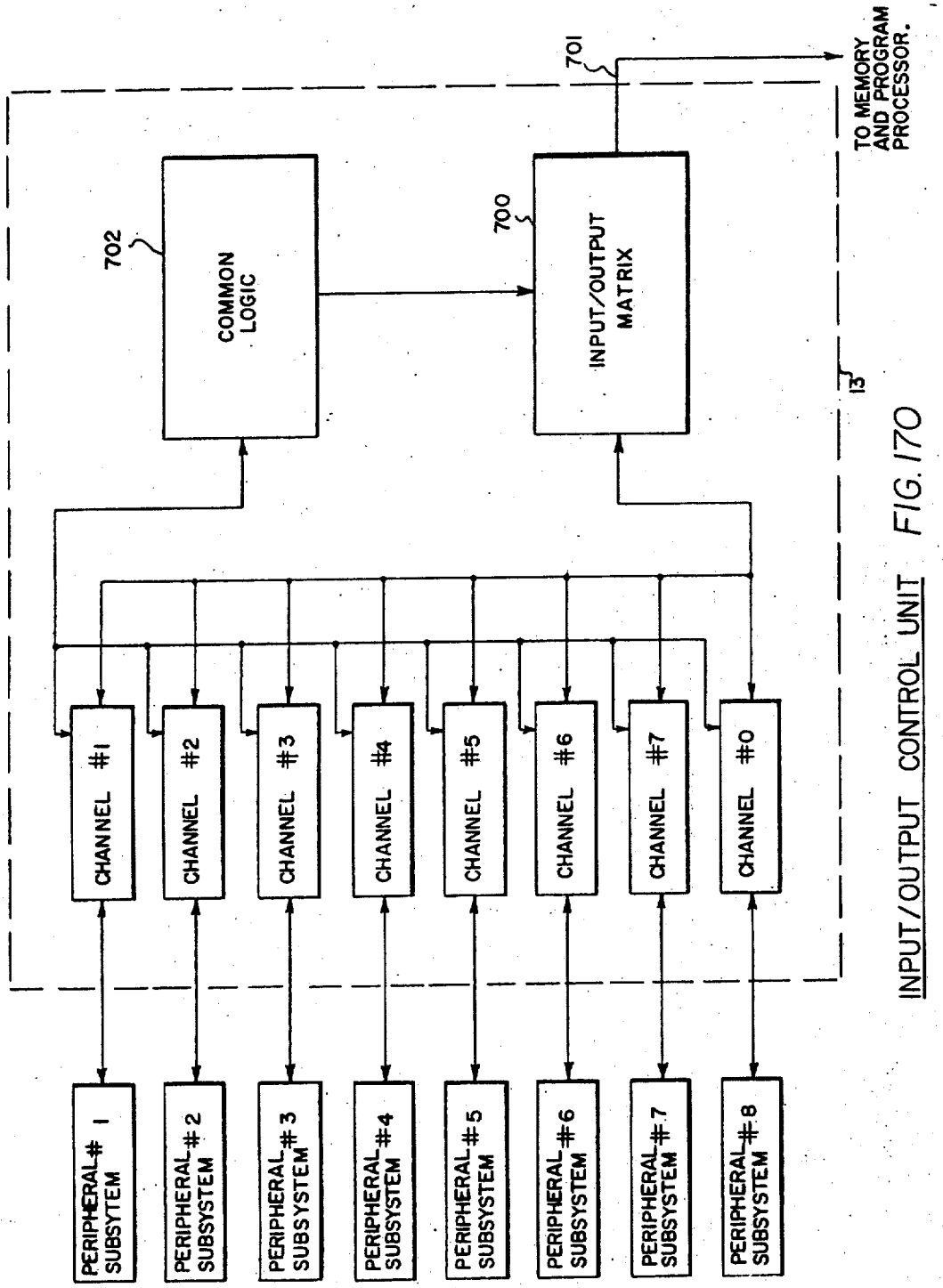
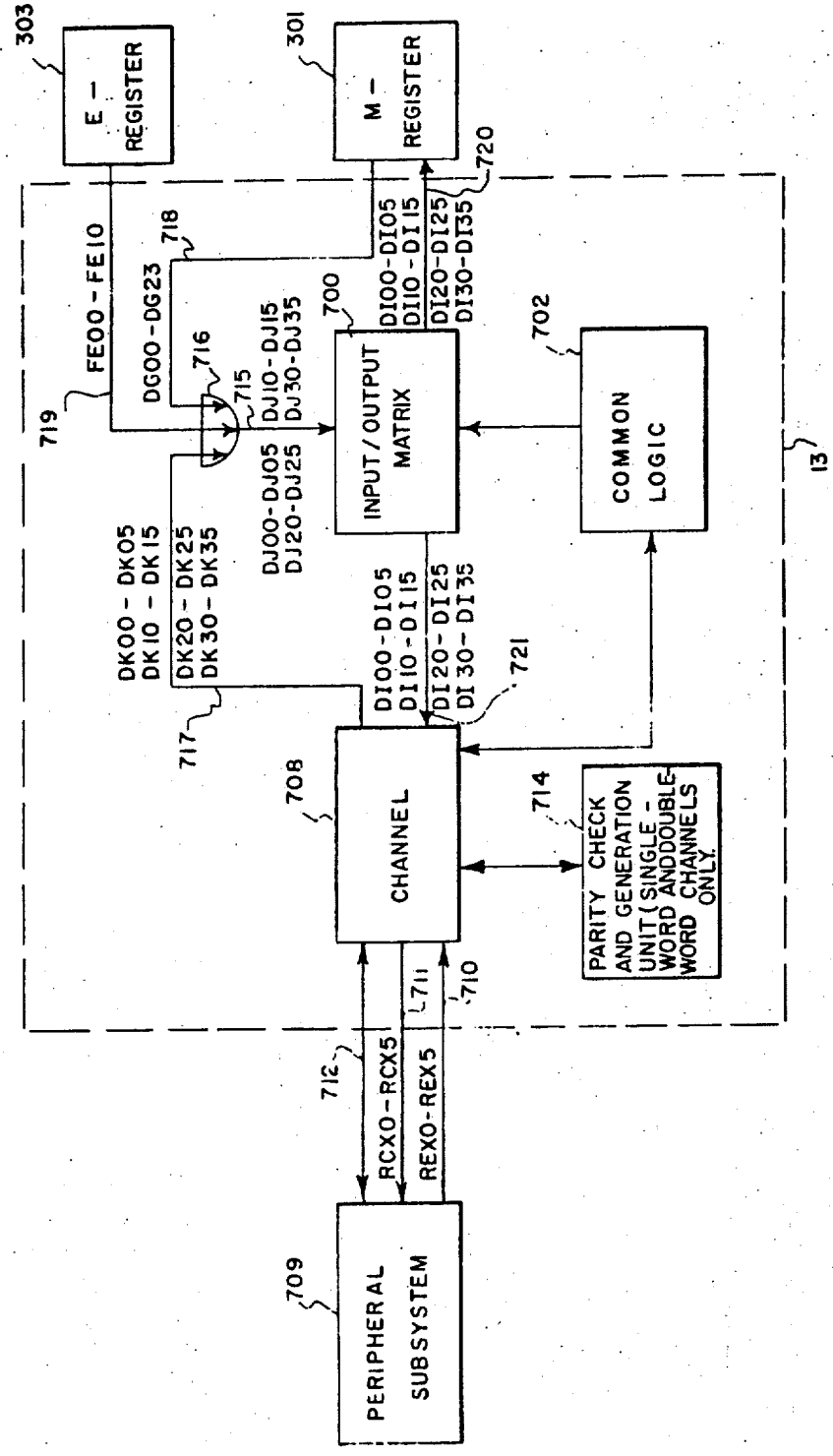


FIG. 170

INPUT/OUTPUT CONTROL UNIT



DATA TRANSFER
INPUT / OUTPUT CONTROL UNIT
FIG. 171

CONTROL APPARATUS IN A DATA PROCESSING SYSTEM

Filed April 14, 1965 363 Sheets-Sheet 337

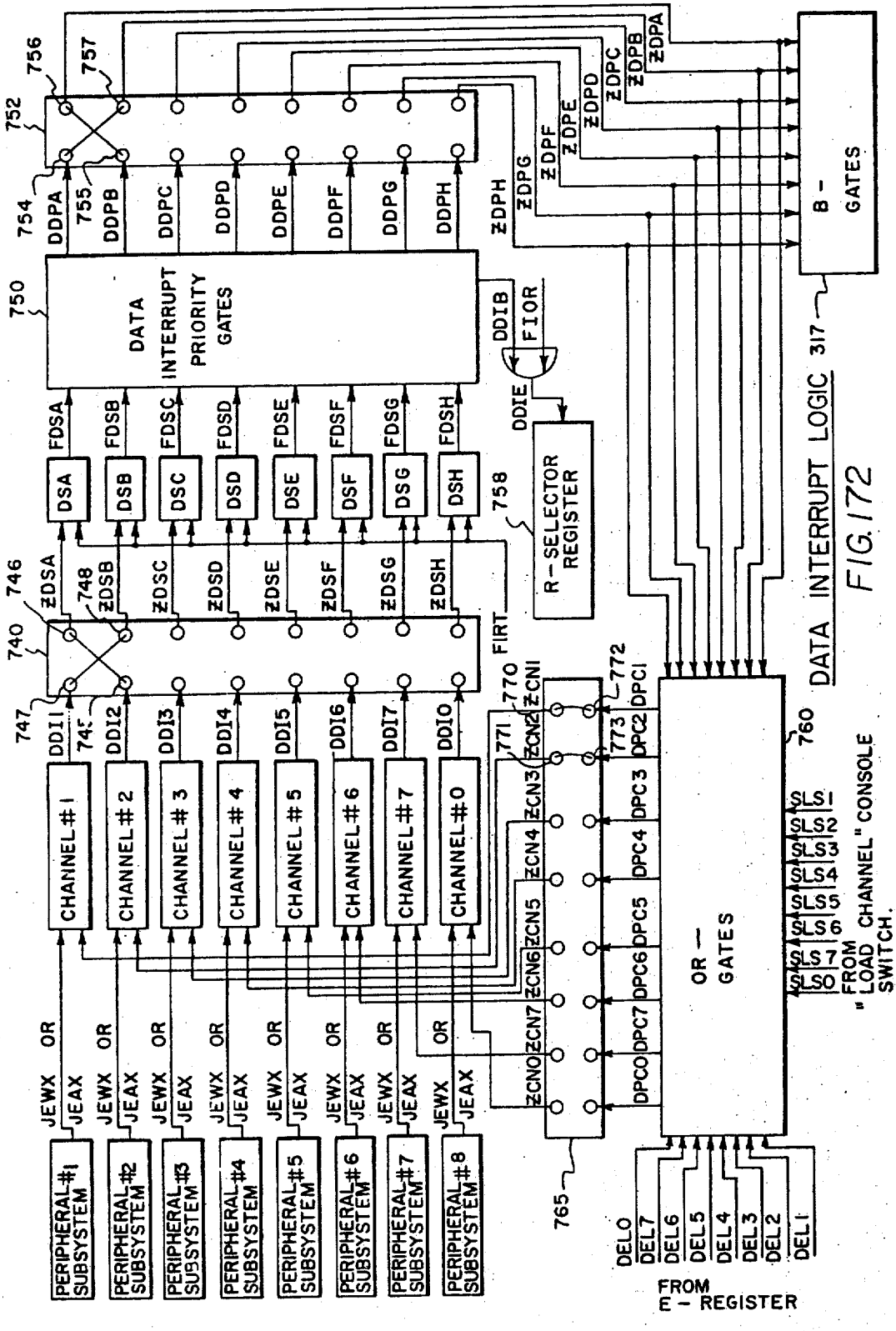
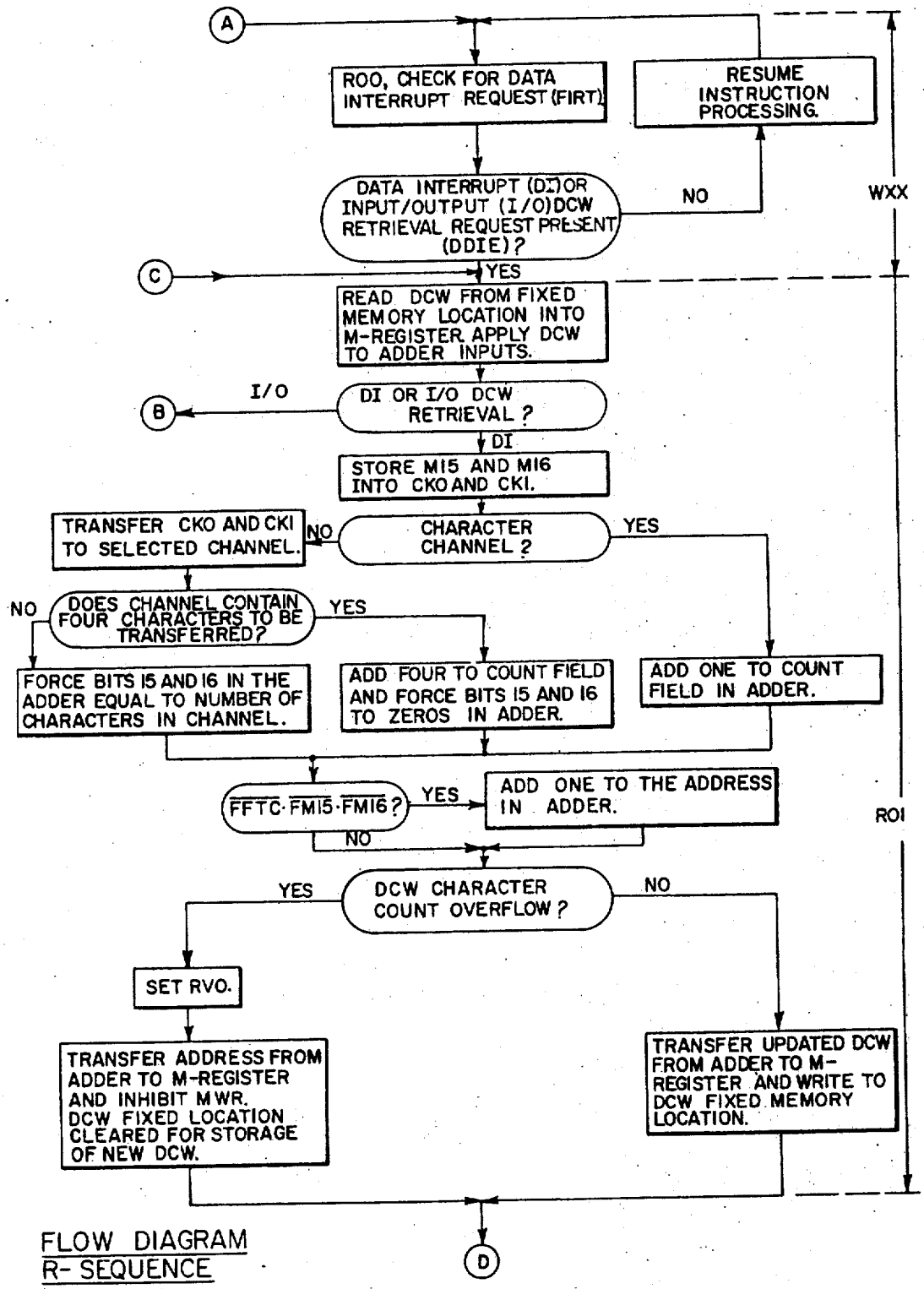


FIG. 172



FLOW DIAGRAM
R-SEQUENCE
FIG. 173a

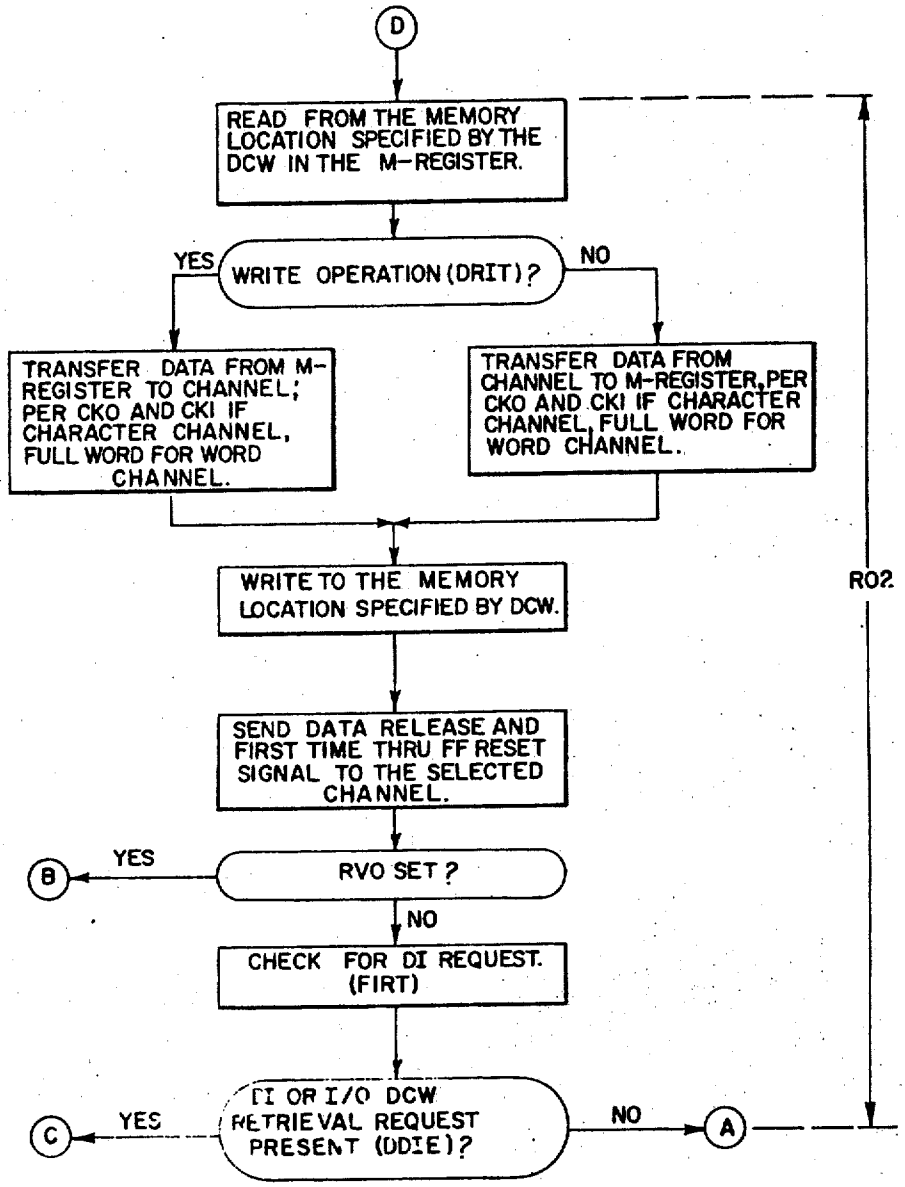

D
READ FROM THE MEMORY LOCATION SPECIFIED BY THE DCW IN THE M-REGISTER.
YES
WRITE OPERATION (DRIT)?
NO
TRANSFER DATA FROM M-REGISTER TO CHANNEL; PER CKO AND CKI IF CHARACTER CHANNEL, FULL WORD FOR WORD CHANNEL.
TRANSFER DATA FROM CHANNEL TO M-REGISTER, PER CKO AND CKI IF CHARACTER CHANNEL, FULL WORD FOR WORD CHANNEL.
WRITE TO THE MEMORY LOCATION SPECIFIED BY DCW.
RO2
SEND DATA RELEASE AND FIRST TIME THRU FF RESET SIGNAL TO THE SELECTED CHANNEL.
B
YES
RVO SET ?
NO
CHECK FOR DI REQUEST. (FIRT)
C
YES
DI OR I/O DCW RETRIEVAL REQUEST PRESENT (DDIE)?
NO
A

FIG. 173b

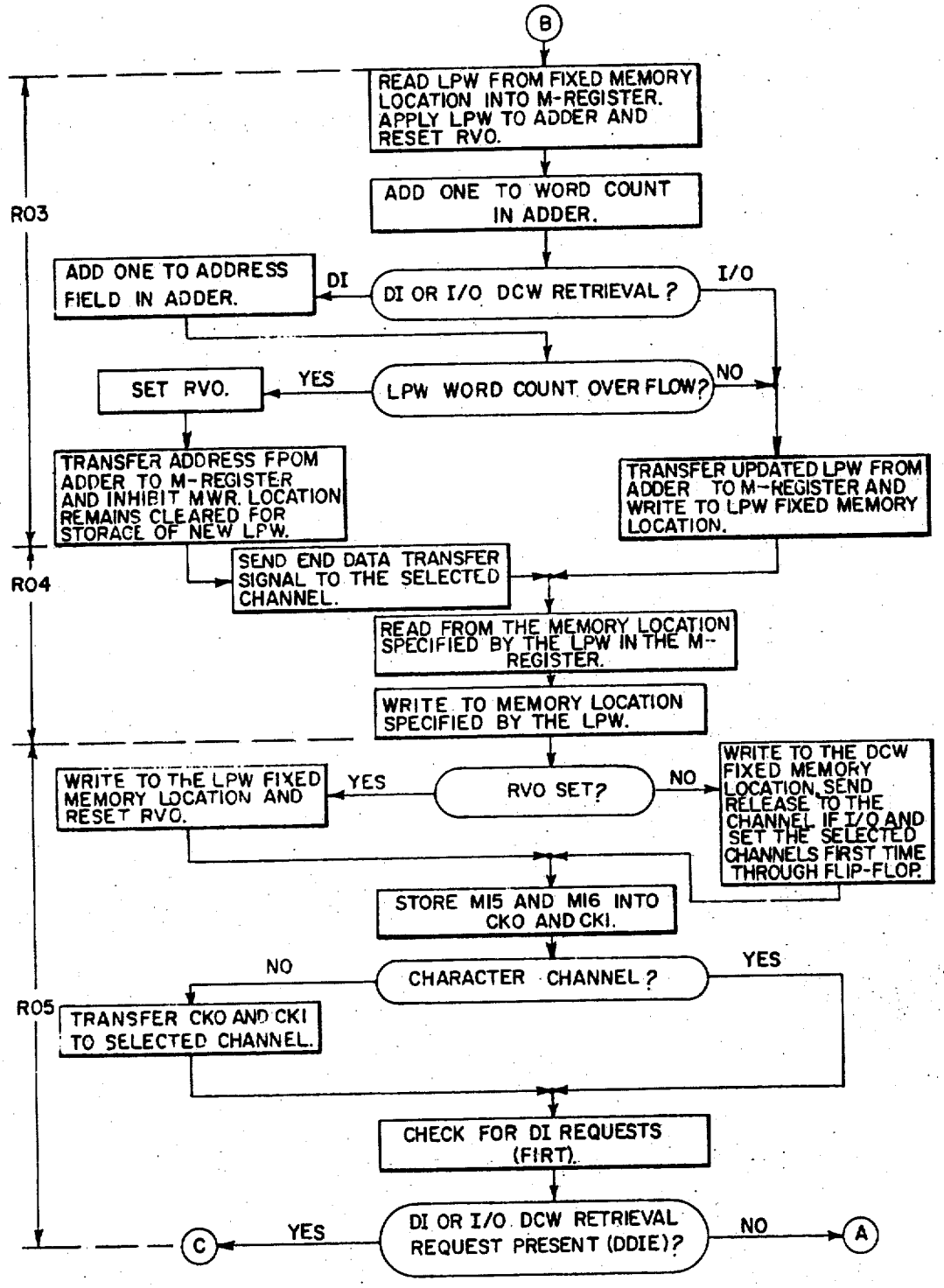
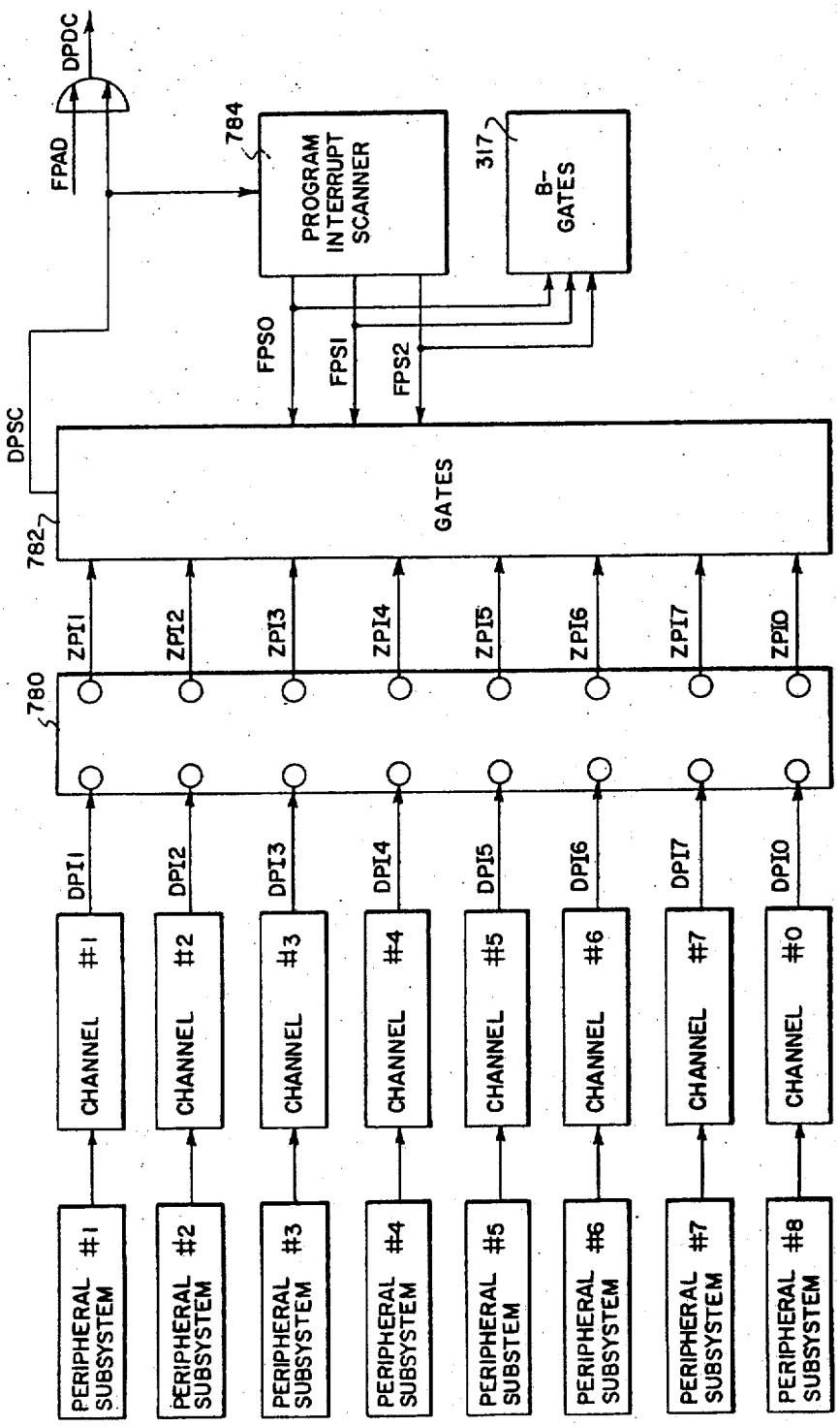

B
READ LPW FROM FIXED MEMORY LOCATION INTO M-REGISTER. APPLY LPW TO ADDER AND RESET RVO.
ADD ONE TO WORD COUNT IN ADDER.
R03
ADD ONE TO ADDRESS FIELD IN ADDER.
DI
DI OR I/O DCW RETRIEVAL ?
I/O
SET RVO.
YES
LPW WORD COUNT OVER FLOW?
NO
TRANSFER ADDRESS FPOM ADDER TO M-REGISTER AND INHIBIT MWR. LOCATION REMAINS CLEARED FOR STORAGE OF NEW LPW.
TRANSFER UPDATED LPW FROM ADDER TO M-REGISTER AND WRITE TO LPW FIXED MEMORY LOCATION.
R04
SEND END DATA TRANSFER SIGNAL TO THE SELECTED CHANNEL.
READ FROM THE MEMORY LOCATION SPECIFIED BY THE LPW IN THE M-REGISTER.
WRITE TO MEMORY LOCATION SPECIFIED BY THE LPW.
WRITE TO THE LPW FIXED MEMORY LOCATION AND RESET RVO.
YES
RVO SET?
NO
WRITE TO THE DCW FIXED MEMORY LOCATION, SEND RELEASE TO THE CHANNEL IF I/O AND SET THE SELECTED CHANNELS FIRST TIME THROUGH FLIP-FLOP.
STORE MI5 AND MI6 INTO CKO AND CKI.
NO
CHARACTER CHANNEL ?
YES
R05
TRANSFER CKO AND CKI TO SELECTED CHANNEL.
CHECK FOR DI REQUESTS (FIRT).
DI OR I/O DCW RETRIEVAL REQUEST PRESENT (DDIE)?
YES
NO
C
A

FIG. 173c



PROGRAM INTERRUPT LOGIC

FIG. 174

FIG. 175
TIMING DIAGRAM - W00 BLOCK DURING PROGRAM INTERRUPT

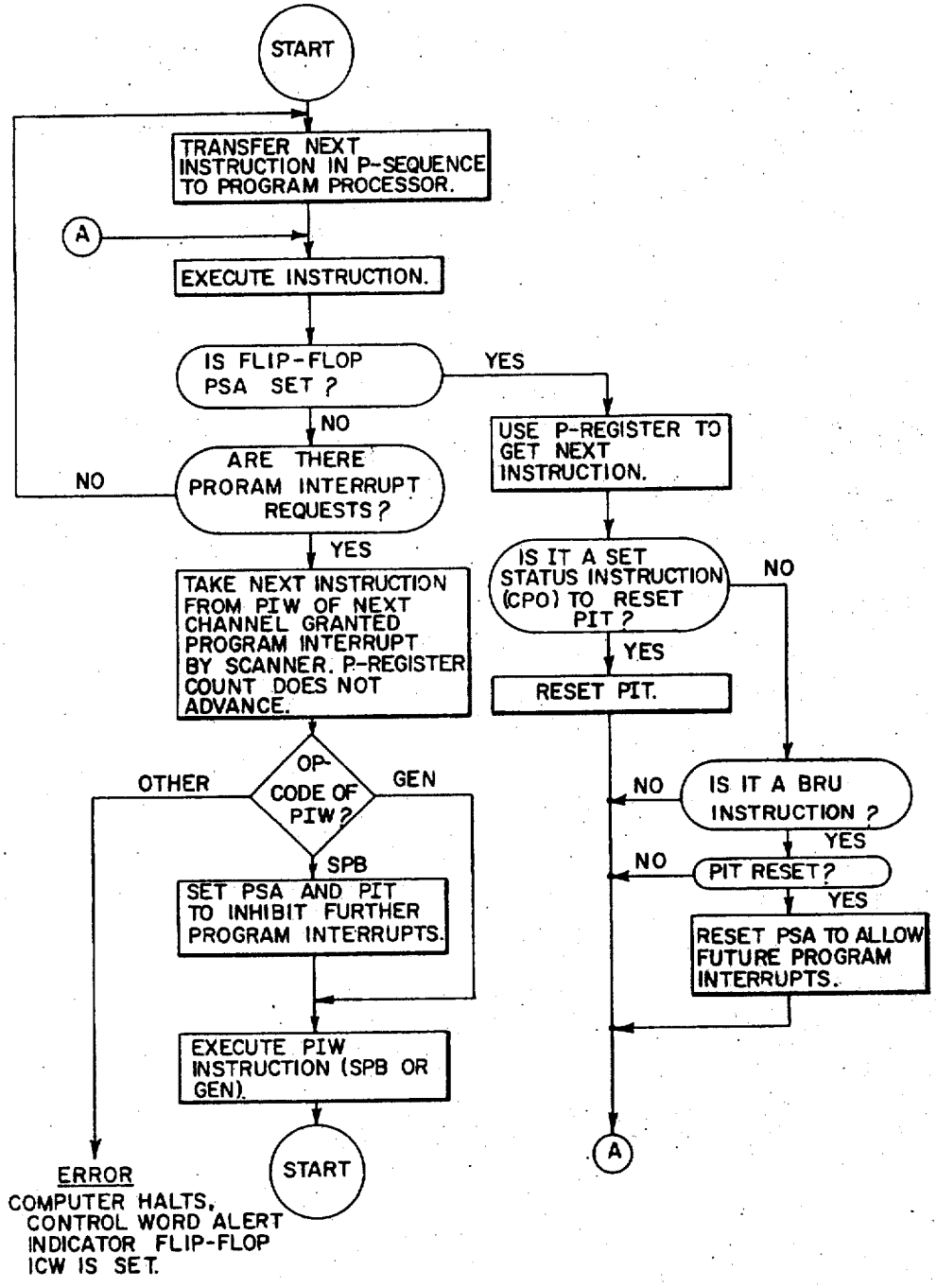
TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
TL0			FPIA·DPIB $\overline{\text{FPIA}}$·DDBX	Transfer address of channel PIW location to Memory. Inhibit transfer of address from D-Register to Memory.
TL3			$\overline{\text{FPIA}}$·$\overline{\text{FMAN}}$ DPCU $\overline{\text{FPIA}}$·DNQX	Inhibit advance of count in P-Register. Inhibit transfer of address control field of instruction word to Q-Register.
TL4				
QTCK	FPIA·$\overline{\text{DGEN}}$·$\overline{\text{DSPB}}$·MAN			Set MAN if instruction in PIW location is not Instruction 07 (GEN) or Instruction 17 (SPB).
TL5		$\overline{\text{FPIT}}$·DBRU ·PSA		Reset PSA if interrupted program to be resumed.

FIG. 176
TIMING DIAGRAM - W01 BLOCK DURING PROGRAM INTERRUPT

TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS	EXPLANATION
TL0	DSPB·FPIA△PSA			Set PSA to inhibit granting of other program interrupt requests.
			FPIA△DPIB	Transfer address of channel PSW location to Memory.
			$\overline{\text{FPIA}}$△DDBX	Inhibit transfer of address in D-Register to Memory.
TL3			$\overline{\text{FXS2}}\cdot\overline{\text{FXS1}}\cdot\overline{\text{FPIA}}$△DPCU	Sequence 0: Inhibit advance of count in P-Register.
TL5		$\overline{\text{FPIT}}$·DBRU·DXIM△PSA		Reset PSA if interrupted program to be resumed.

CONTROL APPARATUS IN A DATA PROCESSING SYSTEM

Filed April 14, 1965 363 Sheets-Sheet 344

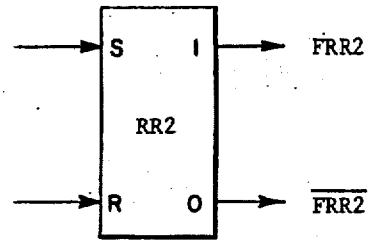


FLOW DIAGRAM
PROGRAM INTERRUPT

FIG. 177

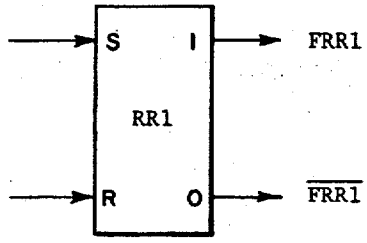
$FRR2 = DR03\ QTC0$

$\overline{FRR2} = DR05\ QTC0 + FCSF\ QTC2$



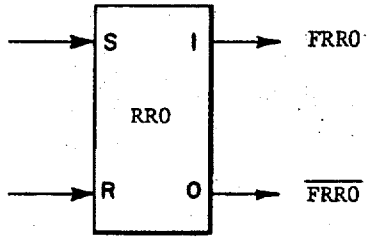
$FRR1 = DR01\ \overline{FMNX}\ QTC0 + DR01\ \overline{DDIB}\ QTC2$

$\overline{FRR1} = (DRRV + DR03)\ QTC0 + FCSF\ QTC2$



$FRR0 = [DR00\ DDIE + DR02\ (FRV0 + DDIE) + DR04]\ QTC0$

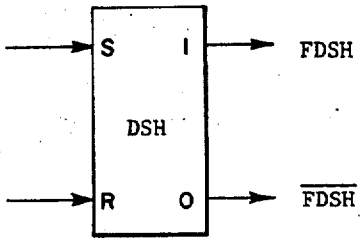
$\overline{FRR0} = (DR01 + DR03 + DR05\ \overline{DDIE})\ QTC0 + FCSF\ QTC2$



INPUT/OUTPUT CONTROL UNIT
FIG. 178

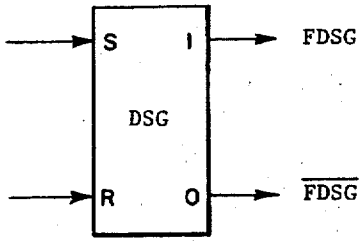
$FDSH = ZDSH\ \overline{FIRT}$

$\overline{FDSH} = DRDI$



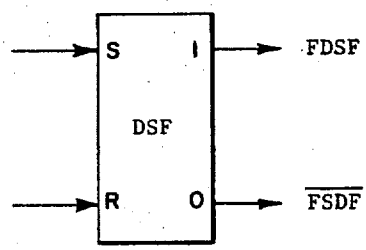
$FDSG = ZDSG\ \overline{FIRT}$

$\overline{FDSG} = DRDI$



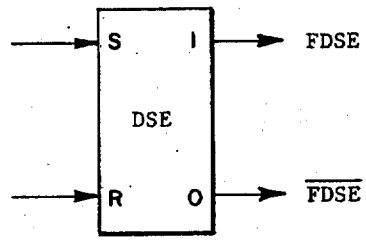
$FDSF = ZDSF\ \overline{FIRT}$

$\overline{FDSF} = DRDI$



$FDSE = ZDSE\ \overline{FIRT}$

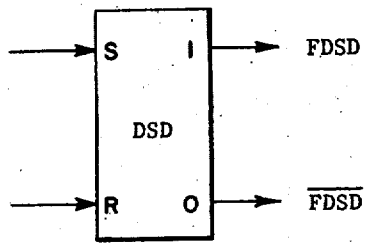
$\overline{FDSE} = DRDI$



INPUT/OUTPUT CONTROL UNIT
FIG. 179

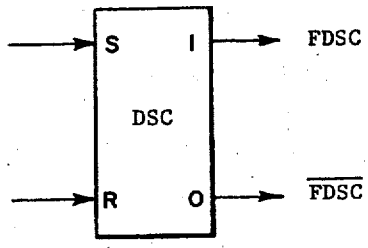
FDSD = ZDSD $\overline{\text{FIRT}}$

$\overline{\text{FDSD}}$ = DRDI



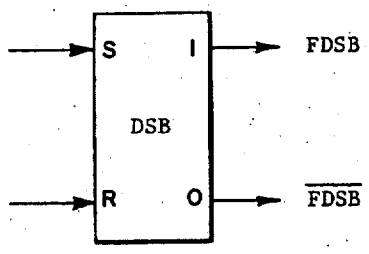
FDSC = ZDSC $\overline{\text{FIRT}}$

$\overline{\text{FDSC}}$ = DRDI



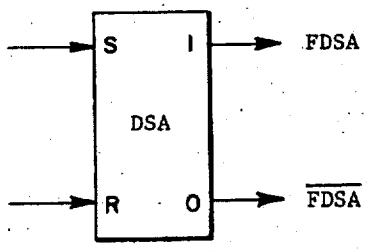
FDSB = ZDSB $\overline{\text{FIRT}}$

$\overline{\text{FDSB}}$ = DRDI



FDSA = ZDSA $\overline{\text{FIRT}}$

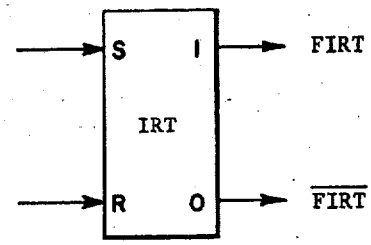
$\overline{\text{FDSA}}$ = DRDI



INPUT/OUTPUT CONTROL UNIT
FIG. 180

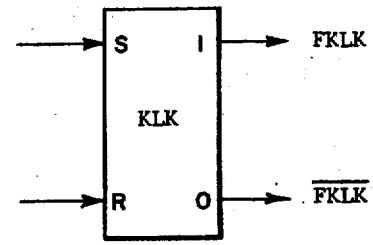
FIRT = DIRS QTLP

$\overline{FIRT}$ = FIRT QTLP



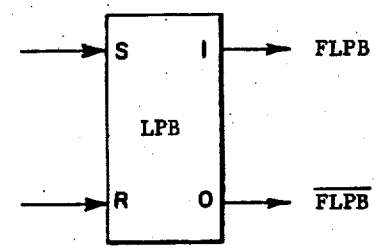
FKLK = DCCE QTCK

$\overline{FKLK}$ = $\overline{DCCE}$ QTCK



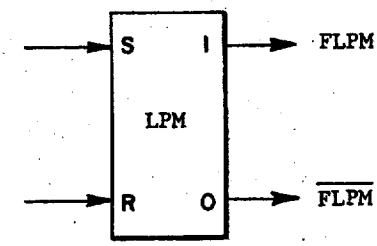
FLPB = FLPM DPRC

$\overline{FLPB}$ = DMLR + FSTT + FSTE



FLPM = SLPS FNBY

$\overline{FLPM}$ = DMLR + FSTT + FSTE

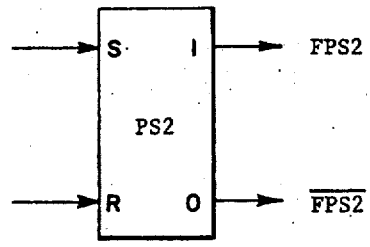


INPUT/OUTPUT CONTROL UNIT

FIG. 181

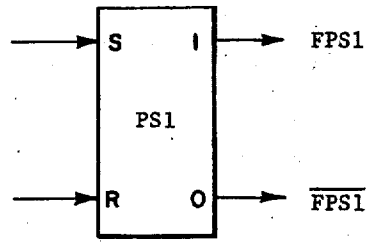
$FPS2 = \overline{FPS1}\ \overline{FPS2}$

$\overline{FPS2} = \overline{FPS1}\ FPS2 + DPSP + DMLR$



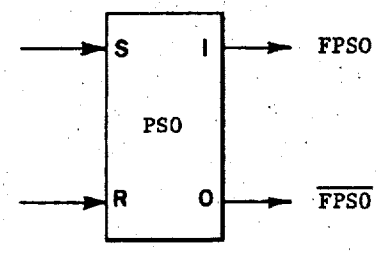
$FPS1 = \overline{FPS0}\ \overline{FPS1}$

$\overline{FPS1} = \overline{FPS0}\ FPS1 + DPSP + DMLR$



$FPS0 = \overline{FPIT}\ \overline{FPS0}\ \overline{DPSC}\ FKLK + DPSP$

$\overline{FPS0} = \overline{FPIT}\ FPS0\ \overline{DPSC}\ FKLK + DMLR$

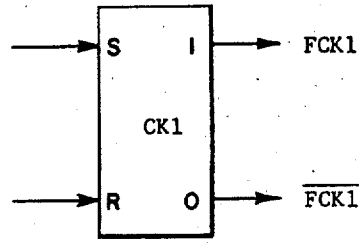


INPUT/OUTPUT CONTROL UNIT

FIG. 182

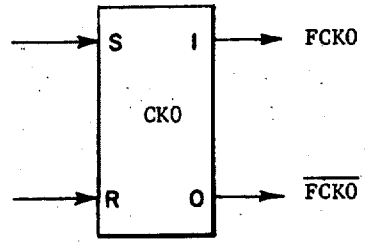
$FCK1 = \overline{FCK0}\ \overline{FCK1} + FM16\ DCTC\ \overline{DSCS}$

$\overline{FCK1} = \overline{FCK0}\ FCK1 + DMLR + DCKR\ \overline{FMNX}$



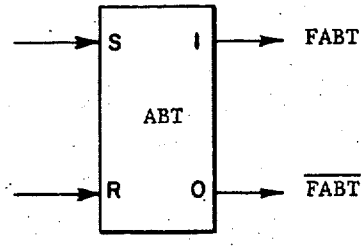
$FCK0 = \overline{FCK0}\ DCK0 + FM15\ DCTC\ \overline{DSCS}$

$\overline{FCK0} = FCK0\ DCK0 + DMLR + DCKR\ \overline{FMNX}$



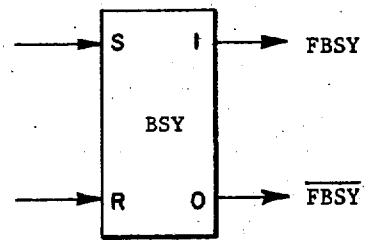
$FABT = DBT2$

$\overline{FABT} = DMLR$



$FBSY = DIBC\ DBRS$

$\overline{FBSY} = DW00\ FTL5$

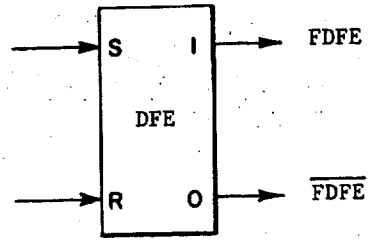


INPUT/OUTPUT CONTROL UNIT

FIG. 183

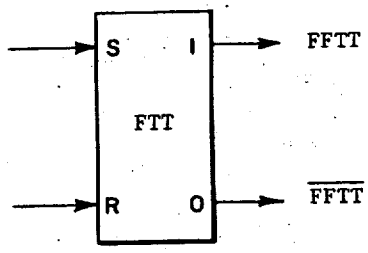
FDFE = DIOD

$\overline{FDFE}$ = DMLR + FGST



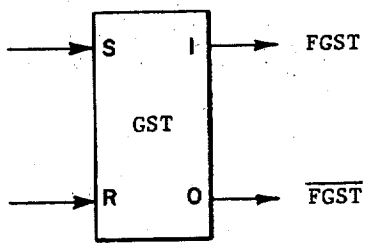
FFTT = DSFT ZCNO

$\overline{FFTT}$ = DRWR ZCNO + DMLR



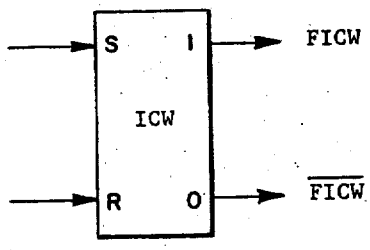
FGST = $\overline{DFUN}$ $\overline{FGST}$

$\overline{FGST}$ = DMLR + DREL



FICW = DIAC + DIIP

$\overline{FICW}$ = DMIR + DMLR + SIWR

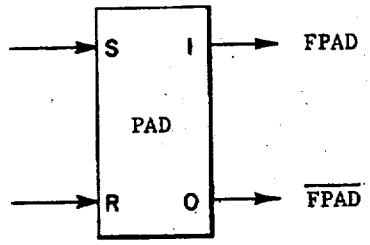


INPUT/OUTPUT CONTROL UNIT

FIG. 184

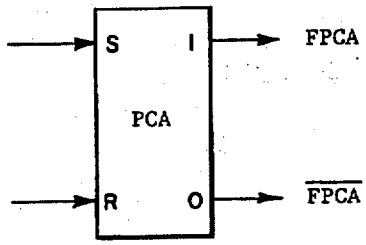
$FPAD = \overline{FPIA}\ \overline{FPSA}\ (FMPI + FTVO\ FFVO + FAIM)$

$\overline{FPAD} = DMLR + \overline{FMPI}\ \overline{FFVO}\ \overline{FPEF}\ \overline{FICW}\ FTL5$



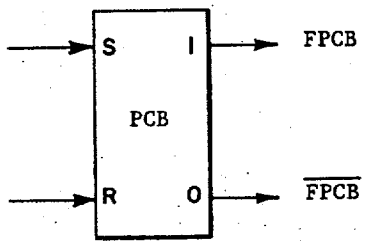
$FPCA = \overline{FPCB}\ FKLK$

$\overline{FPCA} = FPCB\ FKLK + DMLR$



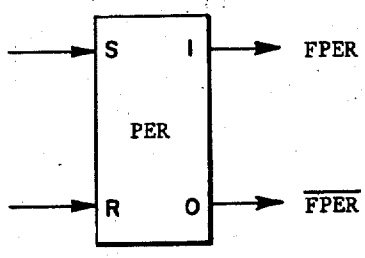
$FPCB = FPCA\ \overline{FLKL}$

$\overline{FPCB} = \overline{FPCA}\ \overline{FKLK} + DMLR$



$FPER = DSPE\ ZCNO$

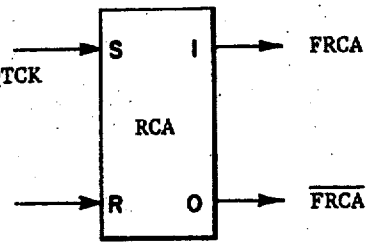
$\overline{FPER} = DIOI\ ZCNO + DMLR$



INPUT/OUTPUT CONTROL UNIT
FIG. 185

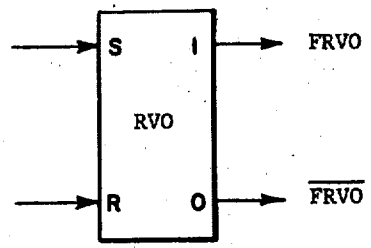
$FRCA = DR03\ DDIB\ FTL2\ \overline{FMEM}\ QTCK + (\overline{FM15}\ \overline{FM16}\ FTL2\ DR01\ \overline{FMNX}\ \overline{DFTC})\ QTCK$

$\overline{FRCA} = FTL5\ QTCK$



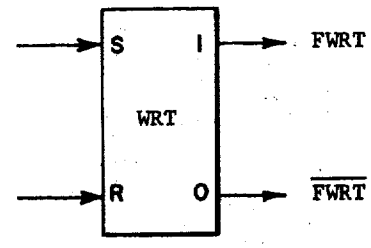
$FRVO = DR13\ FTL3\ \overline{MC23}\ \overline{FMNX}\ DDIB$

$\overline{FRVO} = DIAC + DMVR + DMLR + DR03\ FTL1$



$FWRT = DSWR\ FTL1$

$\overline{FWRT} = \overline{DSWR}\ FTL1 + DMLR$



INPUT/OUTPUT CONTROL UNIT

FIG. 186

LOGICAL COMBINATION SIGNALS:

DME3 DDTE FTL4 $\overline{\text{FWRT}}$	= DBR0
DME2 DDTE FTL4 $\overline{\text{FWRT}}$	= DBR1
DME1 DDTE FTL4 $\overline{\text{FWRT}}$	= DBR2
DME0 DDTE FTL4 $\overline{\text{FWRT}}$	= DBR3
REBX	= DBRB
FPCA $\overline{\text{FPCB}}$	= DBT1
FPCA FPCB	= DBT2
$\overline{\text{FPCA}}$ FPCB	= DBT3
DBTS FABT	= DBTE
$\overline{\text{FPCA}}$ $\overline{\text{FPCB}}$ + FPCA FPCB	= DBTS
DCBC $\overline{\text{FE05}}$ (FE00 + FE01 + FE02 + FE03)	= DBYC
REBX ZCN0	= DCBC
DGEN (FTL0 + FTL2 + FTL4) + $\overline{\text{DGEN}}$ $\overline{\text{FKLK}}$	= DCCE
ZCN0	= DCCS
$\overline{\text{FCK1}}$ $\overline{\text{FCK0}}$	= DCH0
$\overline{\text{FMNX}}$ + $\overline{\text{DCCS}}$ + $\overline{\text{DDTC}}$	= DCK0
DRL2 + DIRT	= DCKR
ZPC0	= DCN0
DCCS MEVP DDFR	= DCPB
FIRT $\overline{\text{FLPM}}$	= DCSE
FM15 DCTC	= DCT0
FM16 DCTC	= DCT1
(DR01 + DR05) $\overline{\text{FMNX}}$ FTL3	= DCTC
FWRT + FDFE	= DDFR
REWX $\overline{\text{DLLI}}$ + REAT $\overline{\text{DLLI}}$	= DDI0

INPUT/OUTPUT CONTROL UNIT

FIG. 187

LOGICAL COMBINATION SIGNALS (Cont.):

DDID + FDSE + FDSF + FDSG + FDSH	= DDIB
FDSA + FDSB + FDSC + FDSD	= DDID
DDIB + FIOR	= DDIE
FDSA	= DDPA
$\overline{\text{FDSA}}$ FDSB	= DDPB
$\overline{\text{FDSA}}$ $\overline{\text{FDSB}}$ FDSC	= DDPC
$\overline{\text{FDSA}}$ $\overline{\text{FDSB}}$ $\overline{\text{FDSC}}$ FDSD	= DDPD
$\overline{\text{DDID}}$ FDSE	= DDPE
$\overline{\text{DDID}}$ $\overline{\text{FDSE}}$ FDSF	= DDPF
$\overline{\text{DDID}}$ $\overline{\text{FDSE}}$ $\overline{\text{FDSF}}$ FDSG	= DDPG
$\overline{\text{DDID}}$ $\overline{\text{FDSE}}$ $\overline{\text{FDSF}}$ $\overline{\text{FDSG}}$ FDSH	= DDPH
FTL4 DDTE	= DDTC
DR01 FMNX + DR02	= DDTE
FE06 DEME	= DDV0
FE07 DEME	= DDV1
FE08 DEME	= DDV2
FE09 DEME	= DDV3
FE10 DEME	= DDV4
DR04 FRVQ + DIAC	= DEDX
DECH $\overline{\text{FE11}}$ $\overline{\text{FE12}}$ $\overline{\text{FE13}}$	= DEL0
DECH FE11 $\overline{\text{FE12}}$ $\overline{\text{FE13}}$	= DEL1
DECH $\overline{\text{FE11}}$ FE12 $\overline{\text{FE13}}$	= DEL2
DECH FE11 FE12 $\overline{\text{FE13}}$	= DEL3
DECH $\overline{\text{FE11}}$ $\overline{\text{FE12}}$ FE13	= DEL4
DECH FE11 $\overline{\text{FE12}}$ FE13	= DEL5

INPUT/OUTPUT CONTROL UNIT

FIG. 188

LOGICAL COMBINATION SIGNALS (Cont.):

DECH $\overline{\text{FE11}}$ FE12 FE13	= DEL6
DECH FE11 FE12 FE13	= DEL7
DEDF $\overline{\text{FGST}}$	= DEME
$\overline{\text{DNBS}}$ $\overline{\text{DPEC}}$ $\overline{\text{DPAN}}$	= DESH
FE00 DEME	= DFN0
FE01 DEME	= DFN1
FE02 DEME	= DFN2
FE03 DEME	= DFN3
FE04 DEME	= DFN4
FE05 DEME	= DFN5
DCN0 FFTT	= DFTC
DW12 FTL2 FWRS	= DFUN
FM00 DGDM	= DG00
FM01 DGDM	= DG01
FM02 DGDM	= DG02
FM03 DGDM	= DG03
FM04 DGDM	= DG04
FM05 DGDM	= DG05
FM06 DGDM	= DG06
FM07 DGDM	= DG07
FM08 DGDM	= DG08
FM09 DGDM	= DG09
FM10 DGDM	= DG10
FM11 DGDM	= DG11
FM12 DGDM	= DG12

INPUT/OUTPUT CONTROL UNIT
FIG. 189

LOGICAL COMBINATION SIGNALS (Cont.):

FM13 DGDM	= DG13
FM14 DGDM	= DG14
FM15 DGDM	= DG15
FM16 DGDM	= DG16
FM17 DGDM	= DG17
FM18 DGDM	= DG18
FM19 DGDM	= DG19
FM20 DGDM	= DG20
FM21 DGDM	= DG21
FM22 DGDM	= DG22
FM23 DGDM	= DG23
$\overline{\text{REBX}}$ ZCNO FGST	= DGCS
DDIB DRIT	= DGDM
DME0 (DJ00 DSH0 + DJ10 DSH1 + DJ20 DSH2 + DJ30 DSH3)	= DI00
DME0 (DJ01 DSH0 + DJ11 DSH1 + DJ21 DSH2 + DJ31 DSH3)	= DI01
DME0 (DJ02 DSH0 + DJ12 DSH1 + DJ22 DSH2 + DJ32 DSH3)	= DI02
DME0 (DJ03 DSH0 + DJ13 DSH1 + DJ23 DSH2 + DJ33 DSH3)	= DI03
DME0 (DJ04 DSH0 + DJ14 DSH1 + DJ24 DSH2 + DJ34 DSH3)	= DI04
DME0 (DJ05 DSH0 + DJ15 DSH1 + DJ25 DSH2 + DJ35 DSH3)	= DI05
DME1 (DJ00 DSH1 + DJ10 DSH0 + DJ30 DSH2)	= DI10
DME1 (DJ01 DSH1 + DJ11 DSH0 + DJ31 DSH2)	= DI11
DME1 (DJ02 DSH1 + DJ12 DSH0 + DJ32 DSH2)	= DI12
DME1 (DJ03 DSH1 + DJ13 DSH0 + DJ33 DSH2)	= DI13
DME1 (DJ04 DSH1 + DJ14 DSH0 + DJ34 DSH2)	= DI14
DME1 (DJ05 DSH1 + DJ15 DSH0 + DJ35 DSH2)	= DI15

INPUT/OUTPUT CONTROL UNIT

FIG. 190

LOGICAL COMBINATION SIGNALS (Cont.):

DME2 (DJ00 DSH2 + DJ10 DSH1 + DJ20 DSH0)	= DI20
DME2 (DJ01 DSH2 + DJ11 DSH1 + DJ21 DSH0)	= DI21
DME2 (DJ02 DSH2 + DJ12 DSH1 + DJ22 DSH0)	= DI22
DME2 (DJ03 DSH2 + DJ13 DSH1 + DJ23 DSH0)	= DI23
DME2 (DJ04 DSH2 + DJ14 DSH1 + DJ24 DSH0)	= DI24
DME2 (DJ05 DSH2 + DJ15 DSH1 + DJ25 DSH0)	= DI25
DME3 (DJ00 DSH3 + DJ10 DSH2 + DJ20 DSH1 + DJ30 DSH0)	= DI30
DME3 (DJ01 DSH3 + DJ11 DSH2 + DJ21 DSH1 + DJ31 DSH0 + DPEC)	= DI31
DME3 (DJ02 DSH3 + DJ12 DSH2 + DJ22 DSH1 + DJ32 DSH0)	= DI32
DME3 (DJ03 DSH3 + DJ13 DSH2 + DJ23 DSH1 + DJ33 DSH0 + DPEC)	= DI33
DME3 (DJ04 DSH3 + DJ14 DSH2 + DJ24 DSH1 + DJ34 DSH0)	= DI34
DME3 (DJ05 DSH3 + DJ15 DSH2 + DJ25 DSH1 + DJ35 DSH0)	= DI35
DMBX FMAI	= DIAC
DCBC + DNBS + DPAN + $\overline{\text{FPRI}}$ DPRC	= DIBC
DR02 FTL3 + DR01 FTL3 FMNX + DW11 FTL1 FWRS + DW12 FTL1 FWRS	= DIDS
DR13 FTL1	= DIMR
FTL3 DGEN $\overline{\text{FBSY}}$ DW01 DIMM FNXF	= DIOD
FTL4 DW03 DGEN $\overline{\text{FBSY}}$	= DIOI
FDFE + DREL + FGST $\overline{\text{DDIB}}$	= DIOZ
DPDC DEIN DGEN	= DIPR
$\overline{\text{DDIB}}$ $\overline{\text{FDFE}}$ $\overline{\text{FLPM}}$ $\overline{\text{DIBC}}$ + DDTE	= DISH
$\overline{\text{DCCS}}$ DR01 $\overline{\text{FMNX}}$	= DIWC
DK00 + DG18	= DJ00
DK01 + DG19	= DJ01
DK02 + DG20	= DJ02

INPUT/OUTPUT CONTROL UNIT

FIG. 191

LOGICAL COMBINATION SIGNALS (Cont.):

DK03 + DG21	= DJ03
DK04 + DG22	= DJ04
DK05 + DG23	= DJ05
DK10 + DG12	= DJ10
DK11 + DG13	= DJ11
DK12 + DG14	= DJ12
DK13 + DG15	= DJ13
DK14 + DG16	= DJ14
DK15 + DG17	= DJ15
DK20 + DG06 + DDV0	= DJ20
DK21 + DG07 + DDV1	= DJ21
DK22 + DG08 + DDV2	= DJ22
DK23 + DG09 + DDV3	= DJ23
DK24 + DG10 + DDV4	= DJ24
DK25 + DG11	= DJ25
DK30 + DG00 + DFN0	= DJ30
DK31 + DG01 + DFN1	= DJ31
DK32 + DG02 + DFN2	= DJ32
DK33 + DG03 + DFN3	= DJ33
DK34 + DG04 + DFN4	= DJ34
DK35 + DG05 + DFN5	= DJ35
REX0 DRDE	= DK00
REX1 DRDE	= DK01
REX2 DRDE	= DK02
REX3 DRDE	= DK03

INPUT/OUTPUT CONTROL UNIT

FIG. 192

LOGICAL COMBINATION SIGNALS (Cont.):

REX4 DRDE	= DK04
REX5 DRDE	= DK05
ZCN0 RMX0 DGCS	= DK30
ZCN0 RMX1 DGCS	= DK31
DGCS RMX2 ZCN0	= DK32
RPTS DESP ZCN0 $\overline{\text{RERX}}$	= DK34
FLPM FNBY	= DLPT
DR02 + DR04	= DMBX
DDIB $\overline{\text{DWS1}}$ $\overline{\text{DWS0}}$ + DCCS + $\overline{\text{DDIB}}$	= DME0
DMEX + DSDF	= DME1
DDIB $\overline{\text{DWS0}}$ + DMEX	= DME2
DDIB + FGST	= DME3
DDIB $\overline{\text{DWS1}}$ + DDIB DCCS	= DMEX
DR01 + DR03 + DR05	= DMRR
DR05 FTL5	= DMVR
$\overline{\text{DWCS}}$ $\overline{\text{DCCS}}$	= DNBS
JPOX ZCN0	= DPAN
DSC0 $\overline{\text{DCSE}}$	= DPC0
DSC1 $\overline{\text{DCSE}}$	= DPC1
DSC2 $\overline{\text{DCSE}}$	= DPC2
DSC3 $\overline{\text{DCSE}}$	= DPC3
DSC4 $\overline{\text{DCSE}}$	= DPC4
DSC5 $\overline{\text{DCSE}}$	= DPC5
DSC6 $\overline{\text{DCSE}}$	= DPC6
DSC7 $\overline{\text{DCSE}}$	= DPC7

INPUT/OUTPUT CONTROL UNIT
FIG. 193

LOGICAL COMBINATION SIGNALS (Cont.):

FPER DGCS	= DPEC
RERX DCNO	= DPRC
ZPI0 $\overline{FPS0}$ $\overline{FPS1}$ $\overline{FPS2}$ + ZPI1 FPS0 $\overline{FPS1}$ $\overline{FPS2}$ + ZPI2 $\overline{FPS0}$ FPS1 $\overline{FPS2}$ + ZPI3 FPS0 $\overline{FPS1}$ $\overline{FPS2}$ + ZPI4 $\overline{FPS0}$ FPS1 FPS2 + ZPI5 FPS0 $\overline{FPS1}$ FPS2 + ZPI6 FPS0 FPS1 FPS2 + ZPI7 FPS0 FPS1 FPS2	= DPSC
$\overline{FRR0}$ $\overline{FRR1}$ $\overline{FRR2}$	= DR00
FRR0 $\overline{FRR1}$ $\overline{FRR2}$ $\overline{FCSF}$	= DR01
$\overline{FRR0}$ FRR1 $\overline{FRR2}$ $\overline{FCSF}$	= DR02
FRR0 FRR1 $\overline{FRR2}$ $\overline{FCSF}$	= DR03
$\overline{FRR0}$ $\overline{FRR1}$ FRR2 $\overline{FCSF}$	= DR04
FRR0 FRR2 $\overline{FCSF}$	= DR05
DR01 + DR03	= DR13
FRCA	= DRCA
ZCN0 DIOD + ZCN0 DREL + ZCN0 DRRR + ZCN0 DEDX	= DRDA
REAT ZCN0 + DGCS	= DRDE
DIRT DDIB + DMLR	= DRDI
DRDA + DMLR	= DRDR
DR05 FTL4 $\overline{DDIB}$ + DW03 FTL4 $\overline{FBSY}$ $\overline{DBYC}$	= DREL
FWRT DWCS	= DRH0
REWX ZCN0	= DRIT
$\overline{DR05}$ + $\overline{FTL2}$	= DRL2
DRDE REPX	= DRPB
$\overline{FWRT}$ DDTC	= DRRR
DR02 $\overline{FRV0}$	= DRRV
DR01 $\overline{DDIB}$	= DRSI
DIDS DIOZ + DWRR + DDTC + DEDX	= DRWR

INPUT/OUTPUT CONTROL UNIT

FIG. 194

LOGICAL COMBINATION SIGNALS (Cont.):

SLS0 + ZDPH + DEL0	= DSC0
SLS1 + ZDPA + DEL1	= DSC1
SLS2 + ZDPB + DEL2	= DSC2
SLS3 + ZDPC + DEL3	= DSC3
SLS4 + ZDPD + DEL4	= DSC4
SLS5 + ZDPE + DEL5	= DSC5
SLS6 + ZDPF + DEL6	= DSC6
SLS7 + ZDPG + DEL7	= DSC7
DWCS $\overline{\text{FGST}}$ $\overline{\text{DDIB}}$	= DSDF
DRL2 + FMNX	= DSFT
$\overline{\text{DWS0}}$ $\overline{\text{DWS1}}$ $\overline{\text{DRH0}}$ DDIB + $\overline{\text{DDIB}}$ FGST DCCS	= DSHX
MEVP DCCS $\overline{\text{FWRT}}$ DIDS DDIB	= DSPE
$\overline{\text{DDIB}}$ FGST	= DSST
DCCS (FWRT + $\overline{\text{DDIB}}$) DIDS + FWRT DWCS DDIB DIDS + DIOD DWCS	= DSWD
DRIT DDIB	= DSWR
FE03 DREL DBYC	= DWBC
DBRS $\overline{\text{DIBC}}$	= DWCC
DSWD DME0	= DWD0
DSWD DME1	= DWD1
DSWD DME2	= DWD2
DSWD DME3	= DWD3
DIDS $\overline{\text{DDIB}}$ DWCS	= DWDF
ZCN0 DWD0	= DWDS
FCK0	= DWS0
FCK1	= DWS1
DDTE FWRT FTL4 + DDEV + DFUN + DREL	= DWST

INPUT/OUTPUT CONTROL UNIT
FIG. 195

LOGICAL COMBINATION SIGNALS (Cont.):

$\overline{FPCA}\ \overline{FPCB}$	= DBTO
$(RERX + RPTS)\ \overline{REBX}$	= DPIO
$DESH\ (DSHX + DRHO)$	= DSHO
$DESH\ DWSO\ \overline{DWS1}\ DDIB\ \overline{DRHO}$	= DSH1
$DESH\ (DWS1\ \overline{DWSO}\ DDIB\ \overline{DRHO} + DCCS\ \overline{FGST}\ \overline{DDIB}\ \overline{DW12} + DSDF)$	= DSH2
$DESH\ (DWCS\ \overline{FGST}\ \overline{DDIB}\ \overline{DSCS} + DWSO\ DWS1\ DDIB\ \overline{DRHO} + DCCS\ \overline{FGST}\ \overline{DDIB}\ DW12)$	= DSH3

INPUT/OUTPUT CONTROL UNIT
FIG. 196

United States Patent Office

3,368,205

Patented Feb. 6, 1968

3,368,205

CONTROL APPARATUS IN A DATA PROCESSING SYSTEM

Robert D. Hunter, Wayland, Mass., and Robert A. Perrine and John E. Wilhite, Phoenix, Ariz., assignors to General Electric Company, a corporation of New York

Filed Apr. 14, 1965, Ser. No. 448,196

4 Claims. (Cl. 340—172.5)

ABSTRACT OF THE DISCLOSURE

In order to provide increased efficiency in the operation of a data processor, the central control apparatus is provided with three sequencers. A first sequencer determines macro-operations for normal instruction execution; a second sequencer determines macro-operations for input/output data transfers; and a third sequencer is used in conjunction with the other two to determine the sequence of micro-operations for each macro-operation.

TABLE OF CONTENTS

TABLE OF CONTENTS—Continued

TABLES OF FREQUENT REFERENCE

GENERAL DESCRIPTION

This invention relates to data processing systems and, in particular, to control apparatus in data processing systems.

In executing a sequence of instruction words in a data processing system, it is necessary that the instructions of the program be executed in the proper sequence. Control of the sequence of instruction execution is normally effected by a counter termed a program counter. Execution of a given instruction in the program sequence to perform a corresponding data processing function normally requires performance of a large number of specific operations, for example, transferring the contents of a memory location to a register, transferring the contents of one register to another, setting certain flip-flops to the 1-state and resetting other flip-flops to the 0-state, or any

one of a multitude of other operations. Control of the sequence of these specific operations to effect execution of the instruction is normally provided by central control apparatus in the data processing system which generates appropriate control signals. These control signals are applied, in the proper time sequence, to the circuits and components of the data processing system which perform the operations.

Prior art data processing systems have normally established a fixed sequence for generation of the control signals which direct the operations required to execute the instructions. This fixed sequence is followed even though certain control signals and the corresponding operations are not employed during execution of some of the instructions. Such an arrangement is wasteful of data processing time and does not most efficiently employ the capabilities of the data processing system. Accordingly, it is desirable to provide apparatus permitting greater flexibility and efficiency in controlling the sequence of operations which effect execution of instructions in a data processing system.

It is common practice in data processing systems to provide a sequence of operations for execution of individual instructions during which a memory read operation for transferring information from memory is followed by a memory write operation for storing information into memory. Often during execution of a given instruction, it may be desirable to have several memory read operations in sequence without providing memory write operations between the memory read operations. Accordingly, it is desirable to provide control apparatus in a data processing system which permits greater flexibility in establishing the sequence of memory read and memory write operations during execution of a given instruction.

Input/output operations for transferring data between memory and a peripheral subsystem may employ one or more registers which are also used during instruction execution. Due to the limited storage capacity of the buffer registers which temporarily store data being transferred between memory and a peripheral subsystem of the data processing system, it is often necessary to interrupt the instruction execution operations being performed and to initiate operations for effecting data transfer between memory and the peripheral subsystem. For example, during a read input/output operation transferring data from the peripheral subsytem to the memory, the buffer register receiving data from the peripheral subsystem may become filled to capacity. The buffer register must then transfer the data to memory before receiving more data from the peripheral subsystem, instruction execution being interrupted to permit the data transfer. Similarly, during a write input/output operation when data is being transferred from memory to the peripheral subsystem, the buffer register may contain no more data for transmission to the peripheral subsystem. The instruction being executed must then be interrupted so that additional data can be transferred from memory to the buffer register. Accordingly, it is desirable to provide a convenient arrangement for interrupting the operations being performed to execute an instruction to permit the operations necessary for an input/output data transfer to be performed.

It is therefore an object of this invention to provide an improved control apparatus in a data processing system.

It is another object of the invention to provide control apparatus in a data processing system permitting more efficient control of operations required to execute individual instructions in the data processing system.

It is another object of the invention to provide control apparatus in a data processing system permitting more flexible control of the sequence of operations required to execute individual instructions in the data processing system.

It is a further object of this invention to provide control apparatus in a data processing system for selective sequencing of memory read and memory write operations to execute a given instruction.

It is a further object of the invention to provide sequence control apparatus in a data processing system which permits selective sequencing of read, write and read/write memory operations to most efficiently execute individual instructions in the data processing system.

It is a further object of this invention to provide an improved arrangement for permitting input/output operations to interrupt instruction execution operations.

It is a further object of the invention to provide improved sequence control apparatus in a data processing system which facilitates interrupation of instruction execution operations to effect data transfer between memory and a peripheral subsystem of the data processing system.

The foregoing objects are achieved, in the illustrated embodiment of the invention, by providing a control register designated the W-Selector Register to control the sequence of operations occurring during instruction execution. The states of the W-Selector Register are employed to provide a sequence of control signals, each control signal identifying a block of micro-operations and causing the micro-operations of the block to be performed by the central processor. The sequence of micro-operations within each block is controlled by the TL-Counter. The group of micro-operations in each block constitutes a basic macro-operation in the central processor. Each instruction capable of being executed by the data processing system requires a predetermined series of macro-operations to be performed by the central processor. The sequence of states of the W-Selector Register and thus the sequence of macro-operations performed by the central processor is determined by the instruction operation code in the I-Register and by predetermined data processing conditions occurring during the performance of the blocks of micro-operations.

Each of the blocks of micro-operations defined by the W-Selector Register includes a memory read micro-operation followed by a memory write micro-operation. Predetermined blocks of micro-operations may be split so that only the first half-sequence of the block, which includes the memory read operation, or the second half-sequence of the block, which includes the memory write operation, is performed.

A control register termed the R-Selector Register is also provided in the data processing system to control the sequence of operations occurring during an input/output data transfer. The states of the R-Selector Register are employed to provide a sequence of control signals which establishes the sequence of blocks of micro-operations to be performed during an input/output data transfer. The M-Register of the program processor is employed for temporarily storing data both during input/output data transfers and during instruction execution operations. When an input/output data transfer is required while an instruction is being executed under control of the W-Selector Register, the control signals provided by the W-Selector Register are inhibited and an appropriate sequence of control signals is provided by the R-Selector Register to control the sequence of macro-operations required to effect the input/output data transfer. Upon termination of the input/output data transfer, instruction execution operations are continued under control of the W-Selector Register. The point during instruction execution operations at which the R-Selector Register is permitted to interrupt the W-Selector Register is indicated by signal DIRS, which issues when the M-Register does not contain significant data during instruction execution operations.

This application is one of several applications covering an entire computer system. Portions of the apparatus herein disclosed are inventions of the following:

Thomas J. Beatson, David E. Keefer, Richard M. Rojko, and John E. Wilhite, as defined by the claims of their application Ser. No. 446,067, filed Apr. 6, 1965;

Richard A. Boennighausen and Byron F. Burch, Jr., Ser. No. 448.195, filed Apr. 14, 1965;

Edwin W. Herron, Robert D. Hunter, and John E. Wilhite, as defined by the claims of their application, Ser. No. 448,197, filed Apr. 14, 1965;

Frank J. Boyle and John E. Wilhite, as defined by the claims of their application, Ser. No. 448,537, filed Apr. 15, 1965;

Edwin W. Herron, Robert D. Hunter, and David E. Keefer, as defined by the claims of their application, Ser. No. 448,538, filed Apr. 15, 1965;

Edwin W. Herron, Robert D. Hunter, and David E. Keefer, as defined by the claims of their application, Ser. No. 448,539, filed Apr. 15, 1965;

Robert D. Hunter, David E. Keefer, and John E. Wilhite, as defined by the claims of their application, Ser. No. 448,540, filed Apr. 15, 1965; and

David E. Keefer, as defined by the claims of his application, Ser. No. 448,541, filed Apr. 15, 1965.

All of the above applications are assigned to the assignee of the present application.

DESCRIPTION OF DRAWINGS

The subject matter of the invention is particularly pointed out and distinctly claimed in the concluding portion of the specification. The invention, however, both as to organization and method of operation may best be understood by reference to the following description taken in connection with the accompanying drawings, in which:

FIGURE 1 is a block diagram of the data processing system to which the instant invention is applicable;

FIGURES 2*a*–2*o* are symbolic diagrams illustrating the organization of the various types of words employed in the system of FIGURE 1;

FIGURES 3*a*–3*b* comprise a circuit diagram of a clock generator employed in the system of FIGURE 1;

FIGURE 4 is a circuit diagram of a gated clock distribution driver employed in the system of FIGURE 1 and illustrates a symbol employed to represent the gated clock distribution driver;

FIGURE 5 is a circuit diagram of a gated clock amplifier employed in the system of FIGURE 1, and illustrates a symbol employed to represent the gated clock amplifier;

FIGURE 6 is a circuit diagram of a flip-flop employed in the system of FIGURE 1, and illustrates a symbol employed to represent the flip-flop;

FIGURE 7 is a circuit diagram of a pulse pedestal flip-flop employed in the system of FIGURE 1;

FIGURE 8 is a circuit diagram of a NAND-gate employed in the system of FIGURE 1, and illustrates a symbol employed to represent the NAND-gate;

FIGURE 9 is a circuit diagram of a NAND-supplement circuit employed in the system of FIGURE 1, and illustrates a symbol employed to represent the NAND-supplement circuit;

FIGURE 10 is a circuit diagram of a logic driver employed in the system of FIGURE 1, and illustrates a symbol employed to represent the logic driver;

FIGURE 11 is the circuit diagram of a full adder employed in the system of FIGURE 1, and illustrates a symbol employed to represent the full adder;

FIGURE 12 is a block diagram of the data storage elements, the data transfer paths between these elements and the major control elements of the data processing system of FIGURE 1;

FIGURE 13 is a block diagram of the circuits providing the input signals to a flip-flop of the system;

FIGURE 14 is a block diagram of a 16K memory unit;

FIGURE 15 is a symbolic diagram illustrating row selection in a memory plane of the 16K memory unit of FIGURE 14;

FIGURE 16 is a symbolic diagram of the 4K memory stacks of the 16K memory unit of FIGURE 14 and illustrates the addressing of the 4K stacks;

FIGURE 17 is a symbolic diagram of a 32K Memory employed in the data processing system of FIGURE 1 and illustrates the logic for selecting a 16K memory unit of the 32K Memory;

FIGURE 18 is a block diagram illustrating the Memory Timing and Control Logic of the 16K memory unit of FIGURE 14;

FIGURES 19*a* and 19*b* comprise a timing diagram illustrating the timing sequence of the Memory employed in the data processing system of FIGURE 1;

FIGURES 20*a*, 20*b*, 20*c*, 20*d* and 20*e* comprise a block diagram of the Central Processor illustrating the data storage elements of the Program Processor, the data transfer paths between these elements and the major control elements of the Program Processor;

FIGURE 21 is a block diagram of the Timing Clock Generator and the Start-Stop Clock Logic of the Central Processor of FIGURE 20;

FIGURE 22 is a timing diagram illustrating the timing sequence of the Timing Clock Generator and TL-Counter of the Central Processor of FIGURE 20;

FIGURE 23 is a symbolic diagram illustrating the Parity Check and Generation Logic of the Central Processor of FIGURE 20;

FIGURE 24 is a symbolic diagram illustrating the relocatable Accumulator in the Memory of the Central Processor of FIGURE 20;

FIGURES 25*a* and 25*b* are symbolic diagrams illustrating the Adder of the Central Processor of FIGURE 20;

FIGURE 26 is a symbolic diagram of the console switches of the Central Processor of FIGURE 20;

FIGURES 27–35 illustrate logical schematic diagrams of the logical circuits providing input signals to the flip-flops of the Memory;

FIGURES 36–43 illustrate logical schematic diagrams of the logical circuits providing logical combination signals, address decode and driver input signals, clock generator input signals, gated clock amplifier input signals, inhibit driver input signals and data driver input signals in the Memory;

FIGURES 44–92 illustrate logical schematic diagrams of the logical circuits providing input signals to the flip-flops of the Program Processor;

FIGURES 93–114 illustrate logical schematic diagrams of the logical circuits providing logical combination signals in the Program Processor;

FIGURES 115 and 116 illustrate logical schematic diagrams of the logical circuits providing full adder input signals in the Program Processor;

FIGURES 117 and 118 illustrate logical schematic diagrams of the logical circuits providing clock generator input signals, gated clock distribution driver input signals and gated clock amplifier input signals in the Program Processor;

FIGURES 119*a* and 119*b* comprise a timing diagram useful in explaining the operation of the system in executing the "instruction fetch" portion of the initial routine, preparatory to address development, if any, and execution of the instruction;

FIGURES 120*a* and 120*b* comprise a flow diagram useful in illustrating the operation of the system in executing the "instruction fetch" portion of the initial routine;

FIGURES 121*a*, 121*b*, 121*c* and 121*d* comprise a timing diagram useful in explaining the operation of the system in executing the "address development" portion of the initial routine, preparatory to instruction execution;

FIGURES 122*a*, 122*b*, 123, 124*a*, 124*b*, 125, 126*a* and 126*b* comprise flow diagrams useful in illustrating the

operation of the system in executing the "address development" portion of the initial routine;

FIGURES 127*a*–127*d*, 128*a*–128*c*, 129*a*–129*c*, 130*a*–130*b*, 131*a*–131*c*, 132*a*–132*h*, 133*a*–133*d*, 134*a*–134*f* and 135*a*–135*l* are timing diagrams useful in explaining the operation of the data processing system in executing certain of the programmable instructions of the system;

FIGURES 136*a*–136*i* comprise a flow diagram useful in illustrating the operation of the system in executing Instruction 22, Shift (SHG);

FIGURES 137*a*–137*e*, 138*a*–138*e*, 139*a*–139*e*, 140*a*–140*f*, 141*a*–141*f*, 142*a*–142*f*, 143*a*–143*d*, 144*a*–144*b*, 145*a*–145*d*, 146*a*–146*d*, 147*a*–147*c*, 148*a*–148*b*, 149, 150*a*–150*c*, 151, 152*a*–152*b*, 153*a*–153*b* and 154*a*–154*c* are timing diagrams useful in explaining the operation of the data processing system in executing certain programmable instructions of the system;

FIGURE 155 is a flow diagram useful in illustrating the operation of the system in executing Instruction 16, Branch on Count (BRC);

FIGURES 156*a*–156*e*, 157, 158*a*–158*b*, 159*a*–159*r* and 160*a*–160*q* are timing diagrams useful in explaining the operation of the data processing system in executing certain programmable instructions of the system;

FIGURES 161*a*–161*d* is a flow diagram useful in illustrating the operation of the system in executing Instruction 26, Variable Length Divide (VLD);

FIGURE 162 is a table illustrating the "edit mode" portion of the operation of the system in executing Instruction 05, Edit (EDT);

FIGURES 163*a*–163*l* comprise a timing diagram useful in explaining the operation of the system in executing Instruction 05, Edit (EDT);

FIGURES 164*a*–164*e* comprise a flow diagram useful in illustrating the operation of the system in executing Instruction 05, Edit (EDT);

FIGURES 165*a*–165*c*, 166 and 167*a*–167*f* are timing diagrams useful in explaining the operation of the data processing system in executing certain programmable instructions of the system;

FIGURE 168 is a flow diagram useful in illustrating the operation of the system in executing Instruction 07, General (GEN);

FIGURE 169 is a table illustrating the alert conditions which occur in the Central Processor and the events resulting from these alert conditions;

FIGURE 170 is a block diagram of the Input/Output Control Unit of the data processing system of FIGURE 1;

FIGURE 171 is a block diagram illustrating data transfer in the Input/Output Control Unit of FIGURE 170;

FIGURE 172 is a block diagram illustrating the logic which controls data interrupt operations in the Input/Output Control Unit of FIGURE 170;

FIGURES 173*a*, 173*b* and 173*c* comprise a flow diagram useful in illustrating the operation of the system during an R-sequence;

FIGURE 174 is a block diagram illustrating the logic which controls program interrupt operations in the Input/Output Control Unit of FIGURE 170;

FIGURE 175 is a timing diagram useful in explaining the operation of the system in executing the W00 block of the initial routine during a program interrupt;

FIGURE 176 is a timing diagram useful in explaining the operation of the system in executing the W01 block of the initial routine during a program interrupt;

FIGURE 177 is a flow diagram useful in illustrating the operation of the system during a program interrupt;

FIGURES 178–186 illustrate logical schematic diagrams of the logical circuits providing input signals to the flip-flops of the Input/Output Control Unit; and

FIGURES 187–196 illustrate logical schematic diagrams of the logical circuits providing logical combination signals in the Input/Output Control Unit.

DATA PROCESSING SYSTEM—GENERAL

With reference to FIG. 1, the illustrated data processing system comprises a Central Processor and a plurality of peripheral subsystems. The major units of the Central Processor are Memory **10**, Arithmetic Unit **11**, Central Processor Control Unit **12**, Input/Output Control Unit **13** and Console **14**. In the description, the term Program Processor is applied to the portion of the Central Processor consisting of the Arithmetic Unit **11**, the Central Processor Control Unit **12** and the Console **14**. The peripheral subsystems which are used with the Central Processor to process data include Typewriter **15** which is associated with Console **14**, Document Handler **16**, Card Reader **17**, Card Punch **18**, Perforated Tape Reader/Punch Unit **19**, Printer **20**, Magnetic Tape Controller **21** and Disc Storage Controller **22**. Magnetic Tape Controller **21** can control a plurality of Magnetic Tape Units **23** and Disc Storage Controller **22** can control a plurality of Disc Storage Units **24**. Any combination of these peripheral subsystems may be employed with the Central Processor to perform a desired data processing function. The lines interconnecting the various components illustrated in FIG. 1 represent symbolically paths of data and control signals.

The Central Processor responds to a plurality of distinct instructions which are supplied in the sequential order necessary to perform a particular data processing operation. Memory **10** stores data words which are to be processed, data words which are the result of processing, instruction words and auxiliary words for addressing and control. The Accumulator of the Central Processor is also located in Memory **10**.

Arithmetic Unit **11** performs binary and decimal arithmetic operations. Central Processor Control Unit **12** controls the sequence of events required for instruction execution in the Central Processor. Arithmetic Unit **11** and Central Processor Control Unit **12**, which together comprise the Program Processor, contain the logical elements necessary to access Memory **10** and to perform all operations required for instruction execution. Arithmetic Unit **11** and Central Processor Control Unit **12** communicate with Memory **10** to obtain instruction words, auxiliary words, data words on which operations are to be performed and control signals for synchronizing the Program Processor timing with operations in Memory **10**.

Input/Output Control Unit **13** provides for orderly sequencing of data transfers between Memory **10** and the plurality of peripheral subsystems and serves to transmit instructions from the Central Processor to the peripheral subsystems. The Input/Output Control Unit also monitors peripheral subsystem operating conditions. Communication between the Central Processor and the various peripheral subsystems occurs through a plurality of channels which are included in the Input/Output Control Unit **13**, each channel being associated with one peripheral subsystem.

Console **14**, in conjunction with Typewriter **15**, permits operator control and communication with the Central Processor. The Console includes switches for controlling Central Processor power and program loading, for initiating and halting Central Processor operation and for resetting alert conditions.

DATA REPRESENTATION

The data processing system of FIG. 1 processes data items represented by the binary code. In the binary code, each element of information is represented by a binary digit, sometimes termed a "bit," each binary digit being either a 1 or a 0. In the data processing system here described, a binary 1 is represented by a positive electrical signal in the order of +3.8 volts and a binary 0 is represented by an electrical signal in the order of +0.2 volt.

The fundamental unit of data employed in processing and communication in the system is the word. A word comprises 24 binary digits. A 25th binary digit for parity

checking is employed when the word appears in Memory 10. The first binary digit of the data word is termed the most-significant digit (MSD) and the last binary digit is termed the least-significant digit (LSD) of the word. The binary digits between the MSD and the LSD are accorded successively decreasing orders of significance.

Three general categories of words are employed in the data processing system of FIG. 1: (1) data words, (2) instruction words, (3) auxiliary words for addressing and control. Two types of data words are processed, viz. alphanumeric data words and binary data words. Auxiliary words are of the following types:

(a) Fixed Location Index Word.
(b) P-Sequence AMS Word.
(c) Remote AMS Word.
(d) Indirect Address Word.
(e) Indirect AMS Word.
(f) P-Sequence SAS Word.
(g) Remote SAS Word.
(h) Control Word.
(i) Status Word.
(j) List Pointer Word.
(k) Data Control Word.

The organization of each category and type of word is illustrated in FIGS. 2*a*–2*o*.

Data words

An alphanumeric data word represents four characters, each character comprising six binary digits, as shown in FIG. 2*a*. The six binary digits of each character of an alphanumeric data word include two zone bits BA and four binary-coded-decimal (BCD) bits. The four characters of an alphanumeric data word are designated characters 0–3 from left to right, as illustrated. Character 0 is accorded the most significance and character 3 the least significance. Characters 1 and 2 are accorded successively decreasing orders of significance in accordance with their positions within the alphanumeric data word.

The six binary digits of each character of an alphanumeric data word permit 64 unique bit combinations which are employed to represent the decimal numerals 0–9, the letters of the alphabet A–Z, and certain other special symbols such as punctuation marks, etc. The following table illustrates the various combinations of zone and BCD bits and the corresponding alphanumeric characters.

ALPHANUMERIC CHARACTER CODE

Zone Bits BA	BCD Bits	Character	Zone Bits BA	BCD Bits	Character
00	0000	0	10	0000	↑
00	0001	1	10	0001	J
00	0010	2	10	0010	K
00	0011	3	10	0011	L
00	0100	4	10	0100	M
00	0101	5	10	0101	N
00	0110	6	10	0110	O
00	0111	7	10	0111	P
00	1000	8	10	1000	Q
00	1001	9	10	1001	R
00	1010	~	10	1010	—(dash or minus)
00	1011	#	10	1011	$
00	1100	@	10	1100	*
00	1101	:	10	1101	)
00	1110	>	10	1110	;
00	1111	≠	10	1111	'(apostrophe)
01	0000	Δ(space)	11	0000	+
01	0001	A	11	0001	/
01	0010	B	11	0010	S
01	0011	C	11	0011	T
01	0100	D	11	0100	U
01	0101	E	11	0101	V
01	0110	F	11	0110	W
01	0111	G	11	0111	X
01	1000	H	11	1000	Y
01	1001	I	11	1001	Z
01	1010	⊞	11	1010	→
01	1011	.(period)	11	1011	,(comma)
01	1100	∥	11	1100	%
01	1101	(	11	1101	=
01	1110	<	11	1110	"
01	1111	↓	11	1111	∧

In a character representing a decimal numeral, the zone bits BA are each a binary 0 and the four BCD bits represent the decimal numeral. For example, the character comprising the binary digits 000100 represents the decimal numeral 4. In decimal arithmetic operations, algebraic sign information is contained in the zone bits of character 3, the least-significant character of the alphanumeric data word. The zone bits of characters 0–2 are ignored. The zone bits of character 3 indicate the sign (±) of the entire alphanumeric data word. The zone bits of character 3 are interpreted as follows:

BA	Sign
00	+
01	+
10	−
11	+

Thus, a minus alphanumeric data word is indicated by the binary configuration 10 in the zone bits of character 3, any other configuration of zone bits being interpreted as plus. In arithmetic results, the zone bits of character 3 will be set to 00 if the result is plus.

If a decimal data field is longer than one alphanumeric data word, the sign of the entire field is contained in character 3 of the least-significant word of the field. Zone bits of all other characters of the field are ignored.

A binary data word represents a single number, the entire twenty-four bits being treated as a single unit of information without separation into characters as in alphanumeric data words. The organization of a binary data word is illustrated in FIG. 2*b*. Binary digit 23 is the MSD and binary digit 0 is the LSD of the binary data word. Binary digit 0 of the binary data word thus represents 2^0 and is equal to the decimal number 1, if binary digit 0 is a binary 1. Binary digit 1 represents 2^1 and is equal to the decimal number 2, if binary digit 1 is a binary 1. Binary digit 5, represents 2^5 and is equal to the decimal number 32, if binary digit 5 is a binary 1. Thus, in general, the (i)th binary digit of the binary data word represents 2^i, if the (i)th binary digit is a binary 1. The (i)th binary digit represents 0 if the digit is a binary 0. The 24 bit binary data word thus provides for decimal number values ranging from 0–16,777,215. Binary words are automatically treated as positive quantities and no provision is made for indicating sign information.

Instruction words

Operations for effecting data processing are performed in the system under the control of a series of instruction words which are stored in Memory 10 and executed one at a time. The sequence in which instructions are executed is called the P-sequence or program sequence and is controlled by a counter.

The organization of an instruction word is illustrated in FIG. 2*c*. The operation code of the instruction word (bits 18–23) identifies the operation or program step to be performed and indicates whether the instruction is a single-address or a two-address instruction. The address field (bits 0–14) is a numerical representation identifying a tentative, effective or final location in Memory 10 from which a data item is to be extracted for processing or in which a processed data item is to be stored, in the instruction to be executed. If the address is tentative, address development is performed to make the address final, so that the instruction can be executed. Each storage location in Memory is identified by a different address. The address control field or ACF (bits 15–17), in conjunction with the operation code, determines the type of address development, if any, to be performed. In certain instructions, the instruction address field contains information other than a memory address.

Auxiliary words

The data processing system permits extensive address manipulation, modification and development to materially reduce total processing time and program memory requirements. The instruction repertoire of the data processing system includes both single-address and two-address instructions. Single-address instructions specify a single operand in Memory **10**, while two-address instructions involve two operands. During instruction processing, the Central Processor follows this general sequence:

(1) The instruction word is obtained from memory.
(2) Any address development specified by the address control field of the instruction word is performed.
(3) If the operation code identifies the instruction as a two-address instruction, the second address is developed.
(4) The instruction is executed.

The address development accomplished in step 2 of the above sequence and the development of the second address accomplished in step 3 of the above sequence are effected in the Central Processor by the use of one or more auxiliary words. Auxiliary words are stored in Memory **10** in the same manner as data words and instruction words.

The Fixed Location Index Word (FLIW) is a type of auxiliary word which is used in addressing and address development. The organization of the Fixed Location Index Word is illustrated in FIG. 2*d*. In address development employing a Fixed Location Index Word, the address field (bits 0–14) of the Fixed Location Index Word is added to the address field of the instruction word. The instruction is then executed using the modified address. If the operation code of the instruction identifies it as a two-address instruction, the location of the Fixed Location Index Word will be used on the second address of the instruction.

In addition to the development of an instruction address by means of a Fixed Location Index Word, address development is accomplished in the Central Processor by employing a P-Sequence AMS Word. The organization of a P-Sequence AMS Word is illustrated in FIG. 2*e*. A P-Sequence AMS Word is a word in the P-sequence whose purpose is to modify the address of the instruction which it follows in the program sequence.

Bits 18 and 19 of the P-Sequence AMS Word define the class of the word as follows:

Bits 18, 19	Class
00	Index
01	Index Pointer
10	Index Link

If the P-Sequence AMS Word is identified as an index, the address field (bits 0–14) is added to the address field or tentative address of the instruction to be executed. If the class bits of the P-Sequence AMS Word identify it as an index pointer, its address field specifies the location, outside the P-sequence, of an indexing quantity which is to be added to the address field or tentative address of the instruction. If the P-Sequence AMS Word is classed as an index link, its address field specifies a location outside the P-sequence which contains a Remote AMS Word.

Bit position 21 of the P-Sequence AMS Word is used to control address development by means of indirect addressing. A binary 1 in bit position 21 of the P-Sequence AMS Word specifies that the address being developed is an effective address which will be employed, not as the address of the operand, but to obtain an Indirect Address Word from Memory **10**. The address field of the Indirect Address Word is then used as a tentative address or as a final address in executing the instruction.

Bit positions 15–17 of the P-Sequence AMS Word are employed to specify continuation or non-continuation of indexing. If bit positions 15–17 contain binary 1's, indexing is continued. After development of the address to the extent called for by the present P-Sequence AMS Word, another P-Sequence AMS Word is obtained from Memory and employed to further develop the address field of the instruction. Any other binary configuration in bit positions 15–17 indicates that another P-Sequence AMS Word is not to be used for further address development. The P-Sequence AMS Word is the only AMS word that is examined for continuation of indexing. Bit positions 20, 22 and 23 are ignored.

A Remote AMS Word is an AMS word outside the P-sequence. A Remote AMS Word is addressed by a P-Sequence AMS Word which is an index link, by another Remote AMS Word which is an index link or by an Indirect AMS Word which is an index link. The Remote AMS Word organization, shown in FIG. 2*f*, is similar to that of the P-Sequence AMS Word except that bit positions 15–17 and 20–23 are ignored, since a Remote AMS Word is only examined for class. If the class bits of a Remote AMS Word classify it as an index, its address field is added to the tentative address to develop an effective address, if indirect addressing is called for, or a final address. If the Remote AMS Word is an index pointer, it is used to derive, from Memory **10**, a word containing an indexing quantity. If the class bits indicate the Remote AMS Word to be an index link, the address field of the Remote AMS Word is used to address another Remote AMS Word.

A P-Sequence AMS Word, as described above, causes an index to be obtained for modifying the address field of the instruction word. The P-Sequence AMS Word is then examined in bit position 21 where the presence of a binary 1 indicates that the developed address of the instruction word is an effective address which is to be employed to obtain an Indirect Address Word. The indirect address specifies a location which contains an Indirect Address Word. The organization of the Indirect Address Word, shown in FIG. 2*g*, is the same as that of a P-Sequence AMS Word except that the binary digits in bit positions 18–23 are ignored. The address field of the Indirect Address Word replaces the address previously developed for the instruction being executed. The indirect addressing system and apparatus described herein includes the invention of Messrs. Frank J. Boyle and John E. Wilhite.

The address control field of the Indirect Address Word can call for further address development by means of an Indirect AMS Word. The Indirect AMS Word is a special form of Remote AMS Word and the organization, shown in FIG. 2*h*, is the same except for bit position 21, which may be employed to specify further indirect addressing. The Indirect AMS Word may be classified as an index, an index pointer or an index link. As an index link, it is employed to obtain a Remote AMS Word to further develop the address. The indirect addressing sequence continues until either the ACF of an Indirect Address Word does not call for an Indirect AMS Word or until an Indirect AMS Word does not specify further indirect addressing. At the conclusion of an indirect addressing sequence, control returns to the P-Sequence AMS Word which initiated the indirect address sequence.

In those instructions which require a second address (other than that of a Fixed Location Index Word), a second address sequence (SAS) is initiated following the development, if any, of the first address. The second address sequence is initiated by extracting the next word in the P-sequence from Memory. This word is a P-Sequence SAS Word and is organized as illustrated in FIG. 2*i*. The P-Sequence SAS Word is examined for class only, as defined by the binary digits in bit positions 18 and 19. Bit

positions 15–17 and 20–23 are ignored. The class of the P-Sequence SAS Word is defined as follows:

Bits 19, 18	Class
00	Operand
01	Operand Pointer
10 or 11	Operand Link.

If the P-Sequence SAS Word is defined by the class bits as an operand, the second address of the instruction is the address of the P-Sequence SAS Word. An operand terminates the second address sequence by its self-identification. Classification of the P-Sequence SAS Word as an operand pointer indicates that the address field of the operand pointer contains the second address of the two-address instruction. If the class bits of the P-Sequence SAS Word identify it as an operand link, the address field is used to obtain a Remote SAS Word which is used to further develop the second address. Since the ACF bit positions 15–17 of the P-Sequence SAS Word are ignored, only one P-Sequence SAS Word is employed to develop the second address.

The organization of a Remote SAS Word, shown in FIG. 2*j*, is identical to that of a Remote AMS Word. The Remote SAS Word is a SAS word outside the P-sequence and is examined for class only. Any desired number of Remote SAS Words may be employed to develop the second address of a two-address instruction.

The above-described auxiliary words in the data processing system are thus employed, in conjunction with the address fields of instruction words, to develop the addresses of the operands upon which the operations called for by the operation codes of the instruction words are performed. Thus, each instruction employs one instruction word which may refer to a Fixed Location Index Word or which may be followed by one or more P-Sequence AMS Words, and when appropriate, by one P-Sequence SAS Word. Each P-Sequence AMS Word may refer to an Indirect Address Word, which may be followed by an Indirect AMS Word. Each Indirect AMS Word may refer to another Indirect Address Word. P-Sequence AMS and P-Sequence SAS Words can also refer to Remote AMS and Remote SAS Words respectively located in Memory **10** but outside the P-sequence in order to determine the address modifier or the second address of the instruction.

FIG. 2*k* illustrates another type of auxiliary word, viz. a control word employed in executing the instruction described in the section "Instruction 06: Move (MOV)." Bits 0–14 of the control word contain an address while bits 15–23 contain a count. FIG. 21 also illustrates a control word which is used during execution of the instruction described in the section "Instruction 16: Branch on Count (BRC)." Bit positions 0–8 and bit positions 15–23 each contain a count. Bit positions 9–14 of the control word are not used.

Another type of auxiliary word, a status word, is employed in executing the instruction described in the section "Instruction 67: Central Processor Operations (CPO)." As illustrated in FIG. 2*m*, bit positions 0–5 and 19–21 of the status word contain information on the states of predetermined indicator flip-flops. Bit positions 6–18, 22 and 23 of the status word are not used.

Control of input/output operations in the data processing system is effected by the auxiliary words illustrated in FIGS. 2*n* and 2*o*. The List Pointer Word of FIG. 2*n* contains an address field in bit positions 0–14 and a count in bit positions 15–23. Bit positions 0–14 of the Data Control Word of FIG. 2*o* contain an address field while bit positions 15–23 contain a count. The List Pointer Word and Data Control Word provide memory addresses to be used for data transfer between Memory and a peripheral subsystem.

SYSTEM CIRCUIT ELEMENTS

Circuits useful as elements of the data processing system of FIG. 1 will now be described. The system will function with these elements or with other similar elements well known in the art; therefore, this invention is not to be considered as limited to the employment of the specific circuits shown and described.

The following circuits find general employment in the system: clock generator, gated clock distribution driver, gated clock amplifier, flip-flop, NAND-gate, NAND-supplement, logic driver, AND-gate, OR-gate and full adder. Illustrated with the respective drawings of the circuit elements are symbols representing these circuits. These symbols are employed in the description of the invention to follow.

Clock Generator

The Clock Generator of FIG. 3 provides the basic clock pulse employed in the data processing system. Independent clock generators are provided in the Memory and in the Program Processor of the system. The Clock Generator of the Program Processor provides clock pulses at a repetition rate of approximately 2.85 megacycles. The Clock Generator of the Memory provides memory clock pulses at a repetition rate of approximately 5.7 megacycles. The generation of clock pulses by the Clock Generator is initiated by Start-Stop Clock Logic and maintained by a feedback loop within the Clock Generator. Termination of clock pulse generation by the Clock Generator is effected by the Start-Stop Clock Logic through interruption of the feedback loop.

The Clock Generator comprises input gates, a pulse shaper and a feedback delay circuit. Input terminals **30**, **31**, **32** and **33** are connected to diode AND-gate **34**. The signals applied to terminals **30–33** are derived from the Start-Stop Clock Logic of the Program Processor or the Memory and serve to initiate pulse generation by the Clock Generator. With AND-gate **34** enabled, the gate output signal is applied to amplifying transistor **36** and to a differentiating circuit comprising capacitor **37** and resistor **38**. The resulting pulse is applied through diode OR-gate **40** to amplifying transistor **41** of the pulse shaper. A second input to diode OR-gate **40** and transistor **41** is provided, after initiation of central clock pulse generation, by feedback pulses from the feedback delay circuit on line **42**.

The pulse shaper portion of the circuit controls the pulse width, the pulse amplitude and the rise time. The amplified pulse output of transistor **41** is applied to the base of transistor **45** and simultaneously applied to the input of delay line **46**. The pulse in the delay line is reflected from the end of the delay line and returns to cancel the signal at the base of transistor **45**. Delay line **46**, which may be of any suitable type, thus determines the duration of the input signal at the base of transistor **45** and the approximate width of the output pulse of the Clock Generator, the pulse width being two times the delay of delay line **46**.

Transistors **45** and **47** form a current mode switch which controls the direction of current flow in line **50**. Transistor **52** in conjunction with resistor **53** comprises a constant current source for the current mode switch. A reference potential, established by voltage divider **49**, is applied to the base of transistor **47**. This reference potential serves as a fine control on the width of the control clock pulses produced by the Clock Generator.

In the current mode switch, the current provided by the constant current source is switched through either transistor **45** or transistor **47**, depending upon the potential difference between the bases of transistors **45** and **47**. If the base potential of transistor **45** is below the reference potential applied to the base of transistor **47**, transistor **47** conducts. Conversely, if the base potential of transistor **45** is greater than the reference potential applied to the base of transistor **47**, transistor **45** conducts.

The direction of current flow through the emitter-collector junctions of transistors 45 and 47 is indicated by arrows in FIG. 3.

The network including transistors 54 and 55 changes the effective direction of current flow through transistor 45, as seen by line 50. Resistor 56 controls the rise time of the pulse. The direction of current flow through the emitter-collector junction of transistor 55 is indicated by an arrow in FIG. 3. Thus, if transistor 45 conducts, a constant current flows into line 50 whereas, if transistor 47 conducts, a constant current flows out of line 50.

Line 50 connects the current mode switch to a capacitive load comprising the base-collector capacities of transistors 57 and 58 and the base-emitter capacities of transistors 60 and 61. Transistors 57 and 58 serve as output amplifiers, transistor 57 amplifying a negative current pulse on line 50 and transistor 58 amplifying a positive current pulse on line 50. The amplifier output clock pulse appears at terminal 63. An approximate waveshape of the output signal from terminal 63 for the Clock Generator of the Program Processor is illustrated in the waveform immediately below output terminal 63.

The network, including transistors 60 and 61, serves to control the central clock pulse amplitude by clamping the voltage at the bases of transistors 57 and 58. The voltage dividing network 64 permits adjustment of the height of the pulse top relative to the reference potential established at the base of transistor 47.

Transistor 41 must remain conductive until the pulse at the base of transistor 45 has been formed through the action of delay line 46. A feedback network comprising line 65 and amplifying transistor 66 renders transistor 45 conductive until the clock pulse has been generated.

After clock pulse generation is initiated in the pulse shaper, the output clock pulse is circulated in a delay line and fed back to the input of the pulse shaper, causing the Clock Generator to continue generation of clock pulses. The pulse repetition rate of the Clock Generator is determined by the time delay in the delay line of the feedback delay circuit. The feedback delay circuit includes an AND-gate 69 having a plurality of input terminals, one input to AND-gate 69 being the central clock pulse generated in the clock pulse shaper. One or more logic signals are also applied to AND-gate 69, the logic signals being used to stop the generation of clock pulses at appropriate times. Transistor 70 serves to amplify the clock pulse. Delay line 71, which may be of any suitable type, delays the pulse approximately 320 nanoseconds in the Clock Generator of the Program Processor and approximately 145 nanoseconds in the Clock Generator of the Memory before applying it to amplifying transistor 72. Voltage divider 73 permits adjustment of the delay time of delay line 71. The delay pulse is divided through line 42 to OR-gate 40 for application to the pulse shaper.

The Clock Generator circuit permits precise control of pulse width, pulse amplitude and rise time. The generated pulse is made symmetrical with respect to a reference potential and pulse width is adjusted by adjusting the reference potential. The circuit is self-compensating with regard to power supply voltage variations, temperature, etc. The start-stop signals at the input terminals of the Clock Generator and the signals at the output terminal, as designated in the Program Processor and Memory, are identified in FIG. 3. In FIG. 3, an asterisk (*) is used to identify the signals in the Clock Generator of the Program Processor while the symbol (#) is used to identify the signals in the Clock Generator of the Memory.

The symbol 75, shown in FIG. 3, is employed to represent the Clock Generator in the description of the Central Processor. The lines entering the right side of symbol represent the input signals to the Clock Generator while the line leaving the right side indicates the control clock pulse output.

Gated Clock Distribution Driver

The Gated Clock Distribution Driver of FIG. 4 amplifies the clock pulses provided by the Clock Generator of the Program Processor and serves as an intermediate driver between the Clock Generator and the Gated Clock Amplifiers. The Gated Clock Distribution Driver comprises an AND-gate and two stages of amplification. An output pulse is produced by the Gated Clock Distribution Driver whenever a clock pulse is provided by the Clock Generator, if the AND-gate is enabled by an appropriate logic input signal.

The AND-gate portion of the Gated Clock Distribution Driver includes transistors 77 and 78 and diodes 80, 81 and 82. Resistors 84 and 85, in conjunction with inductors 86 and 87 respectively and the associated voltage sources, serve as current sources for the AND-gate. The input to the base of transistor 77 comprises the clock pulses at terminal 76 produced by the Clock Generator. A logic input signal at terminal 79, which brackets the clock pulse input, is applied to the base of transistor 78. The simultaneous application of the logic input and clock input signals to the AND-gate causes the clock pulse to appear on line 89 for application to amplifying transistor 90. Transistor 91 provides a second stage of amplification. Resistors 93 and 94, in conjunction with inductors 95 and 96 and with the associated potential sources, serve as current sources for the amplifying transistors. Diodes 97 and 98 function as a voltage clamp to prevent transistor 91 from operating in the region of saturation. The output drive pulses appear at terminals 99.

The symbol 100, shown in FIG. 4, is employed to represent the Gated Clock Distribution Driver. The clock input and logic input signals are applied to terminals 101 and 102 respectively and the amplified drive pulses are derived at terminal 103.

Gated Clock Amplifier

The Gated Clock Amplifier of FIG. 5 amplifies the drive pulse output of the Gated Clock Distribution Driver and serves as a final driver between the Gated Clock Distribution Driver and the flip-flop loading. The Gated Clock Amplifier is similar to a Gated Clock Distribution Driver but includes a transmission line and matching section and a third stage of amplification. The third stage of amplification renders the waveform of the clock signal output of the Gated Clock Amplifier independent of the load.

The drive pulses produced by the Gated Clock Distribution Driver are transmitted to the Gated Clock Amplifier via a coaxial cable 107. The cable is terminated in its characteristic impedance provided by resistors 108 and 109.

The AND-gate portion of the Gated Clock Amplifier includes transistors 111 and 112 and diodes 113, 114 and 115. Resistors 116 and 117, in conjunction with conductors 118 and 119 respectively and the associated voltage sources, serve as current sources for the AND-gate. The drive pulse output of the Gated Clock Distribution Driver is applied to the base of transistor 111. A logic input signal which brackets the drive pulse is applied to the base of transistor 112. The simultaneous application of the logic input and drive input signals to the AND-gate causes the drive pulse to appear on line 121 for application to amplifying transistor 123.

Transistor 124 provides the second stage of amplification. Resistors 125 and 126, in conjunction with inductors 127 and 128 and the associated potential sources, serve as current sources for the amplifying transistors. Diodes 130 and 131 function as a voltage clamp to prevent transistor 124 from operating in the region of saturation. Transistor 124 is coupled through capacitor 134 to the bases of transistors 136 and 137. Transistors 136 and 137 form a complementary pair, transistor 136 amplifying the positive-going portion of the pulse and transistor 137 amplifying

the negative-going portion of the pulse. Inductor 139, resistor 140 and the associated voltage source form a current source.

The clock signals appearing at output terminals 142 serve as control and working clock signals in the Central Processor. These control and working clock signals are employed to drive flip-flops in the Central Processor.

The symbol 143, shown in FIG. 5, represents a Gated Clock Amplifier. The drive and logic input signals are applied to terminals 144 and 145 respectively while the clock signal output is derived from terminal 146.

Flip-flop

The flip-flop of FIG. 6 provides temporary storage of a data word bit or provides temporary storage of a control signal. Generally, when a flip-flop is employed to store a data bit, it comprises one of an array of flip-flops which is called a register. For example, in a register adapted to provide temporary storage for a complete data word, 24 flip-flops are employed, one for each bit of a data word.

The flip-flop circuit is adapted to operate in either one of two stable states and to transfer from the stable state in which it is operating to the other stable state upon application of a trigger signal thereto. In one state of operation (the 1-state) the flip-flop represents a binary 1 and in the other state of operation (the 0-state) the flip-flop represents a binary 0. When the flip-flop is in the 1-state, it is said to be "set." When the flip-flop is in the 0-state, it is said to be "reset."

The flip-flop circuit, shown in FIG. 6, includes a pair of collector-coupled transistor amplifiers comprising transistors 154 and 155 respectively. The coupled transistor amplifier pair is connected to a pair of grounded-emitter transistor output amplifiers comprising transistors 158 and 159 respectively. The output of the amplifier including transistor 158 represents the 1-state of the flip-flop while the output of the output amplifier comprising transistor 159 represents the 0-state of the flip-flop.

The flip-flop is adapted to receive two input signals, a trigger signal for triggering the flip-flop to its 1-state and a trigger signal for triggering the flip-flop to its 0-state. The trigger signal for triggering the flip-flop to its 1-state is applied to "set" input terminal 160. The trigger signal for triggering the flip-flop to its 0-state is applied to "reset" input terminal 161. The trigger signals are supplied by logical gating structures, employing conventional diode AND-gates and OR-gates, which are enabled by clock and logic input signals. Typical diode AND-gate and OR-gate circuits, and their corresponding symbols, are shown in FIG. 11 as part of the input gating structure of the full adder circuit.

The flip-flop output signal representing the 1-state of the flip-flop is delivered at output terminal 164 and the flip-flop output signal representing the 0-state of the flip-flop is delivered at output terminal 165. Thus, when the flip-flop is in the 1-state, the signal at output terminal 164 is high and the signal at output terminal 165 is low. When the flip-flop is in the 0-state, the signal at output terminal 164 is low and the signal at output terminal 165 is high.

In both of the stable states of the flip-flop, one of the transistors 154 and 155 is conducting and the other is non-conducting. Assuming that transistor 154 is conducting, current flows from the +12 volt source through resistor 167, the emitter-collector junction of transistor 154 and resistor 168 to the −12 volt source. The potential at the collector of transistor 154 is coupled through inductor 169 to the base of output amplifier transistor 158 to render transistor 158 conductive. Base current flows through inductor 169 and into the base of transistor 158. Diode 170 is back-biased and does not conduct.

The potential at the collector of transistor 154 is coupled through resistor 171 to the base of transistor 155 and maintains transistor 155 in a state of non-conduction. Transistor 159 is also non-conductive. Current flows from ground through diode 172, inductor 173 and resistor 174 to the −12 volt source. In the condition as described, the flip-flop is in its 0-state with the signals at terminals 164 and 165 being low and high respectively.

When the clock and logic input signals to the logic gates connected to set input terminal 160 provides a trigger signal at terminal 160, the base of transistor 154 experiences a positive voltage swing, reducing the conductivity of the transistor. The resulting decrease in current flow through inductor 169 induces a potential across the inductor terminals, the collector of transistor 154 experiencing a negative voltage swing as a result of this induced potential. This negative voltage swing at the collector of transistor 154 biases transistor 155 into conduction, thereby diverting part of the current flowing through resistor 167 into the emitter-collector junction of transistor 155 and inductor 173. Inductor 173 opposes the change in current flow by an induced voltage across its terminals which drives the collector of transistor 155 more positive. The more positive potential at the collector of transistor 155 is coupled through resistor 175 to the base of transistor 154 further reducing conductivity. Inductors 169 and 173 therefore introduce a regenerative action into the flip-flop, increasing the speed with which transistors 154 and 155 change between a conductive and non-conductive state.

With transistor 154 non-conductive and transistor 155 conductive, output amplifying transistor 158 is rendered non-conductive while output amplifying transistor 159 is rendered conductive. The signal at output terminal 164 is therefore high while the signal at output terminal 165 is low, indicating that the flip-flop is in its 1-state.

The output signals at terminal 164 and 165 of the flip-flop circuit of FIG. 6 are delayed with respect to the clock and logic input signals which produce the trigger signals at terminal 160 or 161. The delay results from the storage of currents in inductors 169 and 173, since output amplifying transistors 158 and 159 do not change state until reversal of current flow in inductors 169 and 173 is effected. The switching of transistors 158 and 159 is then delayed with respect to the switching of transistors 154 and 155. Flip-flop circuits of the type illustrated in FIG. 6 are employed in the system as general purpose flip-flops to provide synchronous logic.

Flip-flops are identified in accordance with the function they perform. For example, a typical flip-flop employed in the Central Processor for control is designated the RCK flip-flop. The RCK designation stands for "Run Clock" and the RCK flip-flop, when in the 1-state, permits the Clock Generator to generate working clock signals. A typical flip-flop employed in the Central Processor for temporary storage of data is the A14 flip-flop. The symbol 177, shown in FIG. 6, is employed to represent a flip-flop. Symbol 177, in this instance, represents the A14 flip-flop.

The A14 flip-flop is employed to temporarily store the 15th bit in the A-Register. The two leads entering the left-hand side of the flip-flop symbol represent the two input terminals. The upper input lead receives the signal which triggers the flip-flop to its 1-state while the lower lead receives the signal which triggers the flip-flop to its 0-state. The notation $FA14=FE14\ QEAX$ indicates the logical gate structure employed to generate the "set" trigger signal. Similarly, the notation $\overline{FA14}=DCLA$ indicates the logical gate structure employed to generate the "reset" trigger signal for the A14 flip-flop.

The two leads leaving the right-hand side of the symbol 177 represent the two output terminals. The upper output lead delivers the 1-output signal of the A14 flip-flop, as indicated by the notation FA14. The lower output lead delivers the 0-output signal of the A14 flip-flop, as indicated by the notation $\overline{FA14}$. Thus, the notation A14 identifies the A14 flip-flop and, when used with the prefix "F" which indicates that the signal source is a flip-flop, identifies the output signals of the flip-flop.

In the description of the data processing system, a flip-flop in its 1-state is also said to be in the "set" state or in the "on" state. Conversely, a flip-flop in its 0-state is also said to be in the "reset" state or in the "off" state. When a

flip-flop is transferred to its 1-state, it is said to be "set" or to be "turned on." Conversely, when a flip-flop is transferred to its 0-state, it is said to be "reset" or to be "turned off."

The flip-flop illustrated in FIG. 7 is a conventional flip-flop but employs pulse pedestal triggering which enables the flip-flop to change state in synchronism with the clock signals applied to the logical input gates. The logical gate structure employed to attain pulse pedestal triggering is shown connected to set input terminal **180** in FIG. 7. A trigger signal applied to terminal **180** causes the flip-flop to assume the 1-state.

The logical gate structure includes a two-input AND-gate **182** and a three-input AND-gate **183** and an OR-gate **184**. The gating structure shown in FIG. 7 permits one of two trigger signals to trigger the flip-flop to the 1-state. Any number of gates may be employed to trigger the flip-flop, depending upon the logic requirements. A similar gating structure is connected to reset input terminal **186**.

AND-gate **182** includes terminal **187** for receiving a clock signal input, terminal **188** for receiving a logic signal input and a resistive-capacitive network comprising capacitor **189** and resistor **190**. A constant current source comprising inductor **191**, resistor **192** and a potential source is connected to a terminal of capacitor **189**. When the logic input signal applied to terminal **188** is a binary 0, i.e. in the order of +0.2 volt, the potential on line **194** is approximately −2.6 volts. Capacitor **189** assumes a charge of approximately 3 volts with the left-hand terminal of the capacitor being positive with respect to the right-hand terminal of the capacitor. Upon application of a clock signal to terminal **187**, the level of the clock signal is shifted by virtue of the charge on capacitor **189** and the resultant potential on line **194** is not of sufficient positive potential to trigger the flip-flop.

When the logic input signal applied to terminal **188** is a binary 1, i.e. in the order of +3.8 volts, the potential on line **194** is in the order of +0.1 volt and the charge on capacitor **189** is dissipated through diode **195**. The dissipation of the charge on capacitor **189** occurs in a period of approximately 300 nanoseconds. Consequently, the logic input signal must be applied to AND-gate **182** approximately 300 nanoseconds prior to application of the clock signal. Upon application of the clock signal to terminal **187**, no level shift occurs since the charge on capacitor **189** is approximately 0 volt and the resultant potential on line **194** is sufficient to trigger the flip-flop AND-gate **183** is similar to AND-gate **182** but is a three-input AND-gate.

The flip-flop circuit of FIG. 7 includes a pair of collector-coupled transistor amplifiers comprising transistors **197** and **198** respectively. The collectors of transistors **197** and **198** are connected to output terminals **200** and **201** respectively, the signal at output terminal **200** representing the 1-state of the flip-flop and the output signal at terminal **201** representing the 0-state of the flip-flop. The flip-flop is adapted to receive trigger signals at input terminals **180** and **186**, the trigger signals being developed by the pulse pedestal gating structure previously described.

Assuming that transistor **197** is non-conductive and that transistor **198** is conductive, the collector of transistor **198** and output terminal **200** are near ground potential. Current flow from the +12 volt source through resistor **202** and diode **203** causes the potential at output terminal **201** to be approximately +3.8 volts. The voltage divider comprising resistors **205** and **206** maintains the base of transistor **198** at a positive potential to maintain transistor **198** in a conductive state.

Upon application of a triggering signal to set input terminal **185**, transistor **197** is biased to conduction. The resultant drop in potential at the collector of transistor **197** is coupled through capacitor **207** to the base of transistor **198**, reducing its conductivity with a resultant increase in the potential at the collector of transistor **198**. This increase in potential at the collector of transistor **198** is coupled through capacitor **208** to the base of transistor **197** further increasing its conductivity. This regenerative action rapidly switches transistor **197** into a highly conductive state and transistor **198** into a state of non-conduction. Simultaneously, the potential at output terminal **200** is increased to approximately +3.8 volts while the potential at output terminal **201** decreases to approximately +0.2 volt.

The pulse pedestal flip-flop and the logic gating structure of FIG. 7 thus provides for a delay between the application of the logic input signal to the gating structure and the change in state of the flip-flop. The change in state of the flip-flop occurs simultaneously with the application of a clock signal to the gating structure. The pulse pedestal type flip-flop illustrated in FIG. 7 is employed in the counters of the data processing system, viz. in the P-Register and in the D-Register.

The symbol employed to represent a pulse-pedestal type flip-flop is the same as that shown for the flip-flop of FIG. 6.

NAND-gate

The NAND-gate of FIG. 8 provides the logical NAND function for input logic signals applied to its input terminals. In the system, a binary 1 is represented by a signal of approximately +3.8 volts and a binary 0 is represented by a signal of approximately +0.2 volt. Therefore, the NAND-gate provides an output signal of approximately +0.2 volt when all of the input signals applied to its input terminals represent binary 1's. Conversely, the NAND-gate provides an output signal of approximately +3.8 volts when one or all of the input signals applied thereto represent binary 0's.

The NAND-gate of FIG. 8 is illustrated, by way of example, as having two input terminals **217** and **218**. The NAND-gates employed in the system are not limited to two input terminals, but may have a greater number of input terminals as required.

Diodes **220** and **221**, in conjunction with resistor **222** and the positive voltage source, comprise a conventional AND-gate. The signal on line **223** resulting from the conjunctive operation in the AND-gate is applied to the base of transistor **225**. Resistor **226** serves to limit current flow while capacitor **227** permits transient overdrive of transistor **225** to permit the transistor to turn on faster. Transistor **225** amplifies and inverts the signal on line **223**. Thus, a NOT AND or NAND relationship exists between the output signal at terminal **228** and the input signals at terminals **217** and **218**.

A symbol **230**, shown in FIG. 8, is employed to represent the NAND-gate. Symbol **230** represents a NAND-gate having two input terminals. The two input signals to the NAND-gate are represented by A and B. The output signal of the NAND-gate is therefore $\overline{A \cdot B}$ or $\overline{A+B}$.

The expression $\overline{A \cdot B}$ or $\overline{A+B}$ is a logic expression for the NOT AND or NAND combination of the individual input signals A and B. This form of expression is used in logical equations, which are also known as Boolean equations. Equations of this type will be used to describe the data processing system. The conjunctive operation on signals, such as C and D, is indicated by writing the signals terms adjacent each other with no operative notation therebetween (CD), or with the operator notation (·) between the terms $(C \cdot D)$. The disjuncture operation on signals, such as C and D, is indicated by writing the signal terms adjacent each other with the operator notation (+) between the terms $(C+D)$.

NAND-supplement

The NAND-supplement circuit is employed with the NAND-gate of FIG. 8 in the data processing system to effectively provide OR logic. The NAND-supplement circuit, shown in FIG. 9, is identical to the NAND-gate of FIG. 8 but does not include a pull-up resistor or a clamping diode. When using a NAND-supplement circuit with a NAND-gate, the collector of the NAND-supplement am-

plifying transistor is connected to the collector of the NAND amplifying transistor. The NAND-supplement circuit thus employs the pull-up resistor of the NAND-gate to develop an output potential. A plurality of NAND-supplement circuits may be employed with a single NAND-gate.

The symbol employed for the NAND-supplement circuit is the same as that for the NAND-gate, with the output lines of the NAND-supplement circuits being tied together to the output terminal of the NAND-gate, as shown at 235 in FIG. 9. With input signals A and B applied to the NAND-gate and input signals C, D and E, F applied to the first and second NAND-supplements respectively, the logic expression $\overline{A \cdot B + C \cdot D + E \cdot F}$ indicates the logic output signal of the gates. The logic output signal may also be written as $(\overline{A}+\overline{B})$ $(\overline{C}+\overline{D})$ $(\overline{E}+\overline{F})$, indicating that the output is a binary 1 if one or more inputs are binary 0.

Logic driver

The logic driver circuit is employed in the system to perform the logical operation of conjunction for positive logic signals applied thereto and to provide amplification of the result of the conjunctive operation. The logic driver comprises an AND-gate and a 2-stage transistor amplifier. The AND-gate comprises diodes 238 and 239 and resistor 240. The input signals to the AND-gate are applied to terminals 241 and 242. If both input signals are binary 1's, a positive signal representing a binary 1 is applied to the base of amplifying transistor 244 through RC network 245. The output of amplifying transistor 244, developed across resistor 246, is applied to the base of amplifying transistor 248. Inductor 250 in the collector circuit of transistor 248 increases the speed of the logic driver. The output of the logic driver appears at terminal 251.

The logic driver circuit of FIG. 10 thus performs an AND logic function and amplifies the result of the AND logic function in a non-inverting amplifier. The output of the logic driver circuit is employed in the system to drive flip-flops and other components.

The symbol 253, shown in FIG. 10, is employed to represent the logic driver. The input signals A, B to the logic driver are applied to terminals 254 and 255 and the output signal $A \cdot B$ is derived at terminal 256. Although two input terminals are shown, the logic driver may incorporate AND-gates accommodating a larger number of input signals.

Full adder

The full adder circuit, illustrated in FIG. 11, is employed in the data processing system to check and generate parity and to perform binary and decimal arithmetic operations. The full adder circuit includes a carry circuit 255, a sum circuit 256 and various logic circuits providing inputs to the carry and sum circuits.

Carry circuit 255 comprises output transistors 258 and 259 and phase-inverting transistor 260. Transistors 258 and 259 provide the complementary output signals C and $\overline{C}$ respectively, while phase-inverting transistor 260 provides complementary drive signals for transistors 258 and 259. The input signals to carry circuit 255 are provided by three 2-input AND-gates 261, 262 and 263 and a 3-input OR-gate 264. The input signals to AND-gates 261, 262 and 263 are XY, XZ and YZ respectively. The three inputs to OR-gate 264 are the output signals of AND-gates 261, 262 and 263. Symbol 265 is employed in the description of the data processing system to represent a conventional diode AND-gate, such as AND-gate 261. Symbol 266 is similarly employed in the system description to represent a conventional diode OR-gate, such as OR-gate 264.

Sum circuit 256 comprises output transistors 267 and 268 and phase-inverting transistor 269. Transistors 267 and 268 provide the complementary output signals S and $\overline{S}$ respectively while phase-inverting transistor 269 provides complementary drive signals for transistors 267 and 268. The input signals to sum circuit 256 are provided by a 3-input OR-gate 271, a 3-input AND-gate 272, a 2-input AND-gate 273 and a 2-input OR-gate 274. The input signals to OR-gate 271 and AND-gate 272 are XYZ. The inputs to AND-gate 273 are $\overline{C}$ and the output signal of OR-gate 271. The inputs to OR-gate 274 are the output signals of AND-gates 272 and 273.

In operation, with each of input signals X, Y and Z representing binary 0's, transistor 260 is conductive in the linear region of its characteristics. Transistor 260 biases transistor 258 to saturation and biases transistor 259 to a non-conductive state. The output signal C at terminal 277 is therefore in the order of +0.2 volt while the signal $\overline{C}$ at output terminal 278 is in the order of +3.8 volts. A similar condition exists in sum circuit 256 with the output signals S and $\overline{S}$ at terminals 280 and 281 being +0.2 volt and +3.8 volts respectively.

Assuming one of the inputs X, Y and Z to be a binary 1 and the remainder of the inputs to be binary 0's, none of the AND-gates 261, 262 or 263 is enabled and carry circuit 255 remains in the same condition as when inputs X, Y and Z were all binary 0's. Considering sum circuit 256, OR-gate 271 is enabled by the binary 1 input signal and the output signal of OR-gate 271 in conjunction with the signal $\overline{C}$ enables AND-gate 273. The output of AND-gate 273 biases transistor 269 to saturation which in turn biases transistor 267 to a state of non-conduction and transistor 268 to saturation. The resulting output signal S at terminal 280 is in the order of +3.8 volts while the resulting output signal $\overline{S}$ at terminal 281 is in the order of +0.2 volt. Thus, with one of the inputs X, Y and Z a binary 1, the outputs S and $\overline{C}$ represent binary 1's and outputs $\overline{S}$ and C represent binary 0's.

Assuming that two of the three inputs X, Y and Z are binary 1's, one of the AND-gates 261, 262 or 263 is enabled. The resulting output signal from OR-gate 264 drives transistor 260 into saturation which in turn biases transistor 258 to a state of non-conduction and transistor 259 to saturation. The resulting output signals C and $\overline{C}$ at terminals 277 and 278 are +3.8 volts and +0.2 volt respectively.

Considering sum circuit 256, AND-gate 273 is disabled since the signal $\overline{C}$ at terminal 278 is +0.2 volt. Transistor 269 is accordingly biased to conduction in the linear region of its characteristics, driving transistor 267 to saturation and transistor 268 to a state of non-conduction. The resulting output signals S and $\overline{S}$ at terminals 280 and 281 are +0.2 volt and +3.8 volts respectively. Thus, with two of the input signals X, Y and Z being binary 1's, output signals C and $\overline{S}$ represent binary 1's while output signals $\overline{C}$ and S represent binary 0's.

With all three of the inputs X, Y and Z being binary 1's, carry circuit 255 remains in the same state with output signal C representing a binary 1 and output signal $\overline{C}$ representing a binary 0. The three binary 1 input signals enable AND-gate 272 to drive transistor 269 to saturation. Transistor 269 in turn changes the state of transistors 267 and 268, driving transistor 267 to a state of non-conduction and transistor 268 to saturation. The resulting output signal S and $\overline{S}$ at terminals 280 and 281 are +3.8 volts and +0.2 volt respectively. Thus, with all three of the input signals X, Y and Z being binary 1's, output signals S and C represent binary 1's while output signals $\overline{S}$ and $\overline{C}$ represent binary 0's.

Although the circuit has been described with the number of binary 1's represented by input signals X, Y and Z progressing from 0–3, the operation of the circuit is not limited to this sequence since the binary 1's represented by input signals X, Y and Z can change from a given combination to any other combination. The rela-

tionship between the input signals X, Y and Z and the state of output signals C, $\overline{C}$, S and $\overline{S}$ is indicated in the following table:

X Y Z	S $\overline{S}$ C $\overline{C}$
0 0 0	0 1 0 1
0 0 1	1 0 0 1
0 1 0	1 0 0 1
1 0 0	1 0 0 1
0 1 1	0 1 1 0
1 0 1	0 1 1 0
1 1 0	0 1 1 0
1 1 1	1 0 1 0

The symbol 283, shown in FIG. 11, is employed to represent the full adder circuit. The three input lines on the left side of the symbol represent the three input signals X, Y and Z. The four lines extending from the right side of the symbol represent the output symbols S, $\overline{S}$, C and $\overline{C}$.

Register

A register is adapted to provide temporary storage of data being processed or data being transferred between system components. The register comprises a plurality of flip-flops, one flip-flop being provided in the register for each bit of data to be stored in the register. Some registers of the Central Processor store a four character data word and, therefore, comprise 24 flip-flops. Other registers of the system, e.g. the A-Register, store less than a data word and comprise a proportional number of flip-flops.

The flip-flops of a register are identified according to the register designation and the numerical significance of the data bits stored therein. Thus, a particular register flip-flop is designated as the "M*i*" flip-flop, where "M" identifies the M-Register and "*i*" identifies the order of significance of the bit stored in the flip-flop. For example, the M23 flip-flop stores the 24th or the most-significant bit of the M-Register. The M06 flip-flop stores the least-significant bit of character 2 in the M-Register. Data movement between the registers of the Central Processor is by parallel transfer of the bits stored in the flip-flops of one register to the corresponding flip-flops of the receiving register.

DATA PROCESSING SYSTEM—DETAILS

The data processing system is shown symbolically in FIG. 12 to illustrate the elements of the system which store data, the paths of data transfer between these elements, and major control elements of the system.

Instruction words for data processing, auxiliary words for addressing and control, data words which are to be processed and data words which are the result of processing are stored in Memory **10** during data processing operations. Temporary storage of data words is provided, during data processing operations, in the various registers of the system. Transfer of data between registers and other elements of the system, as indicated by the interconnecting lines of FIG. 12, is effected by the parallel transfer of binary digits from the source register to the receiving register or element.

Memory **10** operates asynchronously with the other elements of the Central Processor and contains its own timing logic, control logic and address register. Memory **10** communicates with the other elements of the Central Processor, viz. the Program Processor and the Input/Output Control Unit, to obtain binary coded addresses, to receive and furnish data words, instruction words and auxiliary words, and to receive and provide control signals which enable synchronization of memory timing with the timing of the remainder of the Central Processor. Address information is transmitted to Memory **10** through B-Gates **317**.

I-Register **313** stores the operation code comprising bits 18–23 of the instruction word. The operation code in I-Register **313** controls the type of operation to be executed by the system. The address control field of the instruction word, comprising bits 15–17, is stored in Q-Register **306**. The address control field in Q-Register **306** controls the development of the instruction address field and may control the development of the second address of a two-address instruction. The address field of the instruction word comprising bits 0–14 is normally stored, after development if required, in D-Register **311** but may, during the execution of certain instructions, be stored in E-Register **303**. The instruction address field stored in the D-Register or the E-Register indicates the memory location from which an operand word is to be read or into which a word is to be stored. The address in Memory of the next instruction to be executed by the Program Processor is stored in P-Register **310**. As execution of an instruction is nearing completion, the next instruction address stored in P-Register **310** is applied to Adder **302**, enabling the Central Processor to obtain the next instruction from Memory **10**.

All words read from Memory **10** or transferred to Memory **10** must pass through M-Register **301**. Therefore, when an instruction is to be obtained from Memory **10**, the address in the P-Register **310** is transferred, by way of Adder **302** and D-Register **311**, through B-Gates **317** and the instruction word is read from the addressed memory location into the M-Register. The operation code, the address control field and the address field of the instruction word in the M-Register are then transferred to I-Register **313**, Q-Register **306** and either D-Register **311** or E-Register **303** respectively, and the instruction is then executed. Operand words employed during instruction execution are obtained from Memory **10** by the same process, i.e. by initially transferring the operand word to the M-Register.

When a word is transferred from Memory **10** to M-Register **301**, the 25th or parity bit is transmitted to Parity Check and Generation Logic **316**. Parity Check and Generation Logic **316** samples the word in the M-Register and, employing the parity bit received from Memory **10**, determines whether or not a parity error exists in the word transferred from Memory **10**. Similarly, when a word is temporarily stored in M-Register **301** and is to be transferred to Memory **10**, Parity Check and Generation Logic **316** samples the word in the M-Register **301** and generates the correct parity bit which is transferred to Memory **10** as the 25th bit of the word.

Arithmetic operations, shift operations, compare operations, logic operations, etc. on operand words are performed primarily by employing M-Register **301**, Adder **302** and E-Register **303**. Adder **302** is adapted to perform the operation of addition on information received simultaneously from M-Register **301** and E-Register **303**. Adder **302** also performs the operation of addition on information received simultaneously from M-Register **301** and from P-Register **310**, D-Register **311** or CC-Register **304**. The output of Adder **302** may be transferred to E-Register **303**, M-Register **301**, D-Register **311**, P-Register **310** or CC-Register **304**, as illustrated by the data transfer paths in FIG. 12. CC-Register **304**, N-Register **305** and Q-Register **306** serve as counters or temporary storage registers during execution of predetermined instructions.

Control of operations in the Central Processor is accomplished primarily by Operations Control Unit **318**, Timing Control Unit **319**, I-Register **313**, P-Register **310**, D-Register **311**, and A-Register **312**. The contents of I-Register **313** is decoded by Operations Control Unit **318**, which directs appropriate control logic during execution of the instruction by the Central Processor. Operations Control Unit **318** includes a register for selecting blocks of micro-operations, control logic, and decode logic for I-Register **313** and for the selector register. Timing Control Unit **319** controls the internal timing of the Central Processor and enables synchronization of the Program Processor and the Input/Output Control Unit **318** with

Memory 10. Timing Control Unit 319 includes a generator for providing clock pulses to the Central Processor, logic for controlling clock pulse generation, and a counter for defining time periods in the Central Processor.

A-Register 312 is employed to store the address of the most-significant word of the four-word Accumulator in Memory 10. Accumulator Length Indicator Register 314, comprising flip-flops PL2 and PL1, contains a count indicating the number of words in the working Accumulator. The contents of registers 312 and 314 form the address of the most-significant word of the working Accumulator. Accumulator Counter Register 315 comprising flip-flops AC2 and AC1 is employed, in conjunction with A-Register 312, to successively address each word of the working Accumulator.

D-Register 311 is employed primarily as an address register for transmitting address information through B-Gates 317 to Memory 10. P-Register 310 is a counter which is periodically advanced to form the address of the next word, which may be either an instruction word or an auxiliary word, in the P-sequence.

Input/Output Control Unit 13 includes a plurality of channels, each peripheral subsystem used with the Central Processor being connected to one of the input/output channels. Each input/output channel permits the transmission of instructions from the Central Processor to the peripheral subsystem connected to the channel, the transfer of data between Memory 10 and the peripheral subsystem connected to the channel, and the transmission of information indicative of the peripheral subsystem and channel operating conditions to the Central Processor.

The Central Processor receives data through Input/Output Control Unit 13 from the typewriter, document handler, card reader, perforated tape reader, magnetic tape controller and disc storage controller, as indicated symbolically by line 323 in FIG. 12. The Central Processor may transmit data, through Input/Output Control Unit 13 to the typewriter, cardpunch, perforated tape punch, printer, magnetic tape controller and the disc storage controller, as indicated symbolically by line 324.

A description of the system operation is provided in a succeeding section entitled "Instructions" and in the sections following this section. Details of the system structure and of the components of the system are provided in FIGS. 14, 16–18, 20, 21, 23, 25–118, 170–172, 174 and 178–196.

Data processing operations

The operation code of an instruction word may have any one of 53 bit configurations, each configuration representing a different one of 53 instruction codes. Each instruction code directs a fundamentally different data processing operation of the system. The system executes the instruction codes of a succession of instruction words to process data. This succession of instruction words is termed a "program." In any program, some or all of the 53 fundamental operations may be executed one or more times to process data. The numeral assigned to each instruction is the binary operation code of the instruction expressed in octal notation.

The instruction repertoire of the Central Processor includes both single-address and two-address instructions. Single-address instructions employ a single operand address. Two-address instructions employ two operand addresses to execute the operation called for by the operation code of the instruction.

The fundamental data processing operations which the data processing system executes may be grouped by function as indicated below. In the description of the operations, location Y is the memory location corresponding to the address contained in the instruction address field, after development, if any, of the instruction address field.

A. Data Movement Operations—

Instruction 40, Load Single (LDS), single-address: The accumulator working length is set to single and the contents of location Y replace the contents of the single Accumulator. The contents of location Y are not changed.

Instruction 41, Load Double (LDD), single-address: The accumulator working length is set to double and the contents of locations Y–1 and Y replace the contents of the double Accumulator. The contents of location Y–1 and Y are not changed.

Instruction 42, Load Triple (LDT), single-address: The accumulator working length is set to triple and the contents of locations Y–2, Y–1 and Y replace the contents of the triple Accumulator. The contents of the memory field are not changed.

Instruction 43, Load Quadruple (LDQ), single-address: The accumulator working length is set to quadruple and the contents of locations Y–3, Y–2, Y–1 and Y replace the contents of the quadruple Accumulator. The contents of the memory field are not changed.

Instruction 44, Store Single (STS), single-address: The contents of location Y are replaced by the contents of the single Accumulator, regardless of the accumulator working length setting. The accumulator working length and contents are not changed.

Instruction 45, Store Double (STD), single-address: The contents of locations Y–1 and Y are replaced by the double Accumulator, regardless of the accumulator working length setting. The accumulator working length and contents are not changed.

Instruction 46, Store Triple (STT), single-address: The contents of locations Y–2, Y–1 and Y are replaced by the contents of the triple Accumulator, regardless of the accumulator working length setting. The accumulator working length and contents are not changed.

Instruction 47, Store Quadruple (STQ), single-address: The contents of locations Y–3, Y–2, Y–1 and Y are replaced by the contents of the quadruple Accumulator, regardless of the accumulator working length setting. The accumulator working length and contents are not changed.

Instruction 30, Move From First Memory (MFM), two-address: The contents of location Y are moved to a second location, leaving the contents of location Y unchanged. The second location is defined by the address of a Fixed Location Index Word or by a second address sequence.

Instruction 31, Move From Immediate (MFI), two-address: The address field Y of the instruction word is moved to the address field of a second location. The operation code field and address control field of the second location are not changed. The second location is defined by the address of a Fixed Location Index Word or by a second address sequence.

Instruction 32, Move to First Address Field (MTA), two-address: The address field of location Y is replaced by the address field of a second location. The contents of the second location are not changed. The second location is defined by the address of a Fixed Location Index Word or by a second address sequence.

Instruction 06, Move (MOV), two-address: The contents of a memory field whose most-significant word is at location Y are replaced by words moved from a second memory field. The second memory field is not changed. The location of the most-significant word of the second memory field is defined in a control word which also determines the number of words to be moved. The location of the control word is defined by the address of a Fixed Location Index Word or by a second address sequence.

Instruction 20, Explode (EXP), single-address: The working Accumulator is cleared to zero. One to four characters from location Y are distributed to the working Accumulator, one character to the character 3 position of each accumulator word. Accumulator words outside the working Accumulator and the contents of location Y are not changed.

Instruction 21, Implode (IMP), single-address: The least-significant character (character 3) of each working accumulator word is gathered into location Y, the character from the least-significant accumulator word being placed in the least-significant character position of location Y with characters from successively more-significant accumulator words being placed in correspondingly more-significant character positions of location Y. If any accumulator words are excluded because of a working accumulator length less than quadruple, corresponding character positions in location Y are cleared to zero. The accumulator contents and length are not changed.

B. Shift Operations—

Instruction 22, Shift (SHG): The Shift instruction provides a variety of capabilities involving the Accumulator. The type of shift, the length of the Accumulator affected, the direction of shift, and the number of positions to be shifted are determined by the address field of the Shift instruction which, in this case, does not refer to a memory location.

C. Arithmetic Operations—

Instruction 50, Add Decimal Single (ADS), single-address: The contents of location Y are added to the contents of the working Accumulator. The result is placed in the Accumulator and the contents of location Y are not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the working Accumulator are set to zero. The accumulator length is not affected by this instruction.

Instruction 60, Subtract Decimal Single (SDS), single-address: The contents of location Y are subtracted from the contents of the working Accumulator. The results are placed in the Accumulator and the contents of location Y are not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the working Accumulator are set to zero. The accumulator length is not affected by this instruction.

Instruction 51, Add Decimal Double (ADD), single-address: The contents of locations Y–1 and Y are added to the contents of the working Accumulator. The result is placed in the Accumulator and the memory field is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the working Accumulator are set to zero. If the working Accumulator is double, triple or quadruple, the length is not changed. If the working Accumulator is single, the length is set to double and the added working accumulator word is cleared to zero before addition occurs.

Instruction 61, Subtract Decimal Double (SDD), single-address: The contents of locations Y–1 and Y are subtracted from the contents of the working Accumulator. The result is placed in the Accumulator and the memory field is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the working Accumulator are set to zero. If the working Accumulator is double, triple or quadruple, the length is not changed. If the working Accumulator is single, the length is set to double and the added working accumulator word is cleared to zeros before subtraction occurs.

Instruction 52, Add Decimal Triple (ADT), single-address: The contents of locations Y–2, Y–1 and Y are added to the contents of the working Accumulator. The result is placed in the Accumulator and the memory field is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the working Accumulator are set to zero. If the working Accumulator is triple or quadruple, the length is not changed. If the working Accumulator is double or single, the length is set to triple and the one or two added working accumulator words are cleared to zeros before addition occurs.

Instruction 62, Subtract Decimal Triple (SDT), single-address: The contents of locations Y–2, Y–1 and Y are subtracted from the contents of the working Accumulator. The result is placed in the Accumulator and the memory field is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the working Accumulator are set to zero. If the working Accumulator is a triple or quadruple, the length is not changed. If the working Accumulator is double or single, the length is set to triple and the one or two added working accumulator words are cleared to zeros before subtraction occurs.

Instruction 53, Add Decimal Quadruple (ADQ), single-address: The contents of locations Y–3, Y–2, Y–1 and Y are added to the contents of the working Accumulator. The result is placed in the Accumulator and the memory field is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the Accumulator are set to zero. If the working Accumulator is quadruple, the length is not changed. If the working Accumulator is triple, double or single, the length is set to quadruple and the one, two or three added working accumulator words are cleared to zeros before addition occurs.

Instruction 63, Subtract Decimal Quadruple (SDQ), single-address: The contents of locations Y–3, Y–2, Y–1 and Y are subtracted from the contents of the working Accumulator. The result is placed in the Accumulator and the memory field is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the Accumulator are set to zero. If the working Accumulator is quadruple, the length is not changed. If the working Accumulator is triple, double or single, the length is set to quadruple and the one, two or three added working accumulator words are cleared to zeros before subtraction occurs.

Instruction 54, Add to Memory Single (AMS), single-address: If the working Accumulator is single, its contents are added to the contents of location Y and the result is stored in location Y. The Accumlulator contents and length are not changed. The sign of the result is placed in the zone bits of the least-significant character of location Y; zone bits of all other characters in location Y are set to zero. If the accumulator length is double, triple or quadruple, the contents of location Y are not changed and the Overflow flip-flop is set to the 1-state.

Instruction 55, Add to Memory Double (AMD), single-address: If the working Accumulator is single or double, its contents are added to the contents of locations Y–1 and Y and the result is stored in locations Y–1 and Y. The accumulator contents and length are not changed. The sign of the result

is placed in the least-significant character position of location Y; zone bits of all other characters in locations Y–1 and Y are set to zeros. If the accumulator working length is triple or quadruple, the contents of locations Y—1 and Y are not changed and the Overflow flip-flop is set to the 1-state.

Instruction 56, Add to Memory Triple (AMT), single-address: If the working Accumulator is single, double or triple, its contents are added to the contents of locations Y–2, Y–1 and Y and the result is stored in these locations. The accumulator contents and length are not changed. The sign of the result is placed in the least-significant character position of location Y; zone bits of all other characters of the result are set to zero. If the accumulator working length is quadruple, the contents of locations Y–2, Y–1 and Y are not changed and the Overflow flip-flop is set to the 1-state.

Instruction 57, Add to Memory Quadruple (AMQ), single-address: The contents of the working Accumulator are added to the four-word memory field in locations Y–3, Y–2, Y–1 and Y, and the result is stored in these locations. The accumulator contents and working length are not changed. The sign of the result is placed in the least-significant character position of location Y; zone bits of all other characters of the result are set to zero.

Instruction 34, Add Binary to Memory (ABM), two-address: The contents of location Y are added absolutely, in binary, to the contents of a second location. The result is placed in the second location. The contents of location Y are not changed. The second location is defined by the address of a Fixed Location Index Word or by a second address sequence.

Instruction 35, Subtract Binary From Memory (SBM), two-address: The contents of location Y are subtracted absolutely, in binary, from the contents of a second location. The result is placed in the second location. The contents of location Y are not changed. The second location is defined by the address of a Fixed Location Index Word or by a second address sequence.

Instruction 33, Add Immediate to Memory (AMI), two-address: The address field Y of the instruction word is added in binary to the address field of a second location. The result is placed in the address field of the second location. The operation code field and the address control field of the second location are not changed. The second location is defined by the address of a Fixed Index Word or by a second address sequence.

Instruction 27, Variable Length Multiply (VLM), single-address: The contents of locations Y–1 and Y, which constitute an eight-character multiplicand, are multiplied by the least-significant character in the Accumulator. The generated nine-character product is placed in the nine most-significant character positions of the Accumulator. The multiplier character from the Accumulator is lost during each VLM execution. The multiplicand field in locations Y–1 and Y is not changed. The accumulator working length is not changed. The VLM instruction must be executed once for each multiplier digit. The sign of the product appears in the zone bits of the least-significant character of the product; the zone bits of all other product digits are set to zero.

Instruction 26, Variable Length Divide (VLD), single-address: The eight-character memory field in locations Y–1 and Y is divided into the contents of the nine most-significant character positions of the Accumulator. The quotient is placed in the least-significant position of the Accumulator. The contents of the memory field in locations Y–1 and Y and the accumulator working length setting are not changed. To form the quotient of a divisor in Memory and a dividend in the Accumulator, the VLD must be repeated as many times as the number of desired quotient digits.

D. Compare Operations—

Instruction 03, Compare Alphanumeric Accumulator to Memory (CAA), single-address: The contents of the working Accumulator are compared with the contents of a memory field whose least-significant location is Y and whose length corresponds to the length of the working Accumulator. Comparison is based upon the absolute binary contents of the two fields, although the fields can be either binary or alphanumeric. The result of the comparison (Accumulator less, equal, or greater) is placed in the comparison indicators. The accumulator contents and length and the memory field contents are not changed.

Instruction 02, Compare Decimal Accumulator to Memory (CDA), single-address: The contents of the working Accumulator are compared algebraically with the contents of a memory field whose least-significant location is Y and whose length corresponds to the length of the working Accumulator. The only zone bits considered in the comparison are the signs of the memory and accumulator fields. When the signs are compared, a minus sign (10 in the zone bits) is less than all three plus sign configurations (00, 01 or 11). The result of the decimal comparison (accumulator field less, equal, or greater) is placed in the comparison indicators. The accumulator contents and length and the memory field contents are not changed.

Instruction 04, Compare Second to First Memory (CMM), two-address: The contents of location Y and the contents of a second location are compared. Comparison is based upon the absolute binary contents of the two fields, although the fields can be either binary or alphanumeric. The result of the comparison (second location less, equal or greater) is placed in the comparison indicators. The contents of the compared locations are not changed. The second location is defined by the address of a Fixed Location Index Word or by a second address sequence.

Instruction 01, Compare Memory to Immediate (CMI), two-address: Address field Y of the instruction word and the address field of a second location are compared absolutely. The result of the comparison (second location address field less, equal or greater) is placed in the comparison indicators. The instruction word and the contents of the second location are unchanged. The second location is defined by the address of a Fixed Location Index Word or by a second address sequence.

E. Branch Operations—

Instruction 10, Branch Unconditionally (BRU), single-address: The contents of the P-Register are replaced by the instruction address field Y, causing a program branch to location Y.

Instruction 17, Store Program Counter and Branch (SPB), two-address: The contents of the P-Register (the address of the word following the SAS word in the P-sequence) are stored in Memory, and the instruction address field Y is placed in the P-Register, effecting a branch to location Y. The location in which the contents of the P-Register are stored is defined by the address of a Fixed Location Index Word or by a second address sequence. In any location in which the contents of the P-Register are stored, only the ad-

dress field is affected. The other bit positions remain unchanged.

Instruction 13, Branch if Equal (BRE), single-address: If the specified condition exists in the comparison indicators, the contents of the P-Register are replaced by the instruction address field Y, causing a program branch to location Y. If the specified condition does not exist, the branch is not made and the next instruction is executed in normal sequence. Execution of the BRE instruction does not affect the comparison indicators.

Instruction 14, Branch if Less (BRL), single-address: If the specified condition exists in the comparison indicators, the contents of the P-Register are replaced by the instruction address field Y, causing a program branch to location Y. If the specified condition does not exist, the branch is not made and the next instruction is executed in normal sequence. Execution of the BRL instruction does not affect the comparison indicators.

Instruction 12, Branch if Greater (BRG), single-address: If the specified condition exists in the comparison indicators, the contents of the P-Register are replaced by the instruction address field Y, causing a program branch to location Y. If the specified condition does not exist, the branch is not made and the next instruction is executed in normal sequence. Execution of the BRG instruction does not affect the comparison indicators.

Instruction 15, Branch if Zero (BRZ), single-address: A test is made of the working accumulator bits. If all accumulator bits are zero, the contents of the P-Register are replaced by address field Y, causing a program branch to location Y. If any working accumulator bits are not zero, the next instruction is executed in normal sequence.

Instruction 11, Branch if Minus (BRM), single-address: The sign of the Accumulator, contained in the least-significant character of the Accumulator, is tested. If the sign is minus (10 in the zone bits), the contents of the P-Register are replaced by instruction address field Y, causing a program branch to location Y. If the sign is not minus, the next instruction is executed in normal sequence.

Instruction 16, Branch on Count (BRC), two-address: This instruction executes a branch for a specified number of times and employs a counter to control the number of executions. The branch address is contained in the instruction word address field Y. A second location contains the control word which serves as a counter. The location of the control word is defined by the address of a Fixed Location Index Word or by a second address sequence.

F. Logic Operations—

Instruction 24, AND to Memory (ANM), two-address: The contents of location Y are used to modify the contents of a second location. The result is placed in the second location, leaving the contents of location Y unchanged. The modification is as follows: for each bit position of location Y that contains a zero, the corresponding bit position of the second location is set to zero; for each bit position of location Y that contains a one, the corresponding bit position of the second location is left unchanged. The second location is defined by the address of a Fixed Location Index Word or by a second address sequence.

Instruction 23, OR-Inclusive to Memory (RIM), two-address: The contents of location Y are used to modify the contents of a second location. The result is placed in the second location, leaving the contents of location Y unchanged. The modification is as follows: for each bit position in location Y that contains a one, the corresponding bit position of the second location is set to one; for each bit position of location Y that contains a zero, the corresponding bit position of the second location is left unchanged. The second location is defined by the address of a Fixed Location Index Word or by a second address sequence.

Instruction 25, OR-Exclusive to Memory (RXM), two-address: The contents of location Y are used to modify the contents of a second location. The result is placed in the second location, leaving the contents of location Y unchanged. The modification is as follows: for each bit position of location Y that contains a one, the corresponding bit position of the second location is inverted (zero becomes one, one becomes zero); for each bit position of location Y that contains a zero, the corresponding bit position of the second location is left unchanged. The second location is defined by the address of a Fixed Location Index Word or by a second address sequence.

G. Accumulator Operations—

Instruction 36, Load Accumulator Location and Length (LAL), single-address: The accumulator location and length are established as specified by the instruction address field Y. The most-significant 13 bits of address field Y, together with the two assumed least-significant zeros, define the location of the most-significant word of the four-word Accumulator. The most-significant word of the four-word Accumulator is therefore always at a zero, modulo 4 location. Bits 1 and 0 of address field Y establish the length of the Accumulator as follows:

00	Quadruple
01	Triple
10	Double
11	Single

Address field Y can also be interpreted as specifying the location of the most-significant word of the working Accumulator. In relocating the Accumulator, the contents of both the old and new locations are unchanged.

Instruction 37, Store Accumulator Location and Length (SAL), single-address: The current accumulator location and length are stored in the address field of location Y. The operation code field and the address control field of the contents of the location Y are not changed. The most-significant 13 bits stored in the address field of location Y, together with two assumed least-significant zeros, indicate the location of the most-significant word of the four-word Accumulator. Bits 1 and 0 stored in the address field of location Y indicate the working length of the Accumulator as follows:

00	Quadruple
01	Triple
10	Double
11	Single

The 15-bit field stored in location Y can also be interpreted as specifying the location of the most-significant word on the working Accumulator. In executing the instruction, the actual accumulator location, contents and length are not changed.

H. Special Purpose Operations—

Instruction 05, Edit (EDT), single-address: The contents of the memory field whose least-significant word is at location Y are processed from right to left under control of four, eight, twelve or sixteen format characters in the working Accumulator. For each format character, a single character is placed or retained in the working Accumulator. The number of result characters equals the number of format characters. The memory field and the accumulator working length are not changed.

Instruction 67, Central Processor Operations (CPO), two-address: The central processor operations instructions permit the status of specific indicators to be stored in Memory or permit status to be set in these indicators. The second location is defined by the address of a Fixed Location Index Word or by a second address sequence.

Instruction 00, Halt (HLT), single-address: This instruction causes the Central Processor to cease processing instructions. The Halt instruction establishes control conditions which inhibit the honoring of requests for program interrupts, but which allow data transfer sequences to occur if any peripheral subsystems have not terminated operations. Actuation of the RUN switch on the Console causes continuous instruction processing to be resumed, starting with the next instruction in the P-sequence.

Instruction 07, General (GEN), two-address: The General instruction is used to execute one of the following operations:

(1) A peripheral subsystem input/output operation or a control operation is initiated, resettable status is reset and the remaining peripheral status is stored in a second location.

(2) Any existing resettable status in the addressed peripheral subsystem is reset and the remaining peripheral status is stored in a second location.

(3) The existing peripheral status is stored in a second location. The second location required in the execution of the above operations is defined by the address of a Fixed Location Index Word or by a second address sequence.

GLOSSARY AND INDEX OF SIGNALS

The signals provided by the system circuit elements are tabulated below. The major element of the Central Processor, e.g. Memory, Program Processor or Input/Output Control Unit, in which the signal originates is indicated. The portion of the description which illustrates or describes the respective source elements for each of the signals is also identified.

CSTR, Memory, FIG. 36: A signal which gates the sense amplifier output signals representing the bits of a word read from a memory location to the Data Drivers.

DAAC, Program Processor, FIG. 93: A signal which issues when the contents of the Accumulator Counter Register are 0, 0. This state of the Accumulator Counter Register, in conjunction with the contents of the A-Register, forms the address of the accumulator word A location.

DAAE, Program Processor, FIG. 93: A signal which issues when the address in the D-Register is the same as the address stored in the A-Register and the Accumulator Counter Register. This signal inhibits the clearing of the M-Register and the restoration of the operand word to Memory during execution of one of the Instructions 50–57 and 60–63.

DABM, Program Processor, FIG. 93: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to Instruction 34, Add Binary to Memory (ABM).

DABX, Program Processor, FIG. 93: A signal which issues during particular blocks of micro-operations and during the execution of predetermined instructions to transfer the accumulator address in the A-Register and the Accumulator Counter Register through the B-Gates to the B-Register of Memory.

DAC1, Program Processor, FIG. 93: A signal which issues when flip-flop AC1 of the Accumulator Counter Register is set to the 1-state.

DAC2, Program Processor, FIG. 93: A signal which issues when flip-flop AC2 of the Accumulator Counter Register is set to the 1-state.

DACD, Program Processor, FIG. 93: A signal which issues at predetermined times during the execution of predetermined instructions to reduce the count in the Accumulator Counter Register by one. If the Accumulator Counter Register and the A-Register contain the address of the accumulator word A location, signal DACD modifies the contents of the Accumulator Counter Register to form the address of the most-significant word of the working accumulator.

DACS, Program Processor, FIG. 93: A signal which issues upon actuation of the appropriate console switch to set flip-flop ARM to the 1-state, causing the typewriter to type out the contents of the A-Register.

DACU, Program Processor, FIG. 93: A signal which causes the count in the Accumulator Counter Register to be advanced by one. If the state of the Accumulator Counter Register, in conjunction with the contents of the A-Register, addresses the most-significant word of the working Accumulator, signal DACU reduces the count in the Accumulator Counter Register to zero.

DAD0, Program Processor, FIG. 93: A signal which issues during a decimal addition of two words in the Adder to cause six to be added to character 0 of the result to correct the result.

DAD1, Program Processor, FIG. 93: A signal which issues during a decimal addition of two words in the Adder to cause six to be added to character 1 of the result to correct the result.

DAD2, Program Processor, FIG. 93: A signal which issues during a decimal addition of two words in the Adder to cause six to be added to character 2 of the result to correct the result.

DAD3, Central Processor, FIG. 93: A signal which issues during a decimal addition of two words in the Adder to cause six to be added to character 3 of the result to correct the result.

DADC, Program Processor, FIG. 93: A signal which indicates that an instruction of the Add Decimal Class, which includes Instructions 50–57, is being executed by the Central Processor.

DADU, Program Processor, FIG. 93: A signal which causes flip-flops AC1 and AC2 of the Accumulator Counter Register to be reset to the 0-state.

DAEX, Program Processor, FIG. 93: A signal which gates the accumulator word address in the A-Register and the Accumulator Length Indicator Register to the E-Register.

DAGI, Program Processor, FIG. 93: A signal which issues when the addressed accumulator word is outside the accumulator length specified by certain instructions; e.g. if the address in the A-Register and the Accumulator Counter Register is the address of accumulator word B and the Program Processor is executing Instruction 50, Add Decimal Single (ADS).

DAGL, Program Processor, FIG. 93: A signal which indicates that the A-Register and the Accumulator Counter Register contain the address of an accumulator word outside the working accumulator, as defined by the Accumulator Length Indicator Register.

DALS, Program Processor, FIG. 93: A signal which issues during the execution of Instruction 22, Shift (SHG), to indicate that the Accumulator is being shifted for the last time prior to completion of execution of the instruction.

DAMC, Memory, FIG. 36: A signal which prevents one 16K memory unit of a 32K memory from initiating a memory cycle before the other 16K memory unit has completed a memory cycle.

DAMG, Program Processor, FIG. 93: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instruc-

tions in the Add to Memory group, viz. Instructions 54–57.

DAMI, Program Processor, FIG. 93: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 33, Add Immediate to Memory (AMI).

DAMR, Program Processor 93: A signal which is generated in the Program Processor and transmitted to Memory to enable a word to be read from Memory into the M-Register of the Program Processor; signal DAMR distinguishes a read/restore memory cycle from a clear/write memory cycle.

DAMS, Program Processor, FIG. 93: A signal which issues when the address control field of an instruction word has a value greater than 0, indicating that the address field of the instruction word is to be developed, if the instruction is a single-address instruction or if the instruction is a two-address instruction and the value of the address control field is 7, or that the second address of the instruction is the address of a Fixed Location Index Word, if the instruction is a two-address instruction and the value of the address control field is in the range of 1–6.

DANM, Program Processor, FIG. 94: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to Instruction 24, AND to Memory (ANM).

DAOD, Program Processor, FIG. 94: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions of the Add Decimal Class, viz. Instructions 50–57, one of the instructions of the Subtract Decimal Group, viz. Instructions 60–63, or Instruction 26.

DAOX, Program Processor, FIG. 94: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the Instructions 23–25.

DARS, Program Processor, FIG. 94: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the Instructions 50–53 and 60–63.

DASB, Program Processor, FIG. 94: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to either Instruction 34 or Instruction 35.

DASF, Program Processor, FIG. 94: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the Instructions 51–53, 55–57 and 61–63.

DASG, Program Processor, FIG. 94: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions of the Add Decimal Class, viz. Instructions 50–57, or one of the instructions of the Subtract Decimal Group, viz. Instructions 60–63.

DAUD, Program Processor, FIG. 94: A signal which resets flip-flop AC1 of the Accumulator Counter Register to the 0-state and also resets flip-flop AC2 to the 0-state during the execution of predetermined instructions employing a three-word Accumulator.

DAUQ, Program Processor, FIG. 94: A signal which issues during the execution of predetermined instructions to reset flip-flop AC2 of the Accumulator Counter Register to the 0-state.

DB00–DB14, Program Processor, FIGS. 94–95: Address signals provided by the B-Gates for application to the B-Register of Memory to define the memory location from which a word is to be read or in which a word is to be stored.

DB45, Program Processor, FIG. 95: A signal which issues in response to a data interrupt in the Input/Output Control Unit to enable gate B05 of the B-Gates.

DBAC, Program Processor, FIG. 95: A signal which issues when the contents of the Accumulator Counter Register are 0, 1. This state of the Accumulator Counter Register, in conjunction with the contents of the A-Register, forms the address of the accumulator word B location.

DBCB, Program Processor, FIG. 95: A signal which issues during the execution of one of the Instructions 12–14 when the branch condition is not satisfied.

DBCL, Program Processor, FIG. 95: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions of the Branch Class, viz Instructions 10–17.

DBCT, Program Processor, FIG. 95: A signal which issues during the execution of Instruction 10, or during the execution of one of the Instructions 12–14 when the branch condition is satisfied. This signal causes working clock signal QSPX to issue, transferring the branch address to the P-Register.

DBCU, Program Processor, FIG. 95: A signal which advances the count in the TB-Counter during the execution of predetermined instructions, e.g. Instruction 26, Variable Length Divide (VLD).

DBL0, Program Processor, FIG. 95: A signal which issues at predetermined times and during the execution of predetermined instructions to change the state of the TL-Counter from TL2 to TL0.

DBL2, Program Processor, FIG. 95: A signal which issues at predetermined times and during the execution of predetermined instructions to change the state of the TL-Counter from TL0 to TL2.

DBNX, Program Processor, FIG. 95: A signal which issues during the execution of one of the Instructions 12–14 when the branch condition is not satisfied and development of the address field of the instruction word is completed. This signal causes flip-flops ESX and MSX to be reset to the 0-state.

DBRC, Program Processor, FIG. 95: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to Instruction 16, Branch on Count (BRC).

DBRE, Program Processor, FIG. 95: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to Instruction 13, Branch if Equal (BRE).

DBRF, Program Processor, FIG. 95: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the Instructions 12–14.

DBRG, Program Processor, FIG. 95: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to Instruction 12, Branch if Greater (BRG).

DBRL, Program Processor, FIG. 95: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to Instruction 14, Branch if Less (BRL).

DBRM, Program Processor, FIG. 95: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to Instruction 11, Branch if Minus (BRM).

DBRP, Memory, FIG. 36: A signal which resets the flip-flops of the B-Register to the 0-state upon completion of a memory cycle and during the initial clearing of the Central Processor.

DBRR, Memory, FIG. 36: A signal which issues during time period T01 of a read operation and during time period T07 of a write operation.

DBRS, Memory, FIG. 36: A signal which controls the transfer of an address from the B-Gates of the Program Processor to the B-Register of a 16K memory unit at the beginning of a memory cycle.

DBRS, Program Processor, FIG. 95: A signal which issues in block W01 during development of the second address of Instruction 07, General (GEN), to check the busy status of the addressed channel.

DBRU, Program Processor, FIG. 95: A signal which issues when the operation code of the I-Register has a bit configuration corresponding to Instruction 10, Branch Unconditionally (BRU).

DBRZ, Program Processor, FIG. 95: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to Instruction 15, Branch if Zero (BRZ).

DBSA, Program Processor, FIG. 95: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to either Instruction 16, Branch on Count (BRC), or Instruction 17, Store Program Counter and Branch (SPB).

DBUD, Program Processor, FIG. 95: A signal which issues during the execution of predetermined instructions involving a four-word Accumulator and which causes flip-flop AC2 of the Accumulator Counter Register to be reset to the 0-state.

DBXM, Program Processor, FIG. 95: A signal which issues when an index has been obtained for addition to the instruction address field during the execution of Instruction 10 or during the execution of one of the Instructions 12–14, if the branch condition is satisfied. This signal causes flip-flop MSX to be reset to the 0-state.

DBYD, Program Processor, FIG. 95: A signal which issues during the performance of a shift operation to set flip-flop AC1 of the Accumulator Counter Register to the 1-state, if the operation involves the triple Accumulator, and to also set flip-flop AC2 to the 1-state, if the operation involves the triple or quadruple Accumulator, reducing the count in the Accumulator Counter Register by two.

DBYU, Program Processor, FIG. 95: A signal which issues during a right shift when the accumulator length affected by the shift is triple to set flip-flop AC2 of the Accumulator Counter Register to the 1-state, advancing the count in the Accumulator Counter Register by two.

DCAA, Program Processor, FIG. 96: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to Instruction 03, Compare Alphanumeric Accumulator to Memory (CAA).

DCAB, Program Processor, FIG. 96: A signal which issues when flip-flops CC1 and CC0 of the CC-Register are set to the 1-state.

DCAC, Program Processor, FIG. 96: A signal which issues when the contents of the Accumulator Counter Register are 1, 0. This state of the Accumulator Counter Register, in conjunction with the contents of the A-Register, forms the address of the accumulator word C location.

DCAD, Program Processor, FIG. 96: A signal which changes the states of the flip-flops of the Accumulator Counter Register to reduce the count by one. If the count in the Accumulator Counter Register is 0, signal DCAD changes the count so that the contents of the Accumulator Counter Register, in conjunction with the contents of the A-Register, forms the address of the most-significant accumulator word affected by the instruction.

DCAU, Program Processor, FIG. 96: A signal which changes the states of the flip-flops of the Accumulator Counter Register to advance the count by one. If the count in the Accumulator Counter Register, in conjunction with the contents of the A-Register, forms the address of the most-significant accumulator word affected by the instruction, DCAU reduces the count in the Accumulator Counter Register to zero.

DCCU, Program Processor, FIG. 96: A signal which issues at predetermined times and during the execution of predetermined instructions to advance the count in the CC-Register by one.

DCCZ, Program Processor, FIG. 96: A signal which indicates that all the flip-flops of the CC-Register are reset to the 0-state.

DCDA, Program Processor, FIG. 96: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to Instruction 02, Compare Decimal Accumulator to Memory (CDA).

DCE1, Program Processor, FIG. 96: A signal which issues when the count in the CC-Register is 11.

DCEE, Program Processor, FIG. 96: A signal which issues when the operation code in I-Register has a bit configuration corresponding to one of the Instructions 02, 03, 05, 15, 20 and 21.

DCEL, Program Processor, FIG. 96: A signal which issues during the execution of predetermined instructions to indicate that the count in the CC-Register is equal to the number of character positions of the Accumulator affected by the instruction, i.e. when the count in the CC-Register is four times the number of accumulator words affected by the instruction.

DCER, Program Processor, FIG. 96: A signal which issues at predetermined times and during the execution of predetermined instructions to cause working clock signal QCER to issue, complementing the contents of the E-Register.

DCES, Program Processor, FIG. 96: A signal which issues at predetermined times during the execution of predetermined instructions to cause working clock signal QCER to issue, forming the complement of the word in the E-Register.

DCID, Program Processor, FIG. 96: A signal which issues during execution of Instruction 26, Variable Length Divide (VLD), to cause flip-flop MPL to be set to the 1-state.

DCL4, Program Processor, FIG. 96: A signal which issues during the execution of Instruction 22, Shift (SHG), when the direction of shift is left and when the number of character positions through which the Accumulator is to be shifted is less than four. This signal establishes a count of four minus N in the CC-Register, where N is the number of character positions through which the Accumulator is to be left shifted. The count in the CC-Register then indicates the number of character positions through which each accumulator word must be right shifted in the M- and E-Registers to be equivalent to a left shift through the number of character positions required by the instruction.

DCLA, Program Processor, FIG. 96: A signal which issues at predetermined times and during the execution of predetermined instructions to cause the flip-flops of the A-Register to be reset to the 0-state.

DCLC, Program Processor, FIG. 96: A signal which issues at predetermined times and during the execution of predetermined instructions to cause the flip-flops of the CC-Register to be reset to the 0-state.

DCLM, Memory, FIG. 36: A signal which issues during time period T02 of a read operation to reset the flip-flops of the M-Register to the 0-state.

DCLQ, Program Processor, FIG. 96: A signal which issues at predetermined times and during the execution of predetermined instructions to cause the flip-flops of the Q-Register to be reset to the 0-state.

DCM0, Program Processor, FIG. 96:A signal which issues at predetermined times and during the execution of predetermined instructions to cause flip-flops M18–M23 of the M-Register to be reset to the 0-state.

DCM1, Program Processor, FIG. 96: A signal which issues at predetermined times and during the execution of predetermined instructions to cause flip-flops M12–M17 of the M-Register to be reset to the 0-state.

DCM2, Program Processor, FIG. 96: A signal which issues at predetermined times and during the execution of predetermined instructions to cause flip-flops M06–M11 of the M-Register to be reset to the 0-state.

DCM3, Program Processor, FIG. 96: A signal which issues at predetermined times and during the execution of predetermined instructions to cause flip-flops M00–M05 of the M-Register to be reset to the 0-state.

DCM4, Program Processor, FIG. 96: A signal which issues at predetermined times and during the execution of predetermined instructions to cause Memory Output Parity flip-flop MUJ to be reset to the 0-state.

DCMC, Program Processor, FIG. 97: A signal which issues at predetermined times and during the execution of either Instruction 67 or Instruction 07 to cause the flip-flops of the M-Register to be reset to the 0-state.

DCMI, Program Processor, FIG. 97: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 01, Compare Memory to Immediate (CMI).

DCMM, Program Processor, FIG. 97: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 04, Compare Second to First Memory (CMM).

DCPG, Program Processor, FIG. 97: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to either Instruction 03 or Instruction 02.

DCPO, Program Processor, FIG. 97: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 67, Central Processor Operations (CPO).

DCSL, Program Processor, FIG. 97: A signal which issues during the execution of Instruction 22, Shift (SHG), if the direction of shift is left.

DCTV, Program Processor, FIG. 97: A signal which issues when the count in the CC-Counter is 12 or greater.

DCW9, Program Processor, FIG. 97: A signal which issues during the execution of either Instruction 02 or Instruction 03, if the Program Processor is performing the micro-operations of block W09.

DD00–DD13, Program Processor, FIG. 97: Signals which issue when the corresponding flip-flops D00–D13 respectively of the D-Register are set to the 1-state.

DDAA, Program Processor, FIG. 97: A signal which issues during the divide operation of Instruction 26, Variable Length Divide (VLD), when the A-Register and the Accumulator Counter Register contain the address of accumulator word A.

DDAC, Program Processor, FIG. 97: A signal which issues when the contents of the Accumulator Counter Register are 1, 1. This state of the Accumulator Counter Register, in conjunction with the contents of the A-Register, forms the address of the accumulator word D location.

DDBI, Program Processor, FIG. 97: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions of the group comprising Instructions 01, 05, 11, 15, 21, 25, 31, 35, 41, 45, 51, 55 and 61.

DDBO, Program Processor, FIG. 98: A signal which issues during the execution of predetermined instructions employing the two least-significant accumulator words.

DDBX, Program Processor, FIG. 98: A signal which issues at predetermined times and during the execution of predetermined instructions to cause the contents of the D-Register to be applied to the B-Gates.

DDC0, Program Processor, FIG. 98: A signal which issues to indicate a carry from character 0 of the adder output and which sets Carry Remember flip-flop CRE to the 1-state during the execution of predetermined instructions.

DDC1, Program Processor, FIG. 98: A signal which issues during a decimal operation in the Adder indicating a carry from character 1 of the adder output.

DDC2, Program Processor, FIG. 98: A signal which issues during a decimal operation in the Adder which indicates a carry from character 2 of the adder output.

DDC3, Program Processor, FIG. 98: A signal which issues during a decimal operation in the Adder which indicates a carry from character 3 of the adder output.

DDCA, Program Processor, FIG. 98: A signal which issues during execution of Instruction 26, Variable Length Divide (VLD), when the A-Register and the the Accumulator Counter Register do not contain the address of accumulator word C.

DDCD, Program Processor, FIG. 98: A signal which issues at predetermined times and during the execution of predetermined instructions to reduce the count in the D-Register by one.

DDCS, Program Processor, FIG. 98: A signal which issues during a decimal operation to enable the decimal correction circuits of the Adder.

DDDA, Program Processor, FIG. 98: A signal which issues during performance of the divide operation of Instruction 26, Variable Length Divide (VLD), when the A-Register and Accumulator Counter Register contain the address of accumulator word D.

DDES, Program Processor, FIG. 98: A signal which issues during the execution of Instruction 22, Shift (SHG), to identify an alpha or a decimal shift or during the performance of the shift operation of Instruction 26, Variable Length Divide (VLD).

DDEV, Program Processor, FIG. 98: A signal which issues during execution of Instruction 07, General (GEN), during block W11 to define the time during which the device code is transmitted from the E-Register to the Input/Output Control Unit to select the peripheral device, for example, to select one of several tape units connected to the selected channel.

DDFS, Program Processor, FIG. 98: A signal which issues during the execution of Instruction 22, Shift (SHG), if the direction of shift is left and if the shift is a decimal or alpha shift.

DDLS, Program Processor, FIG. 98: A signal which issues during a decimal add operation when the signs of the operand words are the same or during a decimal subtract operation when the signs of the operand words are different, to enable the decimal correction circuits of the Adder.

DDLX, Program Processor, FIG. 98: A signal which issues during execution of Instruction 22, Shift (SHG), to cause the length of the Accumulator affected by the shift, as specified by the instruction word, to be set into the Accumulator Length Indicator Register flip-flops.

DDNA, Program Processor, FIG. 98: A signal which issues during performance of the divide operation of Instruction 26, Variable Length Divide (VLD), to indicate that the A-Register and the Accumulator Counter Register contain an accumulator word address other than that of accumulator word A.

DDOM, Program Processor, FIG. 98: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to Instruction 26, Variable Length Divide (VLD), or Instruction 27, Variable Length Multiply (VLM).

DDST, Program Processor, FIG. 98: A signal which issues to gate the contents of the D-Register to the Y-input terminals of the first rank of full adders.

DDUS, Program Processor, FIG. 98: A signal which issues during a decimal add operation when the signs of the operand words are different or during a decimal subtract operation when the signs of the operand words are the same, to enable the decimal correction circuits of the Adder.

DDVL, Program Processor, FIG. 98: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 26, Variable Length Divide (VLD).

DDVS, Program Processor, FIG. 98: A signal which issues during execution of Instruction 26, Variable Length Divide (VLD), to indicate that the Program Processor is performing the shift operation of the Instruction.

DDXB, Program Processor, FIG. 98: A signal which issues at predetermined times and during the execution of predetermined instructions to cause the address in the D-Register to be applied to the B-Gates of Memory.

DEAP, Program Processor, FIG. 98: A signal which issues at predetermined times during the execution of either Instruction 06, Move (MOV), or Instruction 36, Load Accumulator Location and Length (LAL), to cause working clock signal QEAX to issue, transferring the contents of flip-flops E02–E14 of the E-Register to the A-Register and transferring the contents of flip-flops E00 and E01 to the Accumulator Length Indicator Register flip-flops PL1 and PL2.

DECH, Program Processor, FIG. 99: A signal which issues during an input/output operation to select one of eight channels in the Input/Output Control Unit.

DEDF, Program Processor, FIG. 99: A signal which issues during execution of Instruction 07, General (GEN), when development, if any, of the first address is complete and the selected channel is not busy.

DEDT, Program Processor, FIG. 99: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 05, Edit (EDT).

DEEI, Program Processor, FIG. 99: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the Instructions 05, 20 or 21.

DEEZ, Program Processor, FIG. 99: A signal which issues when flip-flops E00–E05 of the E-Register are all reset to the 0-state, indicating that character 3 of the word in the E-Register is a zero.

DEI0, Program Processor, FIG. 99: A signal which issues at predetermined times and during the execution of predetermined instructions to indicate that flip-flop WR0 of the W-Selector Register will be reset to the 0-state.

DEI1, Program Processor, FIG. 99: A signal which issues at predetermined times and during the execution of predetermined instructions to indicate that flip-flop WR1 of the W-Selector Register will be reset to the 0-state.

DEI2, Program Processor, FIG. 99: A signal which issues at predetermined times and during the execution of predetermined instructions to indicate that flip-flop WR2 of the W-Selector Register will be reset to the 0-state.

DEI3, Program Processor, FIG. 99: A signal which issues at predetermined times and during the execution of predetermined instructions to indicate that flip-flop WR3 of the W-Selector Register will be reset to the 0-state.

DEIN, Program Processor, FIG. 99: A signal which issues when all the flip-flops of the W-Selector Register are reset to the 0-state or when an invalid instruction has been transferred to Memory, to indicate that the Program Processor is to next perform the micro-operations of block W00.

DELS, Program Processor, FIG. 99: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to Instruction 05, Edit (EDT), or one of the instructions in the Load and Store Class, viz. Instructions 40–47.

DERM, Program Processor, FIG. 99: A signal which indicates that the sign of the word in the E-Register is minus.

DERS, Program Processor, FIG. 99: A signal which issues upon actuation of an appropriate console switch during the manual mode to cause the typewriter to type out the contents of the E-Register.

DESA, Program Processor, FIG. 99: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to either Instruction 37, Store Accumulator Location and Length (SAL), or Instruction 20, Explode (EXP).

DESC, Program Processor, FIG. 99: A signal which causes the contents of the E-Register to be applied to the Adder when the contents of either the A-Register or the E-Register are to be typed out by the typewriter during the manual mode of operation.

DESP, Program Processor, FIG. 99: A signal which issues in block W03 during a program interrupt to one of the channels of the Input/Output Control Unit.

DESS, Program Processor, FIG. 99: A signal which issues at predetermined times and during the execution of predetermined instructions to set flip-flop ESX to the 1-state, applying the contents of the E-Register to the adder inputs.

DEST, Program Processor, FIG. 99: A signal which issues at predetermined times and during the execution of predetermined instructions to gate the contents of the E-Register to the Y-input terminals of the first rank of full adders.

DEXL, Program Processor, FIG. 99: A signal which issues during the execution of Instruction 20, Explode (EXP), when the last accumulator word is being processed during execution of the instruction. Signal DEXL sets flip-flop PSX to the 1-state to apply the address of the next instruction in the P-sequence to the Adder.

DEXP, Program Processor, FIG. 99: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 20, Explode (EXP).

DF00–DF14, Program Processor, FIGS. 99–100: Signals which issue when the respective states of flip-flops D00–D14 of the D-Register are the same as the states of the corresponding flip-flops AC1 and AC2 of the Accumulator Counter Register and flip-flops A02–A14 of the A-Register.

DFBS, Program Processor, FIG. 100: A signal which issues at predetermined times and during execution of predetermined instructions to enable gate B03 of the B-Gates, addressing memory location 10 (octal).

DFDE, Program Processor, FIG. 100: A signal which issues during execution of Instruction 05, Edit (EDT), during the edit mode when the last character of the data word in the E-Register has been processed but additional format characters remain to be processed. Signal DFDE causes the Program Processor to perform the micro-operations of the W03S block upon completion of the W14S block.

DFXW, Program Processor, FIG. 100: A signal which issues when the address control field of the instruction word has a value in the range of 1–6, indicating that Fixed Location Index Word is to be used to modify the instruction address field, if the instruction is a single-address instruction, or that the second address of the instruction is the address of a Fixed Location Index Word, if the instruction is a two-address instruction.

DGCP, Program Processor, FIG. 100: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to Instruction 67, Central Processor Operations (CPO), or Instruction 07, General Instruction (GEN).

DGE1, Program Processor, FIG. 100: A signal which issues during execution of Instruction 22, Shift (SHG), or during the shift operation of Instruction 26, Variable Length Divide (VLD), to cause the Parity Check and Generation Logic to generate parity for the contents of the M-Register.

DGE2, Program Processor, FIG. 100: A signal which issues during the execution of Instruction 26, Variable Length Divide (VLD), to cause the Parity Check and Generation Logic to generate parity for the contents of the M-Register.

DGE3, Program Processor, FIG. 100: A signal which causes parity to be generated in the Parity Adder in

blocks W03 and W04 during the execution of predetermined instructions.

DGE4, Program Processor, FIG. 100: A signal which issues during execution of Instruction 05, Edit (EDT), Instruction 21, Implode (IMP), and during a left shift operation to reset flip-flop SP2 to the 0-state.

DGEN, Program Processor, FIG. 100: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 07, General Instruction (GEN).

DGEP, Program Processor, FIG. 100: A signal which issues at predetermined times and during the execution of predetermined instructions to cause the Parity Check and Generation Logic to generate the correct parity bit for a word in the M-Register.

DGMM, Program Processor, FIG. 100: A signal which issues at predetermined times during execution of predetermined instructions to transfer data from the input/output matrix to the M-Register or to transfer signals representative of the states of predetermined indicator flip-flops to the M-Register during execution of Instruction 67, Central Processor Operations (CPO).

DGPP, Program Processor, FIG. 100: A signal which issues at predetermined times during the execution of one of the Instructions 22, 26 or 27 to permit partial parity for the partial word in the M-Register to be stored in flip-flop SP2.

DHLT, Program Processor, FIG. 100: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 00, Halt (HLT).

DHSM, Program Processor, FIG. 100: A signal which issues during the execution of Instruction 00, Halt (HLT), when an index has been obtained during time period TL4 of block W01 for developing the instruction address field. Signal DHSM causes working clock signal QSEX to issue.

DHXM, Program Processor, FIG. 101: A signal generated during development of the address field of Instruction 00, Halt (HLT), in block W01, when an index has been obtained for developing the instruction address field.

DIAD, Memory, FIG. 36: A signal which issues in response to an address transferred from the B-Gates which exceeds the memory capacity. This signal does not issue in a system with a 32K memory.

DIAM, Program Processor, FIG. 101: A signal which issues when an instruction word contains an invalid operation code.

DICR, Program Processor, FIG. 101: A signal which issues upon actuation of the console Power-On switch to initially reset certain control flip-flops, e.g. RCK, ISP, to the 0-state.

DIFP, Program Processor, FIG. 101: A signal which issues in block W03 during the execution of predetermined instructions and under predetermined conditions during an input/output operation to inhibit signal DAMR, preventing the transfer of the contents of the addressed memory location into the M-Register.

DIG3, Program Processor, FIG. 101: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions of Instruction Group 3, viz. 01, 04, 06, 07, 16, 17, 23, 24, 25, 30–35 and 67.

DIG4, Program Processor, FIG. 101: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions of Instruction Group 4, viz. Instructions 16, 17, 23–25, 30, 31, 33, 34 and 35.

DIG5, Program Processor, FIG. 101: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions of Instruction Group 5, viz. Instructions 01, 07, 16, 17, 31–33 and 67.

DIGA, Program Processor, FIG. 101: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions in Instruction Group A, viz. Instructions 00, 10, 20, 30, 40, 50 and 60.

DIGB, Program Processor, FIG. 101: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions in Instruction Group B, viz. Instructions 01, 11, 21, 31, 41, 51 and 61.

DIGC, Program Processor, FIG. 101: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions in Instruction Group C, viz. Instructions 02, 12, 22, 32, 42, 52 and 62.

DIGD, Program Processor, FIG. 101: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions in Instruction Group D, viz. Instructions 03, 13, 23, 33, 43, 53 and 63.

DIGE, Program Processor, FIG. 101: A signal which issues when the operation code in the I-Register has a bit configuration which corresponds to one of the instructions in Instruction Group E, viz. Instructions 04, 14, 24, 34, 44 and 54.

DIGF, Program Processor, FIG. 101: A signal which issues when the operation code in the I-Register has a bit configuration which corresponds to one of the instructions in Instruction Group F, viz. Instructions 05, 15, 25, 35, 45 and 55.

DIGG, Program Processor, FIG. 101: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions in Instruction Group G, viz. Instructions 06, 16, 26, 36, 46 and 56.

DIGH, Program Processor, FIG. 101: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions in Instruction Group H, viz. Instructions 07, 17, 27, 37, 47, 57 and 67.

DIIP, Program Processor, FIG. 101: A signal which indicates an illegal instruction in a Program Interrupt Word (PIW). Signal DIIP sets Manual flip-flop MAN to the 1-state.

DILX, Program Processor, FIG. 101: A signal which issues at predetermined times and during the execution of one of several predetermined instructions to transfer the accumulator length specified by the instruction to the flip-flops of the Accumulator Length Indicator Register, establishing a new accumulator working length.

DIML, Program Processor, FIG. 101: A signal which issues during the execution of Instruction 21, Implode (IMP), when the last accumulator word is being processed.

DIMM, Program Processor, FIG. 101: A signal which issues during address field development in block W01 when an index word is obtained for modifying the instruction address field.

DIMP, Program Processor, FIG. 101: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 21, Implode (IMP).

DIN1, Memory, FIG. 36: A signal which issues during a write operation to activate the set of Inhibit Drivers associated with the lower 4K stack of the lower 8K unit of a 16K memory unit.

DIN2, Memory, FIG. 36: A signal which issues during a write operation to activate the set of Inhibit Drivers associated with the upper 4K stack of the lower 8K unit of a 16K memory unit.

DIN3, Memory, FIG. 36: A signal which issues during a write operation to activate the set of Inhibit Drivers associated with the lower 4K stack of the upper 8K unit of a 16K memory unit.

DIN4, Memory, FIG. 36: A signal which issues during a write operation to activate the set of Inhibit Drivers associated with the upper 4K stack of the upper 8K unit of a 16K memory unit.

DINL, Program Processor, FIG. 101: A signal which issues during execution of Instruction 21, Implode (IMP), when the accumulator word being addressed is other than the last accumulator word to be processed by the instruction.

DINP, Program Processor, FIG. 101: A signal which issues when either an invalid operation code is transferred to the I-Register or an invalid address is transferred to Memory.

DINV, Program Processor, FIG. 101: A signal which issues when an invalid operation code is transferred to the I-Register.

DIOA, Program Processor, FIG. 101: A signal which issues during the manual mode of operation to cause the contents of the A-Register or the E-Register to be typed out by the typewriter when appropriate console switches are actuated.

DIPT, Program Processor, FIG. 102: A signal which issues at predetermined times and during the execution of predetermined instructions to inhibit the transfer of parity signal DM24 to Memory.

DIRK, Program Processor, FIG. 102: A signal which issues at predetermined times and during the execution of predetermined instructions to inhibit the starting of the Program Processor Clock Generator by Memory Data Available signal DMDA from Memory.

DIRM, Program Processor, FIG. 102: A signal which issues upon actuation of the console Power-On switch to set the System Reset flip-flop SRE to the 1-state.

DIRS, Program Processor, FIG. 102: A signal which issues at predetermined times during instruction execution to indicate that the M-Register does not contain significant information and is available for data transfer with the channels of the Input/Output Control Unit. Signal DIRS defines the points at which the W-sequence can be interrupted.

DLAA, Program Processor, FIG. 102: A signal which issues during execution of Instruction 22, Shift (SHG), to indicate that the shift is a left shift and that accumulator word A is being addressed.

DLAL, Program Processor, FIG. 102: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 36, Load Accumulator Location and Length (LAL).

DLAO, Program Processor, FIG. 102: A signal which is generated when the accumulator word corresponding to the highest-order accumulator word affected by the instruction is being addressed.

DLAX, Program Processor, FIG. 102: A signal which issues during development of the address field of Instruction 36, Load Accumulator Location and Length (LAL), when an index has been obtained.

DLDG, Program Processor, FIG. 102: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions of the Load Group, viz. Instructions 40, 41, 42 and 43.

DLFT, Program Processor, FIG. 102: A signal which is generated in the Program Processor during execution of Instruction 22, Shift (SHG), when the shift is an alpha or a decimal shift and when the direction of shift is left or during the shift operation of Instruction 26, Variable Length Divide (VLD).

DLLA, Program Processor, FIG. 102: A signal which issues during execution of Instruction 22, Shift (SHG), when the direction of shift is left and when the last accumulator word to be processed is being addressed.

DLLI, Program Processor, FIG. 102: A signal which causes a double-word channel with eight characters in the buffer to gain a higher data interrupt priority.

DLSB, Program Processor, FIG. 102: A signal which issues when the direction of shift is left during the execution of Instruction 22, Shift (SHG), or during performance of the shift operation of Instruction 26, Variable Length Divide (VLD), when the length of the accumulator affected by the shift is single, in the case of Instruction 22, or when accumulator word B is being addressed, in the case of either Instruction 22 or Instruction 26.

DLSC, Program Processor, FIG. 102: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions in the Load and Store Class, viz. Instructions 40–47.

DLSL, Program Processor, FIG. 102: A signal which issues during the execution of one of the Instructions 40–47, when the last accumulator word affected by the instruction is being processed.

DLSM, Program Processor, FIG. 102: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions of the Load and Store Class, viz Instructions 40–47, or one of the instructions of the Add to Memory Group, viz Instructions 44–47.

DLSR, Program Processor, FIG. 102: A signal which issues during the execution of predetermined instructions to control the resetting of flip-flop M23 of the M-Register to the 0-state in accordance with the states of flip-flops E00 of the E-Register and/or M00 of the M-Register.

DLSS, Program Processor, FIG. 102: A signal which issues during the execution of predetermined instructions to control the setting of flip-flop M23 of the M-Register to the 1-state in accordance with the states of flip-flops E00 of the E-Register and/or M00 of the M-Register.

DM24, Program Processor, FIG. 102: A signal representing the parity bit generated for the word in the M-Register which is transmitted to the Memory Inhibit Drivers along with the word in the M-Register.

DMAD, Program Processor, FIG. 102: A signal which issues during execution of Instruction 27, Variable Length Divide (VLD), when the A-Register and the Accumulator Counter Register contain the address of accumulator word D.

DMAI, Program Processor, FIG. 102: A signal which issues when Memory Address Invalid flip-flop MAI is set to the 1-state to indicate that the address transmitted to Memory exceeds the memory capacity.

DMBU, Program Processor, FIG. 102: A signal which issues when the Memory Busy Level flip-flop MBU in memory is set to the 1-state to indicate that the Memory is performing a read, clear, write or restore operation.

DMDA, Memory, FIG. 36: A signal which issues during a read operation to indicate to the Program Processor that the word read from Memory is available.

DMDC, Program Processor, FIG. 102: A signal which is generated during the performance of the micro-operations of block W07 if a carry is generated from character 0 of the adder output.

DMEC, Program Processor, FIG. 102: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions of the Move Class, viz. Instructions 30–37.

DMFI, Program Processor, FIG. 102: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 31, Move From Immediate (MFI).

DMFM, Program Processor, FIG. 103: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 30, Move From First Memory (MFM).

DMIC, Program Processor, FIG. 103: A signal which is generated in the Program Processor upon completion of development of the second address of Instruction 06, Move (MOV).

DMIR, Program Processor, FIG. 103: A signal which issues during the execution of Instruction 67, Central Processor Operations (CPO), if a "request status of processor" operation is being performed, to reset flip-flops ICW and MPI to the 0-state.

DMIS, Program Processor, FIG. 103: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions of the Miscellaneous Class which includes Instructions 00–07.

DMLR, Program Processor, FIG. 103: A signal which issues when Manual Reset flip-flop MRE is set to the 1-state to preset the TL-Counter to TL2 and to reset certain flip-flops of the Program Processor to the 0-state.

DMND, Program Processor, FIG. 103: A signal which issues during the execution of Instruction 27, Variable Length Multiply (VLM), when the address in the A-Register and the Accumulator Counter Register is that of accumulator word A, B, or C.

DMNS, Program Processor, FIG. 103: A signal which initiates operation of the typewriter during the manual mode when an appropriate console switch is actuated.

DMOV, Program Processor, FIG. 103: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 06, Move (MOV).

DMPR, Program Processor, FIG. 103: A signal which issues upon actuation of the control console Memory Parity Alert switch to reset Word Parity Error flip-flop WJE to the 0-state.

DMQX, Program Processor, FIG. 103: A signal which issues during time period TL3 of the W00 block of micro-operations to cause the contents of flip-flops M15–M17 of the M-Register to be transferred to the Q-Register.

DMRC, Program Processor, FIG. 103: A signal which issues in block W10S to reset predetermined flip-flops of the Program Processor during execution of Instruction 00, Halt (HLT), or when an invalid address is transmitted to Memory during an input/output operation, to set Manual flip-flop MAN to the 1-state.

DMRM, Program Processor, FIG. 103: A signal which indicates that the sign of the word in the M-Register is minus.

DMRW, Memory, FIG. 36: A signal which issues in response to a read or a write command from the Program Processor.

DMSE, Program Processor, FIG. 103: The signal which issues at predetermined times to set Manual flip-flop MAN to the 1-state.

DMSS, Program Processor, FIG. 103: A signal which issues at predetermined times and during the execution of predetermined instructions to cause flip-flop MSX to be set to the 1-state, applying the contents of the M-Register to the adder inputs.

DMST, Memory, FIG. 36: A signal which issues in response to a read or a write command from the Program Processor and which initiates the generation of memory clock pulses by the Clock Generator of the selected 16K memory unit.

DMTA, Program Processor, FIG. 103: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 32, Move To First Address Field (MTA).

DMVC, Program Processor, FIG. 103: A signal which issues during execution of Instruction 27, Variable Length Multiply (VLM), when the address in the A-Register and the Accumulator Counter Register contain the address of accumulator word D. Signal DMVC causes working clock signal QSCX to issue.

DMVD, Program Processor, FIG. 103: A signal which issues during time period TL1 of block W15 during execution of Instruction 06, Move (MOV).

DMVE, Program Processor, FIG. 103: A signal which issues during time period TL2 of block W06 during execution of Instruction 06, Move (MOV).

DMVF, Program Processor, FIG. 103: A signal which issues during time period TL1 of block W02 during execution of Instruction 06, Move (MOV).

DMVL, Program Processor, FIG. 103: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 27, Variable Length Multiply (VLM).

DMWS, Program Processor, FIG. 103: A signal which issues during predetermined blocks of micro-operations to set the Inhibit Address Register flip-flop IAR in Memory, inhibiting resetting of the B-Register flip-flops to the 0-state at the end of a read cycle and inhibiting the transfer of a new address into the B-Register at the start of the write cycle which follows.

DMWT, Memory, FIG. 36: A signal which issues in response to the write command from the Program Processor to initiate a write-only cycle.

DMXR, Program Processor, FIG. 103: A signal which issues at predetermined times and during the execution of predetermined instructions to cause flip-flop MSX to be reset to the 0-state, inhibiting the transfer of information from the M-Register to the adder inputs.

DNAI, Program Processor, FIG. 103: A signal which issues during development of the instruction address field to indicate that the next word to be read from Memory is an Indirect AMS Word.

DNAT, Program Processor, FIG. 104: A signal which issues to indicate that the count in the N-Register is at least 8.

DNCS, Program Processor, FIG. 104: A signal which issues at predetermined times and during the execution of predetermined instructions to cause working clock signal QNCD to issue, reducing the count in the N-Register by 1.

DNE1, Program Processor, FIG. 104: A signal which issues to indicate that the count in the N-Register is 1.

DNE4, Program Processor, FIG. 104: A signal which issues during the execution of Instruction 05, Edit (EDT), when the count in the N-Register is 4, 8 or 12.

DNEZ, Program Processor, FIG. 104: A signal which issues when the count in the N-Register is 0.

DNL4, Program Processor, FIG. 104: A signal which issues during execution of Instruction 22, Shift (SHG), when the count in the N-Register is less than 4, if a decimal or alpha shift, or is less than 24, if a binary shift.

DNMF, Program Processor, FIG. 104: A signal which issues when flip-flops N00 and N01 of the N-Register are reset to the 0-state.

DNRO, Program Processor, FIG. 104: A signal which is generated in the Program Processor to reset flip-flops N00 and N01 of the N-Register to the 0-state.

DNS0, Program Processor, FIG. 104: A signal which issues during execution of Instruction 21, Implode (IMP), which causes all of the flip-flops of the N-Register to be reset to the 0-state.

DNS2, Program Processor, FIG. 104: A signal which issues during execution of predetermined instructions employing the three least-significant accumulator words when the A-Register and the Accumulator Counter Register contain the address of accumulator word A. Signal DNS2 presets the N-Register to a count of 12.

DNS4, Program Processor, FIG. 104: A signal which issues during the execution of predetermined instructions employing only accumulator word A which causes the N-Register to be preset to a count of 4.

DNS6, Program Processor, FIG. 104: A signal which issues during the execution of predetermined instruc-

tions employing the full Accumulator when the A-Register and the Accumulator Counter Register contain the address of accumulator word A, causing the N-Register to be preset to a count of 16.

DNS8, Program Processor, FIG. 104: A signal which issues during the execution of predetermined instructions employing the 2 least-significant accumulator words when the A-Register and the Accumulator Counter Register contain the address of accumulator word A, presetting the N-Register to a count of 8.

DNSL, Program Processor, FIG. 104: A signal which is generated during execution of one of the Instructions 23–25 to preset the flip-flops of the N-Register to a count of 24.

DNTS, Program Processor, FIG. 104: A signal which issues when flip-flops N02 and N03 of the N-Register are reset to the 0-state.

DNWX, Program Processor, FIG. 104: A signal generated by the Program Processor when the value of the address control field of an instruction word is 7, indicating that the address field of the instruction word is to be developed by employing a P-Sequence AMS Word.

DNZE, Program Processor, FIG. 104: A signal which issues during execution of Instruction 05, Edit (EDT), when the count in the N-Register is 0.

DP07, Memory, FIG. 36: A signal which issues during time period T07 at the beginning of a write operation or in response to an initial clear signal from the Program Processor.

DPCU, Program Processor, FIG. 104: A signal which issues at predetermined times during performance of the initial routine to advance the count to the P-Register by 1, forming the address of the next word in the P-Sequence.

DPDC, Program Processor, FIG. 104: A signal which indicates that a program interrupt request has been transmitted by either the Input/Output Control Unit or the Program Processor.

DPGI, Program Processor, FIG. 104: A signal which issues during execution of one of the instructions of the Add to Memory Group, viz. Instructions 54–57, when the working length of the Accumulator, as specified by the Accumulator Length Indicator Register, is greater than the accumulator length specified by the instruction.

DPIO, Program Processor, FIG. 104: A signal which indicates that a normal program interrupt request has been transmitted by one of the channels of the Input/Output Control Unit.

DPIB, Program Processor, FIG. 104: A signal which issues during a program interrupt to address a predetermined memory location to obtain a Program Interrupt Word (PIW).

DPMS, Program Processor, FIG. 104: A signal which issues upon actuation of an appropriate console switch to enable information exchange between the typewriter and the P-Register.

DPNQ, Program Processor, FIG. 104: A signal which issues when the working accumulator length stored in the Accumulator Length Indicator Register is less than quadruple.

DPSP, Program Processor, FIG. 104: A signal which issues upon completion of a program interrupt to preset the flip-flops of the program interrupt scanner (flip-flops FPS0–FPS2) to a count of one.

DPSR, Program Processor, FIG. 105: A signal which issues during either instruction execution or an input/output operation to enable the contents of the P-Register to be applied to the Adder.

DPST, Program Processor, FIG. 104: A signal which issues at predetermined times and during the execution of predetermined instructions to apply the contents of the P-Register to the adder inputs.

DPSU, Program Processor, FIG. 105: A signal which issues while the Program Processor is executing Instruction 17, Store Program Counter and Branch (SPB), during a program interrupt to set flip-flops PIT and PSA to the 1-state.

DQBX, Program Processor, FIG. 105: A signal which issues during operations employing a Fixed Location Index Word for transferring the contents of flip-flops Q00–Q02 of the Q-Register through the B-Gates to the B-Register of Memory.

DQDI, Program Processor, FIG. 105: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the Quadruple Accumulator instructions, viz. Instructions 03, 13, 23, 33, 43, 53, 63, 07, 17, 27, 37, 47, 57 and 67.

DQDO, Program Processor, FIG. 105: A signal which indicates that the instruction being performed requires that the working length of the Accumulator be quadruple. Signal DQDO issues during the execution of Instructions 43, 47, 53, 57, 63, 22, 26, 27 and during the execution of Instructions 02, 03, 05, 15, 20, 21, 50–53 and 60–63, when the working length of the Accumulator is set to quadruple.

DQE5, Program Processor, FIG. 105: A signal which issues when the count in the Q-Register is equal to 5.

DR00–DR05, Input/Output Control Unit, FIG. 194: Signals which represent 6 of the 8 possible states of the R-Selector Register to define the corresponding blocks of input/output micro-operations.

DR13, Input/Output Control Unit, FIG. 194: A signal which issues when the flip-flops of the R-Selector Register are preset to define either block R01 or R03.

DRBG, Program Processor, FIG. 105: A signal which issues at predetermined times to reset the flip-flops of the TB-Counter to the 0-state.

DRCF, Program Processor, FIG. 105: A signal which issues during the execution of Instruction 67, Central Processor Operations (CPO), if a "request status of processor" operation is being performed, to reset control flip-flop PEF to the 0-state.

DRE4, Program Processor, FIG. 105: A signal which issues during execution of one of the Instructions 22, Shift (SHG), or 26, Variable Length Divide (VLD), to cause adder output signals $\overline{DS55}$ and $\overline{DS54}$ to be binary 0's, presetting the zone bits of character 3 of the adder output to binary 0's.

DRE5, Program Processor, FIG. 105: A signal which issues during execution of Instruction 05, Edit (EDT), to reset flip-flops E04 and E05 of the E-Register to the 0-state, presetting the zone bits of character 3 of the word in the E-Register to binary 0's.

DRES, Memory, FIG. 36: A signal which issues in response to an initial clear signal from the Program Processor to reset certain memory flip-flops to the 0-state.

DRIM, Program Processor, FIG. 105: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 23, OR Inclusive To Memory (RIM).

DRPI, Program Processor, FIG. 105: A signal which is transmitted to a selected channel to reset the program interrupt receiver of the selected channel.

DRSP, Program Processor, FIG. 105: A signal which is transmitted to a selected channel upon completion of a program interrupt to reset the special program interrupt receiver of the selected channel.

DRW0, Program Processor, FIG. 105: A signal which issues at predetermined times and during the execution of predetermined instructions to reset flip-flop WR0 of the W-Selector Register to the 0-state.

DRW1, Program Processor, FIG. 105: A signal which issues at predetermined times and during the execution of predetermined instructions to reset flip-flop WR1 of the W-Selector Register to the 0-state.

DRW2, Program Processor, FIG. 105: A signal which issues at predetermined times and during the execution of predetermined instructions to reset flip-flop WR2 of the W-Selector Register to the 0-state.

DRW3, Program Processor, FIG. 105: A signal which issues at predetermined times during the execution of predetermined instructions to reset flip-flop WR3 of the W-Selector Register to the 0-state.

DRXM, Program Processor, FIG. 105: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 25, OR Exclusive To Memory (RXM).

$\overline{DS50}$, Program Processor, FIG. 105: A signal which represents the output of full adder S00.

$\overline{DS54}$, $\overline{DS55}$, Program Processor, FIGS. 105, 106: Signals which represent the zone bits of character 3 of the adder output.

$\overline{DS60}$, Program Processor, FIG. 106: A signal which represents the fifth bit of character 2 in the output of the Adder; this signal is always a binary 0 during a decimal operation in the Adder.

$\overline{DS61}$, Program Processor, FIG. 106: A signal which represents the sixth bit of character 2 in the output of the Adder; this signal is always a binary 0 during a decimal operation in the Adder.

$\overline{DS66}$, Program Processor, FIG. 106: A signal which represents the fifth bit of character 1 in the output of the Adder; this signal is always a binary 0 during a decimal operation in the Adder.

$\overline{DS67}$, Program Processor, FIG. 106: A signal which represents the sixth bit of character 1 in the output of the Adder; this signal is always a binary 0 during a decimal operation in the Adder.

$\overline{DS72}$, Program Processor, FIG. 106: A signal which represents the fifth bit of character 0 in the output of the Adder; this signal is always a binary 0 during a decimal operation in the Adder.

$\overline{DS78}$, Program Processor, FIG. 106: A signal which represents the sixth bit of character 0 in the output of the Adder; this signal is always a binary 0 during a decimal operation in the Adder.

DSAL, Program Processor, FIG. 106: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 37, Store Accumulator Location and Length (SAL).

DSAS, Program Processor, FIG. 106: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the Instructions 50, 54 or 60.

DSB0, Program Processor, FIG. 106: A signal which issues during an unlike-signs addition in the Adder to cause six to be subtracted from character 0 of the result, if the result is not greater than nine, in order to correct the result.

DSB1, Program Processor, FIG. 106: A signal which issues during an unlike-signs addition in the Adder to cause six to be subtracted from character 1 of the result, if character 1 is not greater than nine, to correct the result.

DSB2, Program Processor, FIG. 106: A signal which issues during an unlike-signs addition in the Adder to cause six to be subtracted from character 2 of the result, if character 2 is not greater than nine, in order to correct the result.

DSB3, Program Processor, FIG. 106: A signal which issues during an unlike-signs addition in the Adder to cause six to be subtracted from character 3 of the result, if character 3 is not greater than nine, in order to correct the result.

DSBC, Program Processor, FIG. 106: A signal which issues when the A-Register and the Accumulator Counter Register contain the address of accumulator word B or when the accumulator length employed by predetermined instructions is single.

DSBL, Program Processor, FIG. 106: A signal which issues at predetermined times during the execution of one of the Instructions 22–25.

DSBM, Program Processor, FIG. 106: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 35, Subtract Binary From Memory (SBM).

DSCA, Program Processor, FIG. 106: A signal which issues during execution of Instruction 67, Central Processor Operations (CPO), if bit position 0 of the instruction word contains a binary 1. Signal DSCA causes predetermined indicator flip-flops to be set to the 1-state if corresponding bit positions of the status word contain binary 1's.

DSCZ, Program Processor, FIG. 106: A signal which issues during execution of Instruction 67, Central Processor Operations (CPO), if bit position 1 of the instruction word contains a binary 1. Signal DSCZ causes predetermined indicator flip-flops to be reset to the 0-state if corresponding bit positions of the status word contain binary 0's.

DSDG, Program Processor, FIG. 106: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions of the Subtract Decimal Group, viz. Instructions 60–63.

DSDT, Program Processor, FIG. 106: A signal which issues at predetermined times and during the execution of predetermined instructions to cause working clock signal QSDX to issue, transferring the adder output to the D-Register.

DSE4, Program Processor, FIG. 106: A signal which issues when each of the adder output signals is a binary 0.

DSE5, Program Processor, FIG. 107: A signal which issues when each of the adder output signals representing bits 0–14 is a binary 0.

DSE9, Program Processor, FIG. 107: A signal which issues when each of the adder output signals representing bits 0–18 is a binary 0.

DSEG, Program Processor, FIG. 107: A signal which issues during execution of Instruction 07, General (GEN), to enable the channel specified by the instruction word to be selected.

DSES, Program Processor, FIG. 107: A signal which issues at predetermined times during execution of predetermined instructions to set flip-flop ESX to the 1-state, applying the contents of the E-Register to the Adder.

DSGI, Program Processor, FIG. 107: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the Single Accumulator instructions, viz. Instructions 00, 10, 20, 30, 40, 50, 60, 04, 14, 24, 34, 44 and 54.

DSGO, Program Processor, FIG. 107: A signal which indicates that the instruction being performed requires that the working length of the Accumulator be single. Signal DSGO issues during the execution of Instructions 22, 40, 44 and 54, and during the execution of Instructions 02, 03, 05, 15, 20, 21, 54 and 60, when the working length of the Accumulator is set to single.

DSGP, Program Processor, FIG. 107: A signal generated by the Start-Stop Clock Logic. Signal $\overline{DSGP}$ is employed to inhibit circulation of pulses in the Clock Generator.

DSH3–DSH0, Program Processor, FIG. 107: Signals which issue during data transfer between the M-Register and the channels of the Input/Output Control Unit to control the flow of data through the input/output matrix.

DSHC, Program Processor, FIG. 107: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions of the Shift Class, viz. Instructions 20–27.

DSHG, Program Processor, FIG. 107: A signal which

issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 22, Shift (SHG).

DSIG, Program Processor, FIG. 107: A signal which issues during the execution of Instruction 22, Shift (SHG), when the shift is a decimal shift or during the shift operation of Instruction 26, Variable Length Divide (VLD), during the first execution of Instruction 26. Signal DSIG causes the sign of the Accumulator to be preserved.

DSIL, Program Processor, FIG. 107: A signal which issues during a left shift when the A-Register and the Accumulator Counter Register contain the address of accumulator word A or during a right shift when the last accumulator word is being processed.

DSIM, Program Processor, FIG. 107: A signal which issues during a left shift when the A-Register and the Accumulator Counter Register contain the address of accumulator word B or during a right shift when the last accumulator word is being processed.

DSLF, Program Processor, FIG. 107: A signal which issues during the execution of Instruction 22, Shift (SHG), when the direction of shift is left, or during the shift operation of Instruction 26, Variable Length Divide (VLD).

DSMG, Program Processor, FIG. 107: A signal which issues during the execution of Instruction 22, Shift (SHG), when the shift is a decimal shift and the sign of the Accumulator is being stored. Signal DSMG resets flip-flop LSN to the 0-state.

DSML, Program Processor, FIG. 107: A signal which issues at predetermined times and during the execution of predetermined instructions to cause working clock signals QSMA and QSMB to issue, transferring bits 0–14 of the adder output to flip-flops M00–M14 respectively of the M-Register.

DSMP, Program Processor, FIG. 107: A signal which issues when one of the Instructions 22, Shift (SHG), or 27, Variable Length Multiply (VLM), is being executed.

DSMS, Program Processor, FIG. 107: A signal which issues at predetermined times and during the execution of predetermined instructions to cause flip-flop MSX to be set to the 1-state, applying the contents of the M-Register to the adder inputs.

DSMV, Program Processor, FIG. 107: A signal which issues during execution of Instruction 06, Move (MOV), while the micro-operations of block W06 are being performed.

DSNZ, Program Processor, FIG. 107: A signal which issues during the execution of Instruction 22, Shift (SHG), when the count in the N-Register is 0.

DSPB, Program Processor, FIG. 107: A signal which issues when the operation code in the I-Register has the bit configuration corresponding to Instruction 17, Store Program Counter and Branch (SPB).

DSPG, Program Processor, FIG. 107: A signal generated by the Start-Stop Clock Logic to initiate the generation of clock pulses by the Program Processor Clock Generator.

DSPT, Program Processor, FIG. 108: A signal which issues at predetermined times and during the execution of predetermined instructions to cause working clock signal QSPX to issue, transferring the adder output to the P-Register.

DSRD, Program Processor, FIG. 108: A signal which issues during execution of Instruction 22, Shift (SHG), or during the execution of the shift operation of Instruction 26, Variable Length Divide (VLD).

DSRK, Program Processor, FIG. 108: A signal indicating the satisfaction of certain conditons during the execution of predetermined instructions. Signal DSRK permits the Memory Data Available signal DMDA from Memory to start generation of clock pulses in the Program Processor Clock Generator.

DSS4, Program Processor, FIG. 108: A signal which issues during execution of Instruction 22, Shift (SHG), when the sign is being inserted in the Accumulator upon completion of the shift. Signal DSS4 forces the zone bit in bit position 4 of the adder output to a binary 0.

DSSC, Program Processor, FIG. 108: A signal which issues during execution of Instruction 67, Central Processor Operations (CPO), to indicate that bit positions 0 and 1 of the instruction word contain binary 0's. During execution of Instruction 67, signal DSSC causes signals representing the states of predetermined indicator flip-flops to be stored in corresponding flip-flops of the M-Register. The status word in the M-Register is then stored in a specified memory location.

DSTG, Program Processor, FIG. 108: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the instructions of the Store Group, viz. Instructions 44–47.

DSTZ, Program Processor, FIG. 108: A signal which issues when each adder output signal in bit positions 21–23 is a binary 0.

DSW0, Program Processor, FIG. 108: A signal which issues at predetermined times and during the execution of predetermined instructions to set flip-flop WR0 of the W-Selector Register to the 1-state.

DSW1, Program Processor, FIG. 108: A signal which issues at predetermined times and during the execution of predetermined instructions to set flip-flop WR1 of the W-Selector Register to the 1-state.

DSW2, Program Processor, FIG. 108: A signal which issues at predetermined times and during the execution of predetermined instructions to set flip-flop WR2 of the W-Selector Register to the 1-state.

DSW3, Program Processor, FIG. 108: A signal which issues at predetermined times and during the execution of predetermined instructions to set flip-flop WR3 of the W-Selector Register to the 1-state.

DT00, Memory, FIG. 36: A signal which issues during time period T00.

DT01, Memory, FIG. 36: A signal which issues during time period T01 of a memory cycle.

DT07, Memory, FIG. 36: A signal which issues during time period T07 of a memory cycle.

DT08, Memory, FIG. 36: A signal which issues during time period T08 of a memory cycle.

DT13, Memory, FIG. 36: A signal which issues during time period T13 of a memory cycle.

DTAF, Program Processor, FIG. 108: A signal which issues during execution of an instruction employing accumulator words A, B and C when the A-Register and the Accumulator Counter Register contain the address of either accumulator word A or B.

DTAT, Program Processor, FIG. 108: A signal which isues during execution of an instruction employing accumulator words A, B and C when the A-Register and the Accumulator Counter Register contain the address of accumulator word C.

DTMV, Program Processor, FIG. 108: A signal which issues during execution of Instruction 06, Move (MOV), while the micro-operations of block W02 are being performed.

DTPI, Program Processor, FIG. 108: A signal which issues when the operation code in the I-Register has a bit configuration corresponding to one of the Triple Accumulator instructions, viz. Instructions 02, 12, 22, 32, 42, 52, 62, 06, 16, 26, 36, 46 and 56.

DTPO, Program Processor, FIG. 108: A signal which indicates that the instruction being executed requires that the working length of the Accumulator be triple. Signal DTPO issues during the execution of Instructions 42, 46 or 56 and during the execution of Instructions 50, 52, 53, 60, 62, 63, 02, 03, 05, 15, 20 and 21, when the working length of the Accumulator is set to triple.

DTSB, Program Processor, FIG. 108: A signal which issues during execution of Instruction 22, Shift (SHG), when the state of the signal in the least-significant bit position of the Accumulator is to be tested after the shift is completed.

DULS, Memory, FIG. 36: A signal which issues in response to the 15th bit of the address to select either the upper or the lower 16K memory unit of a 32K memory.

DUSE, Program Processor, FIG. 108: A signal which issues during the initial routine when both flip-flops CL1 and CL2 are reset to the 0-state, indicating that address development is not to be continued with either a P-Sequence AMS Word or an Indirect Address Word, or during execution of Instruction 06, Edit (EDT), to indicate that the format character requires the data character to be used in the result.

DVBJ, Program Processor, FIG. 108: A signal which issues when the last accumulator word is being processed during execution of one of the Instructions 02, Compare Decimal Accumulator to Memory (CDA), or 03, Compare Alphanumeric Accumulator to Memory (CAA).

DVCA, Program Processor, FIG. 109: A signal which issues when development of the second address of a two-address instruction has been completed.

DVCB, Program Processor, FIG. 109: A signal which issues during address development in sequence 2 of block W01 to indicate that address development is not to be continued with either a P-Sequence AMS Word or an Indirect AMS Word, i.e., when address development is completed.

DVCD, Program Processor, FIG. 109: A signal generated in the Program Processor when signal DVCB issues during execution of one of the instructions of Instruction Group Five, viz. Instructions 01, 07, 16, 17, 31, 32, 33 and 67.

DVCH, Program Processor, FIG. 109: A signal which issues during the execution of predetermined instructions to indicate that development of either the first address or second address, if the instruction is a two-address instruction, is complete.

DVDF, Program Processor, FIG. 109: A signal which issues while address development is occurring during sequence 2 of block W01 to indicate that address development is to be continued with a P-Sequence AMS Word.

DVEF, Program Processor, FIG. 109: See equation.

DVEQ, Program Processor, FIG. 109: See equation.

DVEV, Program Processor, FIG. 109: A signal which issues during execution of Instruction 16, Branch on Count (BRC), when development of the second address is completed.

DVEW, Program Processor, FIG. 109: A signal which issues during address development in block W01 when the Program Processor is performing sequence 0, sequence 1–DFXW, sequence 1–$\overline{\text{DFXW}}$ or sequence 3.

DVFB, Program Processor, FIG. 109: A signal which issues during execution of Instruction 05, Edit (EDT), to indicate that a suppression format character has been processed during the edit mode.

DVFK, Program Processor, FIG. 109: See equation.

DVFL, Program Processor, FIG. 109: See equation.

DVFW, Program Processor, FIG. 109: See equation.

DVGC, Program Processor, FIG. 109: See equation.

DVGD, Program Processor, FIG. 109: See equation.

DVGH, Program Processor, FIG. 109: A signal which issues when flip-flop M00 of the M-Register is set to the 1-state during execution of one of the Instructions 05, 20 or 21 or which issues when flip-flop E00 of the E-Register is set to the 1-state during execution of either Instruction 26 or Instruction 27.

DVGJ, Program Processor, FIG. 109: A signal which issues during execution of one of the Instructions 05, 20 or 21 when flip-flop M00 of the M-Register is reset to the 0-state or which issues during execution of Instruction 22, 26 or 27 when flip-flop E00 of the E-Register is reset to the 0-state.

DVHF, Program Processor, FIG. 109: A signal which issues during block W00 of the initial routine when the accumulator working length specified by the Accumulator Length Indicator Register is greater than the length specified by predetermined instructions.

DVHN, Program Processor, FIG. 109: See equation.

DVHW, Program Processor, FIG. 110: See equation.

DVHZ, Program Processor, FIG. 110: See equation.

DVJB, Program Processor, FIG. 110: See equation.

DVJC, Program Processor, FIG. 110: See equation.

DVJD, Program Processor, FIG. 110: See equation.

DVJH, Program Processor, FIG. 110: See equation.

DVJK, Program Processor, FIG. 110: A signal which issues during execution of one of the Instructions 50–57 and 60–63 to set flip-flop COR to the 1-state when the result of an arithmetic operation is minus 0.

DVJQ, Program Processor, FIG. 110: A signal which indicates that development of the first address of a two-address instruction is completed.

DVJR, Program Processor, FIG. 110: A signal which indicates that address development is to be continued with another P-Sequence AMS Word and that indirect addressing is not to be used in developing the address.

DVJS, Program Processor, FIG. 110: A signal which issues during sequences 0, 1–DFXW, 1–$\overline{\text{DFXW}}$ or 3 when an index or operand has not yet been obtained during address development.

DVJV, Program Processor, FIG. 110: See equation.

DVKB, Program Processor, FIG. 110: See equation.

DVLC, Program Processor, FIG. 110: See equation.

DVLR, Program Processor, FIG. 110: A signal which issues during the divide operation of Instruction 26, Variable Length Divide (VLD).

DVMA, Program Processor, FIG. 110: See equation.

DVMC, Program Processor, FIG. 110: See equation.

DVMF, Program Processor, FIG. 110: See equation.

DVMG, Program Processor, FIG. 111: See equation.

DVMJ, Program Processor, FIG. 111: See equation.

DVML, Program Processor, FIG. 111: See equation.

DVMM, Program Processor, FIG. 111: See equation.

DVMP, Program Processor, FIG. 111: See equation.

DVMR, Program Processor, FIG. 111: See equation.

DVMS, Program Processor, FIG. 111: See equation.

DVRA, Program Processor, FIG. 111: A signal which issues during execution of Instruction 05, Edit (EDT), to identify a format character of the use class.

DVRB, Program Processor, FIG. 111: A signal which is used during execution of Instruction 05, Edit (EDT), to identify a format character of either the ignore or sign classes.

DVRC, Program Processor, FIG. 111: A signal which issues during execution of Instruction 05, Edit (EDT), to indicate a "suppress and float $" format character.

DVRD, Program Processor, FIG. 111: A signal which issues during execution of Instruction 05, Edit (EDT), to identify a "suppress and * protect" format character.

DVRE, Program Processor, FIG. 111: A signal which issues during execution of Instruction 05, Edit (EDT), to identify a "simple suppress" format character.

DVRF, Program Processor, FIG. 111: A signal which issues during address development in block W01 to designate either sequence 1–DFXW, 1–$\overline{\text{DFXW}}$ or 2.

DVFM, Program Processor, FIG. 111: A signal which issues during time period TL3 of block W11 when the instruction being executed is Instruction 23, 24 or 25.

DVRN, Program Processor, FIG. 111: A signal which issues during time period TL1 of block W02 when the divide operation of Instruction 26, Variable Length Divide (VLD), is being performed.

DVRS, Program Processor, FIG. 111: A signal which indicates that one of the memory synchronizing signals $\overline{DMBU}$ or DMDA has issued. DVRS is one of the several signals required to start generation of clock pulses in the Program Processor Clock Generator.

DVRV, Program Processor, FIG. 111: See equation.

DVRW, Program Processor, FIG. 111: See equation.

DVSA, Program Processor, FIG. 111: A signal which issues during block W02 while one of the Instructions 22 or 26 is being executed and the count in the N-Register is greater than zero.

DVSB, Program Processor, FIG. 111: A signal which issues during execution of one of the Instructions 22 or 26 when the count in the N-Register is less than four, if an alpha or decimal shift, or less than 24, if a binary shift.

DVSC, Program Processor, FIG. 111: A signal which issues during block W03 when the divide operation of Instruction 26, Variable Length Divide (VLD), is being performed and the accumulator word A location is not being addressed.

DVSF, Program Processor, FIG. 111: A signal which issues during execution of Instruction 07, General (GEN), to indicate that flip-flop WRS has been set to the 1-state.

DVSG, Program Processor, FIG. 111: A signal which issues during block W15 when either Instruction 22 or Instruction 26 is being executed.

DVSH, Program Processor, FIG. 111: A signal which issues during block W05 when Instruction 32, Move to First Address Field (MTA), is being executed.

DVSK, Program Processor, FIG. 111: See equation.

DVSL, Program Processor, FIG. 111: A signal which issues when one of the Instructions 04, 23–25, 30, 34 or 35 is being executed.

DVSM, Program Processor, FIG. 112: A signal which issues during block W12 when Instruction 07, General (GEN), is being executed and flip-flop WRS is set to the 1-state.

DVSP, Program Processor, FIG. 112: A signal which issues during the divide operation of Instruction 26, Variable Length Divide (VLD), when the accumulator word C location is being addressed.

DVSS, Program Processor, FIG. 112: A signal which issues during execution of one of the Instructions 22, 26, 27, 50–57 and 60–63.

DVST, Program Processor, FIG. 112: A signal which issues during execution of Instruction 06, Move (MOV), when the address control field of the instruction word has a value other than seven.

DVSV, Program Processor, FIG. 112: A signal which issues during execution of Instruction 06, Move (MOV), when development of the address field of the instruction word is complete.

DVSW, Program Processor, FIG. 112: A signal which issues during execution of Instruction 20, Explode (EXP), when the accumulator word being processed is not the last accumulator word.

DVSY, Program Processor, FIG. 112: A signal which issues during execution of Instruction 06, MOVE (MOV), when Carry Remember flip-flop CRE is set to the 1-state.

DVTA, Program Processor, FIG. 112: A signal which issues in block W11 during execution of Instruction 27, Variable Length Multiply (VLM), when the accumulator word D location is being addressed.

DVTB, Program Processor, FIG. 112: A signal which issues during the divide operation of Instruction 26, Variable Length Divide (VLD), when accumulator word C is being addressed.

DVTC, Program Processor, FIG. 112: A signal which issues during the divide operation of Instruction 26, Variable Length Divide (VLD), when an accumulator word location other than the accumulator word D location is being addressed or when Carry Remember flip-flop CRE is reset to the 0-state.

DVTD, Program Processor, FIG. 112: A signal which issues during execution of Instruction 05, Edit (EDT), when the last accumulator word is being processed during the edit mode.

DVTE, Program Processor, FIG. 112: A signal which issues in block W14 during execution of either Instruction 20, Explode (EXP), or 21, Implode (IMP).

DVTF, Program Processor, FIG. 112: A signal which issues during execution of predetermined instructions (see equation).

DVTG, Program Processor, FIG. 112: A signal which issues during execution of one of the Instructions 02, 03, 22 and 26.

DVTH, Program Processor, FIG. 112: A signal which issues during execution of predetermined instructions (see equation).

DVTJ, Program Processor, FIG. 112: A signal which issues in block W15 during execution of predetermined instructions.

DVTL, Program Processor, FIG. 112: A signal which issues in block W02 during execution of Instruction 21, Implode (IMP).

DVTM, Program Processor, FIG. 112: A signal which issues during execution of one of the Instructions 50–57 and 60–63 if Decimal Correction flip-flop COR is reset to the 0-state.

DVTR, Program Processor, FIG. 112: See equation.

DVTS, Program Processor, FIG. 112: A signal which issues in block W01 during execution of Instruction 32, Move to First Address Field (MTA), when development of the second address is complete.

DVTV, Program Processor, FIG. 112: A signal which issues during execution of either Instruction 01, Compare Memory to Immediate (CMI), or 04, Compare Second to First Memory (CMM).

DVTX, Program Processor, FIG. 112: A signal which identifies sequence 1–DFXW of block W01.

DVTY, Program Processor, FIG. 112: A signal which issues in block W03 during the divide operation of Instruction 26, Variable Length Divide (VLD).

DVVB, Program Processor, FIG. 112: See equation.

DVVH, Program Processor, FIG. 112: See equation.

DVVL, Program Processor, FIG. 113: See equation.

DVVP, Program Processor, FIG. 113: See equation.

DVVQ, Program Processor, FIG. 113: A signal which indicates that the states of flip-flops M00 and E00 of the M-Register and the E-Register respectively are different.

DVVV, Program Processor, FIG. 113: A signal which indicates that the states of flip-flops M00 and E00 of the M-Register and E-Register respectively are identical.

DVVW, Program Processor, FIG. 113: A signal which issues in block W02 during execution of Instruction 22, Shift (SHG), or during the shift operation of Instruction 26, Variable Length Divide (VLD).

DVWA, Program Processor, FIG. 113: See equation.

DVZG, Program Processor, FIG. 113: A signal which issues in block W15 during execution of Instruction 22, Shift (SHG), or during the shift operation of Instruction 26, Variable Length Divide (VLD), during either a left shift when the count in the CC-Register is greater than zero or during a right shift.

DW00–DW12, DW14, DW15, Program Processor, FIG. 113: Signals which represent 15 of the 16 possible states of the W-Selector Register to define the corresponding blocks of micro-operations.

DWDB, Program Processor, FIG. 113: A signal which issues during the performance of the micro-operations of block W02 when the operation being performed is not a decimal operation during the execution of one of the Instructions 22 and 26.

DWDN, Program Processor, FIG. 114: A signal which issues during execution of Instruction 22, Shift (SHG),

when the shift is a decimal shift and when the Accumulator is being shifted through a number of character positions greater than 3.

DWEN, Program Processor, FIG. 114: The signal which issues when the Write Enable flip-flop WEN in Memory is set to the 1-state to cause the Memory-Write flip-flop MWR in the Program Processor to be reset to the 0-state.

DWL1, Program Processor, FIG. 114: A timing signal which issues when the TL-Counter is at TL1 during the W04 block of micro-operations.

DWL3, Program Processor, FIG. 114: A timing signal which issues when the TL-Counter is at TL5 during the W09 block of micro-operations.

DWL4, Program Processor, FIG. 114: A timing signal which issues when the TL-Counter is at TL1 during the W06 block of micro-operations.

DWL5, Program Processor, FIG. 114: A timing signal which issues when the TL-Counter is at TL5 during the W02 block of micro-operations.

DWL6, Program Processor, FIG. 114: A timing signal which issues when the TL-Counter is at TL2 during the W00 block of micro-operations.

DWNL, Program Processor, FIG. 114: A signal which issues during execution of Instruction 22, Shift (SHG), while the micro-operations of block W02 are being performed and when the Accumulator is shifted through a number of bit or character positions less than the length of a word.

DWRR, Program Processor, FIG. 114: A signal which issues in block W12 when flip-flop WRS is set to the 1-state.

DWSM, Program Processor, FIG. 114: A signal which issues during execution of Instruction 27, Variable Length Multiply (VLM), while the micro-operations of block W07 are being performed.

DWT1, Program Processor, FIG. 114: A timing signal which issues when the TL-Counter is at TL1 during the W00 block of micro-operations.

DWT3, Program Processor, FIG. 114: A timing signal which issues when the TL-Counter is at TL3 during the W04 block of micro-operations.

DWT4, Program Processor, FIG. 114: A timing signal which issues when the TL-Counter is at TL3 during the W15 block of micro-operations.

DWT5, Program Processor, FIG. 114: A timing signal which issues when the TL-Counter is at TL1 during the W01 block of micro-operations.

DWT6, Program Processor, FIG. 114: A timing signal which issues when the TL-Counter is at TL4 during the W08 block of micro-operations.

DWT7, Program Processor, FIG. 114: A timing signal which issues when the TL-Counter is at TL1 during the W09 block of micro-operations.

DWT8, Program Processor, FIG. 114: A timing signal which issues when the TL-Counter is at TL3 during the W11 block of micro-operations.

DWT9, Program Processor, FIG. 114: A timing signal which issues when the TL-Counter is at TL1 during the W07 block of micro-operations.

DX00–DX23, Program Processor, FIG. 115: Signals which provide the X-inputs to the first rank full adders S00–S23 of the Adder.

DXIM, Program Processor, FIG. 114: A signal which issues during address development in block W01 when an index is obtained.

DXNW, Program Processor, FIG. 114: A signal which issues during address development in block W01 to indicate that address development is to be continued with a P-Sequence AMS Word.

DXTA, Program Processor, FIG. 114: A signal which issues during address development in block W01 to indicate that development of the first address is completed and that development of the second address of a two-address instruction is to be commenced.

DY00–DY23, Program Processor, FIG. 116: Signals which provide the Y-inputs to the first rank full adders S00–S23 of the Adder.

DZ00, DZ06, DZ12, DZ18, Program Processor, FIG. 117: Signals which provide the Z-inputs to the first rank full adders S00, S06, S12 and S18 respectively of the Adder.

DZB2, Program Processor, FIG. 114: A signal which provides the Z-input to second rank full adder JB2 of the Parity Check and Generation Logic.

DZCP, Program Processor, FIG. 114: A signal which issues during execution of Instruction 05, Edit (EDT), when the character in flip-flops M05–M00 of the M-Register is a zero, a comma or a period.

FA14–FA02, Program Processor, FIGS. 44–47: The 1-output signals of the 13 flip-flops of the A-Register, which stores the address in Memory of the most-significant word of the Accumulator.

FAC2, FAC1, Program Processor, FIG. 70: The 1-output signals of Accumulator Counter Register flip-flops AC2 and AC1 which provide the two low-order bits to address, in conjunction with the contents of the A-Register, the appropriate word of the Accumulator.

FAEC, Program Processor, FIG. 78: The 1-output signals of a control flip-flop which, when set to the 1-state, enables the Y-inputs to full adders S00–S23 and the Z-input to full adder S00 of the Adder.

FAIM, Program Processor, FIG. 78: The 1-output signal of a control flip-flop which is set to the 1-state, in response to either an invalid address or invalid operation code, to terminate instruction processing and to initiate a Program Interrupt to the processor channel.

FAOI, Program Processor, FIG. 78: The 1-output signal of a control flip-flop which, when set to the 1-state, indicates that the next word to be read from Memory is an index or an operand.

FARC, Memory, FIG. 32: The 1-output signal of a control flip-flop which is set to the 1-state at the beginning of time period T02 of a memory cycle and which sets and triggers the generator providing the CSTR signal.

FARM, Program Processor, FIG. 78: The 1-output signal of a control flip-flop which, when set to the 1-state, causes the contents of the A-Register to be typed out by the typewriter.

FB00–FB14, Memory, FIGS. 27–30: The 1-output signals of the 15 flip-flops of the B-Register which store the address of the memory location into which a word is stored or from which a word is extracted during each memory cycle.

FCAY, Program Processor, FIG. 79: The 1-output signal of a control flip-flop which, when set to the 1-state, causes the Z-input to first rank full adder S00 to be energized, adding one to the adder output.

FCC4–FCC0, Program Processor, FIGS. 71–72: The 1-output signals of the 5 flip-flops of the CC-Register which serves as a counter during execution of Instructions 05, 22, 26 and 27.

FCCD, FCCC, FCCB, FCCA, Memory, FIG. 31: The 1-output signals of the flip-flops of the CC-Counter which define the time periods of the memory cycles.

FCL1, Program Processor, FIG. 72: The 1-output signal of a control flip-flop which, when set to the 1-state, indicates that indexing is to be continued with a P-Sequence AMS Word. The state of flip-flop CL1 also indicates the class of a format character during execution of Instruction 05, Edit (EDT).

FCL2, Program Processor, FIG. 72: The 1-output signal of a control flip-flop which, when set to the 1-state, indicates that address development is to be continued with an Indirect Address Word. The state of flip-flop CL2 also indicates the class of a format character during execution of Instruction 05, Edit (EDT).

FCMA Memory, FIG. 32: The 1-output signal of a control flip-flop which is set to the 1-state and reset to the 0-state by consecutive QCPA clock pulses to define the first phase of the two-phase Clock Generator.

FCMB, Memory, FIG. 32: The 1-output signal of a control flip-flop which is set to the 1-state and reset to the 0-state by consecutive QCPB clock pulses to define the second phase of the two-phase Clock Generator.

FCOR, Program Processor, FIG. 79: The 1-output signal of a control flip-flop which, when set to the 1-state, causes the result of an unlike-signs addition in the Adder to be corrected. Signal FCOR causes the result of the unlike-signs addition to be complemented to obtain the correct result. Flip-flop COR is also employed during execution of Instructions 26 and 05.

FCRE, Program Processor, FIG. 79: The 1-output signal of a control flip-flop which is set to the 1-state to store a carry out of the most-significant character of the Adder.

FCS2, Program Processor, FIG. 80: The 1-output signal of a control flip-flop which, when set to the 1-state, causes the contents of flip-flops CC0–CC3 of the CC-Register to be gated to the Y-input terminals of first rank full adders S00–S03 respectively.

FCS3, Program Processor, FIG. 80: The 1-output signal of a control flip-flop which, when set to the 1-state, causes the contents of flip-flops CC3–CC0 of the CC-Register to be gated to the Y-input terminals of first rank full adders S21–S18 respectively.

FCSF, Program Processor, FIG. 79: The 1-output signal of a control flip-flop which is set to the 1-state to cause predetermined flip-flops to be reset to the 0-state, when the Reset Computer switch of the Console is actuated.

FD14–FD00, Program Processor, FIGS. 48–51: The 1-output signals of the 15 flip-flops of the D-Register which stores data, index and instruction addresses and which stores shift information during execution of Instruction 22.

FDCE, Program Processor, FIG. 80: The 1-output signal of a control flip-flop which is set to the 1-state to enable the Adder to perform a decimal operation.

FDCW, Program Processor, FIG. 80: The 1-output signal of a control flip-flop which, when set to the 1-state, enables data transfer between the typewriter and the memory location containing the Data Control Word (DCW) for the typewriter.

FDSX, Program Processor, FIG. 81: The 1-output signal of a control flip-flop which, when set to the 1-state, causes the contents of the D-Register to be applied to first rank full adders S00–S14.

FE23–FE00, Program Processor, FIGS. 52–57: The 1-output signals of the 24 flip-flops of the E-Register which serves as an operating register during address development and instruction execution and which stores command and channel information during execution of Instruction 07.

FEMA, Program Processor, FIG. 81: The 1-output signal of a control flip-flop which, when set to the 1-state, enables the contents of the E-Register to be typed out in response to actuation of the appropriate console switch.

FESX, Program Processor, FIG. 81: The 1-output signal of a control flip-flop which, when set to the 1-state, causes the contents of the E-Register to be transferred to the first rank full adder circuits of the Adder.

FFVO, Program Processor, FIG. 81: The 1-output signal of a control flip-flop which is set to the 1-state if a carry is generated by the Adder during the last accumulator operation of a decimal instruction or if a carry occurs at the end of a binary addition or subtraction.

FGRE, Program Processor, FIG. 82: The 1-output signal of a control flip-flop which, when set to the 1-state, indicates the "greater than" condition during execution of one of the Instructions 01–04, or which, when reset to the 0-state along with flip-flop LES, indicates the "equal to" condition.

FIAR, Memory, FIG. 32: The 1-output signal of a control flip-flop which is set to the 1-state during a read/delay/write cycle to inhibit the resetting of the B-Register flip-flops to the 0-state at the end of the read operation and to inhibit the insertion of a new address into the B-Register at the start of the write operation.

FINH, Memory, FIG. 33: The 1-output signal of a control flip-flop which is set to the 1-state during time period T07 to control the write operation.

FIOR, Program Processor, FIG. 82: The 1-output signal of a control flip-flop which, when set to the 1-state, causes the first Data Control Word (DCW) in the list to be placed in the DCW memory location related to the selected channel.

FIR5–FIR0, Program Processor, FIGS. 69–70; The 1-output signals of the six flip-flops of the I-Register which stores the operation code of an instruction word.

FIRE, Program Processor, FIG. 82: The 1-output signal of a control flip-flop which is set to the 1-state during execution of Instructions 06, Move (MOV) and 16, Branch on Count (BRC), and which serves to increment the count field of the control words employed in executing the instructions.

FISP, Program Processor, FIG. 83: The 1-output signal of a control flip-flop of the Start-Stop Clock Logic which inhibits a new start pulse to the Program Processor Clock Generator until the previous circulating clock pulse has been stopped.

FLCP, Program Processor, FIG. 82: The 1-output signal of a control flip-flop which, when set to the 1-state, causes the TL-Counter to be preset to TL2 upon termination of a data interrupt.

FLES, Program Processor, FIG. 83: The 1-output signal of a control flip-flop which, when set to the 1-state, indicates the "less than" condition during execution of one of the compare Instructions 01–04. When reset to the 0-state, the 0-output signal of flip-flop LES in conjunction with the 0-output signal of flip-flop GRE, indicates the "equal to" condition.

FLSN, Program Processor, FIG. 83: The 1-output signal of a control flip-flop which is reset to the 0-state during an unlike-signs addition, i.e. during an add operation with the signs of the operand words different, or a subtract operation with the signs of the operand words the same.

FM23–FM00, Program Processor, FIGS. 58–63: The 1-output signals of the 24 flip-flops of the M-Register which serves as an operating register during instruction execution and input/output operations and as a path for data being stored in or extracted from Memory.

FMAI, Memory, FIG. 33: The 1-output signal of a control flip-flop which, when set to the 1-state, indicates that a memory address in excess of memory capacity has been transmitted to the B-Register. This flip-flop is not employed in a system having a 32K memory.

FMAN, Program Processor, FIG. 84: The 1-output signal of a control flip-flop which, when set to the 1-state, defines the manual mode of operation of the data processing system, during which instruction processing is halted.

FMBD, Memory, FIG. 33: The 1-output signal of a control flip-flop which is set to the 1-state during the time period T00 at the end of a memory cycle to control the termination of the memory cycle.

FMBU, Memory, FIG. 33: The 1-output signal of a control flip-flop which, when set to the 1-state, indicates that the Memory is busy. The flip-flop is reset to the 0-state a predetermined period before the end of a memory cycle to indicate to the Program Processor that the memory cycle is near completion.

FMBZ, Memory, FIG. 34: The 1-output signal of a control flip-flop which is set to the 1-state at the beginning of a memory cycle to indicate that the Memory is busy and which is reset to the 0-state at the end of a memory cycle to terminate generation of clock pulses by the memory Clock Generator.

FMDA, Memory, FIG. 34: The 1-output signal of a con-

trol flip-flop which is set to the 1-state to indicate that the word being read from Memory is available to the Data Drivers during a read operation.

FMEM, Program Processor, FIG. 84: The 1-output signal of a control flip-flop which, when set to the 1-state, enables the transfer of information between the typewriter and a selected memory location when flip-flop MAN is set to the 1-state.

FMNX, Program Processor, FIG. 84: The 1-output signal of a control flip-flop which is employed during the manual mode of operation.

FMOD, Program Processor, FIG. 84: The 1-output signal of a control flip-flop, which is set to the 1-state during execution of Instructions 26, Variable Length Divide (VLD), and 27, Variable Length Multiply (VLM), to initiate division by multiple subtractions or multiplication by multiple additions.

FMPI, Program Processor, FIG. 85: The 1-output signal of a control flip-flop which causes a program interrupt to the processor channel upon actuation of the console Request Control switch.

FMPL, Program Processor, FIG. 85: The 1-output signal of a control flip-flop which, when set to the 1-state, indicates that Instruction 27, Variable Length Divide (VLD), is being executed for the first time for a given dividend and divisor or that Instruction 27, Variable Length Multiply (VLM), is being executed for the first time for a given multiplicand and multiplier.

FMRD, Program Processor, FIG. 85: The 1-output signal of a control flip-flop which, when set to the 1-state, causes the Memory to initiate a read or clear operation.

FMRE, Program Processor, FIG. 85: The 1-output signal of a control flip-flop which is set to the 1-state in response to the actuation of the Reset Computer console switch and which presets the TL-Counter to TL2, resets certain flip-flops to the 0-state and initiates an idling cycle in the Central Processor.

FMRJ, Program Processor, FIG. 86: The 1-output signal of a control flip-flop which is set to the 1-state during a read operation in Memory to permit a parity check on the word extracted from Memory.

FMSX, Program Processor, FIG. 86: The 1-output signal of a control flip-flop which, when set to the 1-state, gates the contents of the M-Register flip-flops to the X-inputs of the corresponding first rank full adder circuits of the Adder.

FMUJ, Program Processor, FIG. 86: The 1-output signal of Memory Output Parity flip-flop MUJ which stores the parity bit of a word extracted from Memory.

FMWR, Program Processor, FIG. 86: The 1-output signal of a control flip-flop which, when set to the 1-state, initiates a write operation in Memory.

FMXS, Program Processor, FIG. 87: The 1-output signal of a control flip-flop which is employed during the manual mode of operation.

FN04–FN00, Program Processor, FIGS. 73–74: The 1-output signals of the five flip-flops of the N-Register which serves as a counter during execution of Instructions 05, 22, 26 and 27.

FNBY, Program Processor, FIG. 87: The 1-output signal of a control flip-flop which is set to the 1-state during the manual mode of operation when data transfer between the Input/Output Control Unit and the Program Processor are terminated.

FNXF, Program Processor, FIG. 87: The 1-output signal of a control flip-flop which, when set to the 1-state, indicates that the instruction is a two-address instruction and that the second address is being developed.

FP14–FP00, Program Processor, FIGS. 64–67: The 1-output signals of the 15 flip-flops of the P-Register which stores the address of the next instruction or word in the P-sequence.

FPEF, Program Processor, FIG. 87: The 1-output signal of a control flip-flop which is set to the 1-state in response to either an invalid operation code or an invalid memory address.

FPIA, Program Processor, FIG. 88: The 1-output signal of a control flip-flop which, when set to the 1-state, causes a program interrupt subroutine to commence upon granting of a program interrupt request.

FPIM, Program Processor, FIG. 88: The 1-output signal of a control flip-flop which is set to the 1-state in response to the actuation of the Request Control console switch and which causes a program interrupt to the processor channel.

FPIT, Program Processor, FIG. 88: The 1-output signal of a control flip-flop which is set to the 1-state when a program interrupt subroutine is being executed by the Central Processor.

FPL2, FPL1, Program Processor, FIG. 47: The 1-output signals of the 2 flip-flops of the Accumulator Length Indicator Register which stores the working length of the Accumulator and which, in conjunction with the contents of the A-Register, provides the address of an accumulator word.

FPMI, Program Processor, FIG. 88: The 1-output signal of a control flip-flop which, when set to the 0-state, permits a program interrupt to the processor channel upon actuation of the Reset Control console switch.

FPRI, Program Processor, FIG. 89: The 1-output signal of a control flip-flop which is employed to detect execution of the first Instruction 07, General (GEN), during a program interrupt subroutine.

FPRM, Program Processor, FIG. 89: The 1-output signal of a control flip-flop which, when set to the 1-state, enables data transfers to occur between the typewriter and the P-Register during the manual mode of operation.

FPSA, Program Processor, FIG. 89: The 1-output signal of a control flip-flop which is set to the 1-state during execution of a program interrupt subroutine.

FPSX, Program Processor, FIG. 89: The 1-output signal of a control flip-flop which, when set to the 1-state, causes the contents of the P-Register to be applied to the adder inputs.

FQ03–FQ00, Program Processor, FIG. 68: The 1-output signals of the four flip-flops of the Q-Register which serves as a counter during the execution of Instructions 05, 22, 26 and 27.

FRCK, Program Processor, FIG. 83: The 1-output signal of a control flip-flop of the Start-Stop Clock Logic which, when set to the 0-state, causes the generation of clock pulses by the Clock Generator to terminate and which, when set to either the 1-state or the 0-state, enables corresponding Gated Clock Distribution Drivers.

FRDX, Memory, FIG. 34: The 1-output signal of a control flip-flop which, when set to the 1-state, enables the X-Axis Address Decode and Driver Circuits to provide X-Axis read current through a selected row of cores in each plane of a stack.

FRDY, Memory, FIG. 34: The 1-output signal of the control flip-flop which, when set to the 1-state, enables the Y-Axis Address Decode and Driver Circuits to provide Y-Axis read current through a selected column of cores in each plane of a stack.

FRNZ, Program Processor, FIG. 90: The 1-output signal of a control flip-flop which, when set to the 1-state, indicates that at least one of the output signals from the Adder is a binary 1 during execution of Instructions 15, 50–57 and 60–63. Flip-flop RNZ also serves as a control flip-flop during execution of Instruction 05, Edit (EDT).

FRR2–FRR0, Input/Output Control Unit, FIG. 178: The 1-output signals of the three flip-flops of the R-Selector Register which define the blocks of micro-operations performed to control data transfers during an input/output operation.

FSIN, Program Processor, FIG. 90: The 1-output signal of a control flip-flop which is set to the 1-state when the

sign of the word in the M-Register is minus and which is reset to the 0-state when the sign of the word in the M-Register is plus.

FSP1, FSP2, Program Processor, FIG. 76: The 1-output signals of control flip-flops SP1 and SP2 which are employed during execution of Instruction 05, Edit (EDT), and during certain shift operations. During execution of Instruction 05, the states of flip-flops SP1 and SP2 indicate the type of suppression format character that was last processed. During shift operations, flip-flops SP1 and SP2 store partial parity information.

FSRE, Program Processor, FIG. 90: The 1-output signal of a control flip-flop which is set to the 1-state upon actuation of the Power On console switch to preset certain flip-flops in the Central Processor, preparing the Central Processor for the execution of instructions.

FSRI, Program Processor, FIG. 90: The 1-output signal of a control flip-flop which, when set to the 1-state, initiates a right shift in the M- and E-Registers during execution of predetermined instructions.

FSSS, Program Processor, FIG. 91: The 1-output signal of a control flip-flop which is set to the 1-state upon actuation of the Run or Single Instruction console switches to permit execution of instructions by the Central Processor.

FSTE, Program Processor, FIG. 91: The 1-output signal of a control flip-flop which is set to the 1-state in response to the actuation of the Single Instruction console switch.

FSTP, Program Processor, FIG. 91: The 1-output signal of a control flip-flop which is set to the 1-state upon actuation of the Manual console switch.

FSTT, Program Processor, FIG. 91: The 1-output signal of a control flip-flop which is set to the 1-state in response to the actuation of the Run console switch.

FTB1, FTB0, Program Processor, FIG. 74: The 1-output signals of the two flip-flops of the TB-Counter. The TB-Counter controls the timing of the repeated subtractions and additions during execution of Instructions 26, Variable Length Divide (VLD), 27, Variable Length Multiply (VLM), respectively and the timing of Instruction 05, Edit (EDT).

FTL5–FTL0, Program Processor, FIGS. 75–76: The 1-output signals of the six flip-flops of the TL-Counter which defines six time periods during the performance of a block of micro-operations.

FTVO, Program Processor, FIG. 92: The 1-output signal of a control flip-flop which, when set to the 1-state, enables a program interrupt to occur when flip-flop FVO is set to the 1-state.

FWEN, Memory, FIG. 35: The 1-output signal of a control flip-flop which is set to the 1-state to control a write operation.

FWJE, Program Processor, FIG. 92: The 1-output signal of Word Parity Error flip-flop WJE which is set to the 1-state in response to the detection of a parity error in a word extracted from Memory.

FWR3–FWR0, Program Processor, FIG. 77: The 1-output signals of the four flip-flops of the W-Selector Register which defines the block of micro-operations to be performed by the Central Processor.

FWRA, Memory, FIG. 35: The 1-output signal of a control flip-flop which, when set to the 1-state, enables the X-Axis Address Decode and Driver Circuits to provide X-Axis write currents through a selected row of cores in each plane of a stack.

FWRB, Memory, FIG. 35: The 1-output signal of a control flip-flop which, when set to the 1-state, enables the Y-Axis Address Decode and Driver Circuits to provide Y-Axis write currents through a selected column of cores in each plane of a stack.

FWRS, Program Processor, FIG. 67: The 1-output state of a control flip-flop which serves to synchronize operations during an input/output operation if the selected channel is not busy.

FXS2, FXS1, Program Processor, FIG. 92: The 1-output signals of two control flip-flops which indicate the one of the four available address development sequences of the W01 block that is being performed.

JPOX, Input/Output Control Unit: A signal transmitted from a peripheral subsystem to its corresponding channel to indicate that the peripheral subsystem is operable. Signal JPOX enables the circuits of the channel.

MEVP, Input/Output Control Unit: A parity signal employed in checking and generating parity in a channel. In generating parity for a character to be transmitted to a peripheral subsystem, MEVP is a binary 1 if the character contains an even number of binary 1's, providing an odd number of binary 1's for transmission to the peripheral subsystem. In checking parity, signal MEVP is a binary 0 if the character and parity signals received from the peripheral subsystem contain an odd number of binary 1's, indicating correct parity.

QCCD, Program Processor, FIG. 117: The feedback clock pulse in the Central Processor Clock Generator which maintains generation of clock pulses after initiation by the Central Processor Start-Stop Clock Logic.

QCCK, Program Processor, FIG. 117: The clock pulse output of the Central Processor Clock Generator which is applied to the Central Processor Gated Clock Distribution Drivers.

QCER, Program Processor, FIG. 117: A working clock signal provided by a Gated Clock Amplifier which causes the complement of the contents of the E-Register to be formed in the E-Register.

QCGA, Memory, FIG. 37: A signal which controls the generation of the first phase QCPA clock pulses by the two-phase memory timing clock generator.

QCGB, Memory, FIG. 37: A signal which controls the generation of second phase QCPB clock pulses by the two-phase memory timing clock generator.

QCKC, Program Processor, FIG. 117: The output signal of a Gated Clock Distribution Driver which is gated with the 0-output signal of flip-flop RCK.

QCLK, Program Processor, FIG. 117: The output signals of five Gated Clock Distribution Drivers of the Timing Clock Generator which are gated with the 1-output signal of flip-flop RCK.

QCNX, Program Processor, FIG. 117: A working clock signal which is delivered by a Gated Clock Amplifier to effect the parallel transfer of the contents of the CC-Register to the N-Register.

QCPA, Memory, FIG. 40: The first-phase clock pulses which are delivered by a Gated Clock Amplifier and which are employed to control the timing of operations in a memory cycle.

QCPB, Memory, FIG. 40: The second-phase clock pulses which are delivered by a Gated Clock Amplifier and which are employed to control the timing of operations in a memory cycle.

QCPC, Memory, FIG. 40: The clock pulses which are delivered by a Gated Clock Amplifier and which are employed, in conjunction with the phase one and phase two clock pulses, to control operations in a memory cycle.

QCPD, Memory, FIG. 40: The feedback clock pulse in the memory Clock Generator which maintains generation of clock pulses after initiation by the Memory Start-Stop Clock Logic.

QCPG, Memory, FIG. 40: The clock pulses which are delivered by the memory Clock Generator and which are applied to Gated Clock Amplifiers.

QEAX, Program Processor, FIG. 117: A working clock signal which is delivered by a Gated Clock Amplifier to effect the parallel transfer of the contents of the E14–E02 flip-flops of the E-Register to the corresponding flip-flops of the A-Register.

QEEX, Program Processor, FIG. 117: A working clock signal which is delivered by a Gated Clock Amplifier to effect the parallel transfer of the contents of the

E08–E00 flip-flops of the E-Register to the E23–E15 flip-flops respectively of the E-Register.

QMIX, Program Processor, FIG. 117: A working clock signal which is delivered by a Gated Clock Amplifier to effect the parallel transfer of the operation code field of an instruction word from the M23–M18 flip-flops of the M-Register to the IR5–IR0 flip-flops of the I-Register.

QNCD, Program Processor, FIG. 117: A working clock signal which is delivered by a Gated Clock Amplifier to cause the N-Register to change count at predetermined times during the execution of certain instructions.

QNCX, Program Processor, FIG. 117: A working clock signal which is delivered by a Gated Clock Amplifier to effect the parallel transfer of the contents of the N-Register to the CC-Register.

QQNX, Program Processor, FIG. 117: A working clock signal which is delivered by a Gated Clock Amplifier to effect the parallel transfer of the contents of the Q03–Q00 flip-flops of the Q-Register to the N03–N00 flip-flops respectively of the N-Register.

QSCX, Program Processor, FIG. 117: A working clock signal which is delivered by a Gated Clock Amplifier to gate the five least-significant bits in the output of the Adder to the corresponding flip-flops of the CC-Register.

QSDX, Program Processor, FIG. 117: A working clock signal which is delivered by a Gated Clock Amplifier to gate the 15 least-significant output bits of the Adder to the corresponding flip-flops of the B-Register.

QSEX, Program Processor, FIG. 117: A working clock signal which is delivered by a Gated Clock Amplifier to gate the output bits of the Adder to the corresponding flip-flops of the E-Register.

QSHE, Program Processor, FIG. 118: A working clock signal which is delivered by a Gated Clock Amplifier to cause open or circular shifting of the contents of the E-Register during the execution of Instruction 22, Shift (SHG).

QSHM, Program Processor, FIG. 118: A working clock signal which is delivered by a Gated Clock Amplifier to cause the contents of the M-Register to be shifted right.

QSMA, Program Processor, FIG. 118: A working clock signal which is delivered by a Gated Clock Amplifier to gate the output bits 0–5 of the Adder to the corresponding flip-flops M00–M05 of the M-Register.

QSMB, Program Processor, FIG. 118: A working clock signal which is delivered by a Gated Clock Amplifier to gate the output bits 6–14 from the Adder to the corresponding flip-flops M06–M14 of the M-Register.

QSMC, Program Processor, FIG. 118: A working clock signal which is delivered by a Gated Clock Amplifier to gate the output bits 15–23 from the Adder to flip-flops M15–M23 of the M-Register.

QSPX, Program Processor, FIG. 118: A working clock signal which is delivered by a Gated Clock Amplifier to gate the 15 least-significant output bits of the Adder to the corresponding flip-flops of the P-Register.

QTC0, Program Processor, FIG. 118: A control clock signal which is delivered by a Gated Clock Amplifier to change the states of the W-Selector Register and the R-Selector Register; signal QTC0 also changes the state of the TL-Counter from TL0 to TL2 during a split sequence.

QTC2, Program Processor, FIG. 118: A control clock signal which is delivered by a Gated Clock Amplifier to change the states of the W-Selector Register and the R-Selector Register; signal QTC2 also changes the state of the TL-Counter from TL2 to TL0 during a split sequence.

QTCK, Program Processor, FIG. 118: A working clock signal which is delivered by a Gated Clock Amplifier and which, along with the TL-Counter, controls the timing of the micro-operations of a block.

QTLP, Program Processor, FIG. 118: A working clock signal which is delivered by a Gated Clock Amplifier to cause a change in the state of the TL-Counter.

QTRK, Program Processor, FIG. 118: A control clock signal which is delivered by a Gated Clock Amplifier to cause Run Clock flip-flop RCK to be set to the 1-state, preparatory to initiating generation of clock pulses in the Clock Generator.

QTSP, Program Processor, FIG. 118: A control clock signal which is delivered by a Gated Clock Amplifier to cause Inhibit Start Pulse flip-flop ISP to be set to the 0-state, preparatory to initiating clock pulse generation in the Clock Generator.

REAT, Input/Output Control Unit: A signal transmitted from a peripheral subsystem to its corresponding channel to indicate that the peripheral subsystem has sent a character to the channel during a read operation.

REBX, Input/Output Control Unit: A signal which is transmitted by a peripheral subsystem to its corresponding channel when the peripheral subsystem is busy and is not accessible for execution of Instruction 07 (GEN).

REPX, Input/Output Control Unit: A parity signal transmitted from the peripheral subsystem to its corresponding channel along with data signals REX0–REX5.

RERX, Input/Output Control Unit: A signal transmitted by a peripheral subsystem which precedes termination of the busy status of the peripheral subsystem.

REWX, Input/Output Control Unit: A signal which is transmitted from a peripheral subsystem to its corresponding channel, during execution of Instruction 07 (GEN), requesting the channel to transmit device, function and release information to the peripheral subsystem. The peripheral subsystem cannot start transmission of data interrupt requests to channel until after the device, function and release requests are satisfied. Signal REWX also serves to request a character from the channel during a write operation.

RMX0–RMX2, Input/Output Control Unit: Signals transmitted by a peripheral subsystem to its corresponding channel to indicate the major status of the peripheral subsystem.

RPTS, Input/Output Control Unit: A special interrupt signal which issues in channel #0.

W100–W124, Memory: The signals which represent the 25 bits of a word read from stack No. 1 of a 16K memory unit during a read operation and which are applied to the associated set of Sense Amplifiers.

W200–W224, Memory: The signals which represent the 25 bits of a word read from stack No. 2 of a 16K memory unit during a read operation and which are applied to the associated set of Sense Amplifiers.

W300–W324, Memory: The signals which represent the 25 bits of a word read from stack No. 3 of a 16K memory unit during a read operation and which are applied to the associated set of Sense Amplifiers.

W400–W424, Memory: The signals which represent the 25 bits of a word read from stack No. 4 of a 16K memory unit during a read operation and which are applied to the associated set of Sense Amplifiers.

WD00–WD24, Memory, FIG. 43: Signals representing the 25 bits of a word read from Memory and delivered by the Data Drivers for transfer to the M-Register and the MUJ flip-flop of the Program Processor.

$\overline{XX01}$–$\overline{XX08}$, Memory, FIG. 37: The select X line-read output signals of the X-Axis Address Decode and Driver Circuits which effect selection of an X line during a read operation.

XX01–XX08, Memory, FIG. 37: The select X line-write output signals of the X-Axis Address Decode and Driver Circuits which effect selection of an X line during a write operation.

$\overline{XXB1}$–$\overline{XXB8}$, Memory, FIG. 38: The select X group output signals of the X-Axis Address Decode and Driver Circuits which effect selection of an X group in stacks No. 1 and No. 3 of a 16K memory unit during a memory cycle.

XXB1–XXB8, Memory, FIG. 38: The select X group output signals of the X-Axis Address Decode and Driver Circuits which effect selection of an X group in stacks No. 2 and No. 4 of a 16K memory unit during a memory cycle.

$\overline{XY01}$–$\overline{XY08}$, Memory, FIG. 39: The select Y line-write output signals of the Y-Axis Address Decode and Driver Circuits which effects selection of a Y-line during a write operation.

XY01–XY08, Memory, FIG. 39: The select Y line-read output signals of the Y-Axis Address Decode and Driver Circuits which effect selection of a Y-line during a read operation.

$\overline{XYB1}$–$\overline{XYB8}$, Memory, FIG. 40: The select Y group output signals of the Y-Axis Address Decode and Driver Circuits which effect selection of a Y group in stacks No. 1 and No. 2 of a 16K memory unit during a memory cycle.

XYB1–XYB8, Memory, FIG. 40: The select Y group output signals of the Y-Axis Address Decode and Driver Circuits which effect selection of a Y group in stacks No. 3 and No. 4 of a 16K memory unit during a memory cycle.

Z100–Z124, Memory, FIG. 41: The output signals of the set of Inhibit Drivers associated with stack No. 1 of a 16K memory unit.

Z200–Z224, Memory, FIG. 41: The output signals of the set of Inhibit Drivers associated with stack No. 2 of a 16K memory unit.

Z300–Z324, Memory, FIG. 42: The output signals of the set of Inhibit Drivers associated with stack No. 3 of a 16K memory unit.

Z400–Z424, Memory, FIG. 42: The output signals of the set of Inhibit Drivers associated with stack No. 4 of a 16K memory unit.

DCC6, DCC5, Program Processor: Signals representing the number of characters, less than four, in a single- or double-word channel at termination of a data interrupt. Signals DCC5 and DCC6 are applied to the X-input terminals of adder circuits S15 and S16 respectively to force this count into bit positions 15 and 16 of the channel Data Control Word.

DDI1–DDI0, Input/Output Control Unit, FIG. 172: Data interrupt request signals transmitted by channels #1–#0 respectively to a plugboard of the data interrupt logic.

DDPA–DDPH, Input/Output Control Unit, FIG. 172: The output signals of the data interrupt priority gates of the data interrupt logic, only one of which issues at a time. The signal which issues indicates the highest-priority data interrupt flip-flop set to the 1-state, in response to data interrupt requests from the corresponding channels.

DEL1–DEL0, Input/Output Control Unit, FIG. 172: Signals representing the channel code in bit positions 11–14 of the E-Register during execution of Instruction 07, General (GEN), which are applied to gates in the data interrupt logic to effect selection of a channel.

DG00–DG23, Input/Output Control Unit, FIG. 171: Signals transmitted from the M-Register to the input/output matrix of the Input/Output Control Unit to effect transfer of data from Memory to a selected peripheral subsystem during a write operation.

DI00–DI05, DI10–DI15, DI20–DI25, DI30–DI35, Input/Output Control Unit, FIG. 171: The output signals of the input/output matrix of the Input/Output Control Unit which represent data being transmitted to either the channel or to the M-Register during a data transfer between Memory and a peripheral subsystem.

DJ00–DJ05, DJ10–DJ15, DJ20–DJ25, DJ30–DJ35, Input/Output Control Unit, FIG. 171: The input signals to the input/output matrix of the Input/Output Control Unit which may represent data being transmitted to the input/output matrix from either the channel or the M-Register, or which may represent device and function information being transmitted from the E-Register to the input/output matrix during execution of Instruction 07, General (GEN).

DK00–DK05, DK10–DK15, DK20–DK25, DK30–DK35, Input/Output Control Unit, FIG. 171: Signals representing data being transmitted from a channel of the Input/Output Control Unit for application to the input/output matrix during a read operation.

DPC1–DPC0, Input/Output Control Unit, FIG. 172: The output signals of gates of the data interrupt logic, only one of which issues at a given time. The signal which issues is applied to a plugboard to select a channel.

DPI1–DPI0, Input/Output Control Unit, FIG. 174: Signals representing program interrupt requests transmitted from the respective channels #1–#0 to a plugboard of the program interrupt logic.

DPSC, Input/Output Control Unit, FIG. 174: A signal which issues to set flip-flop PIA, initiating a program interrupt, upon coincidence of a program interrupt request signal from a channel and the count in the program interrupt scanner of the program interrupt logic.

FDSA–FDSH, Input/Output Control Unit, FIG. 172: The 1-output signals of the data interrupt priority flip-flops which are set to the 1-state upon application of data interrupt request signals from the corresponding channels #1–#0.

JEAX, Input/Output Control Unit, FIG. 172: A read clock signal transmitted from a peripheral subsystem to its corresponding channel when the peripheral subsystem is ready to transmit a character to the channel during a read operation.

JEWX, Input/Output Control Unit, FIG. 172: A write clock signal which is transmitted from a peripheral subsystem to its corresponding channel to indicate that the peripheral subsystem is ready to receive another character from the channel during a write operation, or to request device, function and release signals during execution of Instruction 07 (GEN).

RCX0–RCX5, Input/Output Control Unit, FIG. 171: Signals representing a data character transmitted from a channel of the Input/Output Control Unit to the associated peripheral subsystem during a write operation.

REX0–REX5, Input/Output Control Unit, FIG. 171: Signals representing a character transmitted from a peripheral subsystem to its corresponding channel of the Input/Output Control Unit during a read operation.

ZCN1–ZCN0, Input/Output Control Unit, FIG. 172: The channel select output signals of a plugboard of the data interrupt logic, only one of which issues at a time. The signal which issues is effective to select a channel.

ZDPA–ZDPH, Input/Output Control Unit. FIG. 172: The output signals of a plugboard of the data interrupt logic, only one of which issues to address the appropriate DCW and LPW memory locations for the channel being granted a data interrupt. The signal which issues also causes generation of an appropriate channel select signal in gates of the data interrupt logic.

ZDSA–ZDSH, Input/Output Control Unit, FIG. 172: The output signals of a plugboard of the data interrupt logic which represent data interrupt requests transmitted from the channels through the plugboard to the data interrupt priority flip-flops.

ZPI1–ZPI0, Input/Output Control Unit, FIG. 174: The output signals of a plugboard of the program interrupt logic which represent program interrupt requests transmitted from the channels through the plugboard to gates which detect coincidence of program interrupt

requests and the count in the program interrupt scanner.

COMPONENT DETAILS AND OPERATION

The internal circuit structure of the components of the system, the circuit configurations interconnecting the components and a description of the operation of the system follow. The details of the circuit structure of the components and of the circuit configurations interconnecting the components are provided in FIGS. 14, 16–18, 20, 21, 23, 25–118, 170–172, 174 and 178–196. These component structures and interconnecting configurations comprise novel assemblages of the previously described clock generator, gated clock distribution driver, gated clock amplifier, flip-flops, NAND-gate, NAND-supplement, logic driver, full adder, AND-gate and OR-gate to perform the function of the data processing system. The connective relationship between these circuit elements can usually be described by logic expressions of the type illustrated in the foregoing sections "NAND-Gate," "NAND-Supplement" and "Logic Driver."

The logical schematic diagram

Generally, two or more output signals from flip-flops and gated clock amplifiers and the signals provided by certain console switches and push buttons are combined logically by NAND-gates, NAND-supplements, logic drivers, full adder circuits, AND-gates and OR-gates to provide input signals to other flip-flops, and to provide gating signals for gated clock amplifiers. Thus, the two trigger signals to a flip-flop (FIGS. 6 and 7) are usually the output signals of respective logical chains of NAND-gates, NAND-supplements, logic drivers, AND-gates and OR-gates, which receive output signals provided by other flip-flops and by gated clock amplifiers and certain console switches and push buttons. These logical chains may be described and illustrated by logical expressions which are actually logical schematic diagrams representing the logical and structural interconnection of the chain providing these two trigger signals to the flip-flop; e.g., see FIG. 27. Thus, the circuits providing the trigger input signals for a flip-flop may be illustrated by two logical schematic diagrams for each flip-flop, as shown in FIG. 27. The flip-flop representations of FIG. 27 and the associated logical schematic diagrams to the left of each flip-flop symbol are termed "Flip-Flop Input" diagrams. The circuits providing the input signals to a full adder circuit may also be illustrated by logical schematic diagrams. The logical schematic diagrams of the logical circuits providing input signals to the full adder circuits are illustrated in the drawings under the heading "Full Adder Input Signals"; e.g., see FIG. 115.

In a similar manner, the logical circuits providing input signals to the clock generator, to the gated clock distribution drivers and to the gated clock amplifiers may be illustrated by logical schematic diagrams. The logical schematic diagrams of the logical circuits providing input signals to the clock generators are illustrated in the drawings under the heading "Clock Generator Input Signals" (see FIG. 117). The logical schematic diagrams of the logical circuits providing input signals to the gated clock distribution drivers are illustrated in the drawings under the heading "Gated Clock Distribution Driver Input Signals" (see FIG. 117). Similarly, the logical schematic diagrams of the logical circuits providing input signals to the gated clock amplifiers are illustrated in the drawings under the heading "Gated Clock Amplifier Input Signals" (see FIGS. 117–118).

The five types of logical schematic diagrams, namely, "Flip-Flop Input" diagrams, "Full Adder Input Signal" diagrams, "Clock Generator Input Signal" diagrams, "Gated Clock Distribution Driver Input Signal" diagrams and "Gated Clock Amplifier Input Signal" diagrams are employed in FIGS. 27–35, 40, 44–92, 115–118 and 178–186 to illustrate the internal circuit structure of the components and the interconnecting circuit configurations between the components of the data processing system.

In addition to the five types of logical schematic diagrams described above, a sixth type of logical schematic diagram, termed the "Logical Combination Signal" diagram, is employed in FIGS. 36–43, 93–114 and 187–196. Logical combination signals are generated within logical chains. Such signals are, therefore, the output signals of a first logical chain and are employed as input signals to one or more additional logical chains. For example, the DVRB signal (FIG. 111) is employed as an input signal to a logical chain whose output signal is applied to the set input terminal of flip-flop CL2 to trigger flip-flop CL2 to its 1-state. The DVRB signal is the output signal of the logical chain which receives input signals from the M00, M01, M02, M03, M04, and M05 flip-flops of the M-Register. The inverted form of the DVRB signal, viz. $\overline{\text{DVRB}}$ is applied as an input signal to a logical chain whose output signal is applied to the reset input terminal of the CL2 flip-flop to set it to its 0-state. Thus, the $\overline{\text{DVRB}}$ logical combination signal is applied as an input signal to a plurality of logical chains whose output signals are applied to the input terminals of the CL2 flip-flop.

In the logical circuits illustrated by the following types of logical schematic diagrams, the logical functions of conjunction are implemented by AND-gates and logic drivers, the disjunctive functions are implemented by OR-gates, and the NAND functions are implemented by NAND-gates and NAND-supplements:

(a) Flip-flop Input diagrams,
(b) Full Adder Input Signal diagrams,
(c) Clock Generator Input Signal diagrams,
(d) Gated Clock Distribution Driver Input Signal diagrams,
(e) Gated Clock Amplifier Input Signal diagrams, and
(f) Logical Combination Signal diagrams.

The employment of a 1-input NAND-gate in series with a NAND-gate is the logical equivalent of an AND-gate. A NAND-gate with each input term inverted is the logical equivalent of an OR-gate with the same input terms in their non-inverted form. Conversely, an OR-gate with each input term in its inverted form is the logical equivalent of a NAND-gate with the same input terms in their non-inverted form.

It is convenient to note in the description of the data processing system that the first alphanumeric character of a signal designation indicates the type of circuit which generates the signal. Alphanumeric character D in the first character position of a signal designation indicates that the signal is generated by a NAND-Gate, a NAND-Supplement, a Logic Driver, an AND-Gate or an OR-Gate, e.g. logical combination signal DW14 (FIG. 113). The alphanumeric character F indicates that the signal is generated by a flip-flop circuit, e.g. signal $\overline{\text{FM28}}$ (FIG. 58). Full adder output signals are identified by the letter M, e.g. signal MS06 (FIG. 25). The alphanumeric character Q identifies a clock related signal, e.g. control clock signal QTRK (FIG. 118). Signals originating in jumper boards employ the alphanumeric character Z. Character R identifies signals originating in the transmitter and receiver circuits of the Input/Output Control Unit.

The following example demonstrates the significance of the logical schematic diagram and its employment to illustrate the interconnection of flip-flops, the clock generator, gated clock distribution drivers, gated clock amplifiers, NAND-gates, NAND-supplements, logic drivers, AND-gates, OR-gates and full-adder circuits. It particularly illustrates that this notation, couched in terms of signal designations in associative and combinatorial arrangements, completely defines the block and schematic circuit element arrangements, once these basic building blocks have been determined, as has been done in FIGS. 3–11, 23 and 25.

FIGURE 13 is a block diagram of the circuits providing the two trigger signals to the CL2 flip-flop. This figure is the block diagram equivalent of the logical schematic diagram of FIG. 72 which defines the FCL2 and $\overline{FCL2}$ trigger signals, the logical schematic diagrams of FIG. 111 which define the DVRB and DVRF logical combination signals, the logical schematic diagrams of FIG. 113 which define the DW00, DW01 and DW14 logical combination signals, the logical schematic diagram of FIG. 109 which defines the DVEW logical combination signal, the logical schematic diagram of FIG. 114 which defines the DWT1 logical combination signal, and the logical schematic diagram of FIG. 194 which defines the DR00 logical combination signal.

The DR00 logical combination signal is the output signal of Logic Driver **330**. Logic Driver **330** receives the $\overline{FRR0}$ output signal of the RR0 flip-flop, the $\overline{FRR1}$ output signal of the RR1 flip-flop and the $\overline{FRR2}$ output signal of the RR2 flip-flop. The output signal of Logic Driver **330** is represented by:

$$DR00=\overline{FRR0}\ \overline{FRR1}\ \overline{FRR2}$$

where $\overline{FRR0}$ $\overline{FRR1}$ $\overline{FRR2}$ represents the logical operation of conjunction of the three input signals applied to Logic Driver **330** and is also a direct representation of the circuit generating DR00. Therefore, the logical schematic diagram $\overline{FRR0}$ $\overline{FRR1}$ $\overline{FRR2}$ may also be employed to illustrate the logical circuit which generates the DR00 signal.

The DW00 logical combination signal is generated by the logical circuit comprising Logic Driver **331**. Logic Driver **331** receives the DR00 output signal of Logic Driver **330**, the $\overline{FWR3}$ output signal of flip-flop WR3, the $\overline{FWR2}$ output signal of flip-flop WR2, the $\overline{FWR1}$ output signal of flip-flop WR1 and the $\overline{FWR0}$ output signal of flip-flop WR0. The output signal of Logic Driver **331** is represented by:

$$DW00=DR00\ \overline{FWR3}\ \overline{FWR2}\ \overline{FWR1}\ \overline{FWR0}$$

Therefore, the above logical schematic diagram also illustrates the logical circuit which generates the DW00 signal.

The logical combination signal DWT1 is the output signal of AND-gate **332**. AND-gate **332** receives the DW00 output signal of Logic Driver **331** and the FTL1 output signal of flip-flop TL1. The output signal of AND-gate **332** is represented by:

$$DWT1=DW00\ FTL1$$

Therefore, the above logical schematic diagram also illustrates the logic circuit which generates the DWT1 signal.

The logical combination signal $\overline{DVRB}$ is generated by the logical circuit comprising NAND-gate **335** and NAND-supplement **336**. NAND-gate **335** receives the $\overline{FM05}$ output signal of flip-flop M05, the $\overline{FM04}$ output signal of flip-flop M04, the FM03 output signal of flip-flop M03, the FM02 output signal of flip-flop M02, the FM01 output signal of flip-flop M01 and the FM00 output signal of flip-flop M00. NAND-supplement **336** receives the FM05 output signal of flip-flop M05, the $\overline{FM04}$ output signal of flip-flop M04, the FM03 output signal of flip-flop M03, the $\overline{FM02}$ output signal of flip-flop M02, the FM01 output signal of flip-flop M01 and the $\overline{FM00}$ output signal of flip-flop M00. The output signal of NAND-gate **335** and NAND-supplement **336** is represented by:

$$\overline{DVRB}=\overline{FM05}\ \overline{FM04}\ FM03\ FM02\ FM01\ FM00+FM05\ \overline{FM04}\ FM03\ \overline{FM02}\ FM01\ \overline{FM00}$$

The above logical schematic diagram therefore also illustrates the logical circuit which generates the $\overline{DVRB}$ signal.

The DW14 logical combination signal is generated in a similar manner by Logic Driver **337**. The DW01 logical combination signal is generated by Logic Driver **338** and the $\overline{DVRF}$ logical combination signal is generated by NAND-gate **340** and NAND-supplement **341**.

The trigger signal applied to the reset input terminal of flip-flop CL2, i.e., the trigger signal which triggers flip-flop CL2 to its 0-state, is generated by the logical circuit comprising OR-gate **342**, AND gate **343** and AND gate **344**. AND gate **343** receives the logical combination signals DW01 and $\overline{DVRF}$, and the FTL3 and $\overline{FM21}$ output signals of flip-flops TL3 and M21 respectively. The output signal of AND-gate **343** is represented by:

$$\overline{DVRF}\ DW01\ \overline{FM21}\ FTL3$$

AND-gate **344** receives the logical combination signals DW14 and $\overline{DVRB}$ and the FTL2 output signal of flip-flop TL2. The output signal of AND-gate **344** is represented by:

$$DW14\ FTL2\ \overline{DVRB}$$

OR-gate **342** receives the logical combination signal DWT1 and the output signals of AND-gate **343** and AND-gate **344**. The output signal of OR-gate **342**, which is the trigger signal to the reset input terminal of the CL2 flip-flop, is therefore represented by:

$$\overline{FCL2}=\overline{DVRF}\ DW01\ \overline{FM21}\ FTL3 + DWT1 + DW14\ FTL2\ \overline{DVRB}$$

Therefore, the above logical schematic diagram also illustrates the logical circuit which generates the trigger signal which sets the CL2 flip-flop to its 0-state.

In a similar manner, the logical schematic diagram

$$FCL2=\overline{DVRF}\ DW01\ FTL3\ FM21 + DW14\ FTL2\ DVRB$$

illustrates the logical circuit which generates a trigger signal which triggers the CL2 flip-flop to its 1-state.

The preceding example demonstrates how the logical schematic diagrams of FIGS. 27–118 and 178–196 provide simple, compact and readily analyzed and related illustrations of the circuit structure of the components and of the circuit configurations interconnecting the components. This has been done by using signal designations in associations and combinations serving not only to identify the signal relationship, but also to define the circuit elements and their circuit relationships.

Memory

Memory **10**, illustrated diagrammatically in FIGS. 1 and 12, stores words which are to be processed, words which have resulted from processing, instruction words to direct data processing and auxiliary words for addressing and control. Memory **10** of the data processing system is adapted to store up to 32,768 words of 25 bits each. Of the memory locations provided for storage of information, forty-one are reserved for special purposes as indicated in the following table:

RESERVED MEMORY LOCATIONS

Memory Locations (Decimal)	Use
1–6	Storage of index words for addressing and address modification.
8	Working storage during execution of instructions 06 and 22.
10, 11	Storage of processor channel program interrupt words.
16–47	Storage of input/output channel control words which are used to control memory access during data transfer operations and to service program interrupt requests by the input/output channels.

The location of the control words associated with each of the eight channels of the Input/Output Control Circuit are as shown in the following table:

INPUT/OUTPUT CHANNEL CONTROL WORDS

Channel Number	Decimal Location of—			
	List Pointer Word (LPW)	Data Control Word (DCW)	Program Interrupt Word (PIW)	Program Interrupt Second Word (PSW)
0	16	17	18	19
1	20	21	22	23
2	24	25	26	27
3	28	29	30	31
4	32	33	34	35
5	36	37	38	39
6	40	41	42	43
7	44	45	46	47

The 32K memory of the data processing system comprises two 16K memory units. Each 16K memory unit comprises a storage element, a memory address register for storing the address of the memory location from which a word is to be extracted or in which a word is to be stored, address decode and driver circuits, inhibit drivers which control the writing of information into the addressed memory location, sense amplifiers through which information is read from the addressed memory location, data drivers for amplifying the output signals of these sense amplifiers, and memory timing and control logic circuits. FIG. 14 illustrates symbolically the elements of a 16K memory unit, the paths of data transfer between these elements and the paths of control signals. Details of the structure of the Memory are provided in FIGS. 15–18 and 27–43.

Core Memory **350** of FIG. 14 constitutes the storage element of a 16K memory unit and is of the coincident-current magnetic core type, which is well-known in the art. The structure and operation of such a storage element is described, for example, in the publication by C. V. L. Smith, "Electronic Digital Computers," chapter 12, McGraw-Hill Book Company, Inc., New York, 1959.

Core Memory **350** is adapted to store up to 16,384 words of 25 bits each, each bit being stored in a corresponding magnetic core. Core Memory **350** comprises an upper 8K unit and a lower 8K unit, each 8K memory unit comprising an upper 4K stack and a lower 4K stack, as shown in FIG. 16. The basic 4K stack comprises 25 planes, each plane having 4,096 magnetic cores. Each plane stores the corresponding bit for the 4,096 words stored in the 4K stack. The core plane comprises a matrix of magnetic cores arranged in 64 rows extending along the X-axis of the plane and 64 columns extending along the Y-axis of the plane. The location of the magnetic core in a matrix row and column is the address of the core and is identified by a number. Corresponding cores in all 25 planes are identified by the same number and store the 25 bits of a word. Therefore, Core Memory **350** may contain up to 16,384 addressable locations (00000–16,383) for storing up to 16,384 words.

Memory address information is received from B-Gates **317** of the Program Processor and is stored in B-Register **352** which is the memory address register. B-Register **352** comprises flip-flops B00–B14. The address is stored in B-Register **352** in 1's complement form, i.e., an address of 00000_8 is represented by flip-flops B00–B14 being in the 1 state. The logical schematic diagrams for the B-Register flip-flops are illustrated in FIGS. 27–30. B-Register **352** (FIG. 14) is adapted to receive the output signals of B-Gates **317** of the Program Processor upon application of the DBRS signal to logical gates connecting the output terminals of B-Gates **317** to corresponding input terminals of the B-Register flip-flops. Since the DBRS signal occurs at the beginning of a memory cycle, B-Register **352** has its contents updated at the start of each memory cycle. The flip-flops of B-Register **352** are reset to the 0-state by signal DBRP when power is initially applied to the data processing system and at the end of each memory cycle.

The stacks of Core Memory **350** are driven through arrays **355** and **356** of isolation diodes by Y-Axis and X-Axis Address Decode and Driver Circuits **357** and **358** respectively. The Y-Axis Address Decode and Driver Circuits **357** decode certain bits of the address in B-Register **352** and drive one of 64 column drive lines in the Y-axis of the selected 4K stack of Core Memory **350**. Similarly, X-Axis Address Decode and Driver Circuits **358** decode certain other bits of the address in B-Register **352** and drive one of 64 row drive lines in the X-axis of the selected 4K stack of Core Memory **350**. X-Axis and Y-Axis Address Decode and Driver Circuits **357** and **358** are enabled by signals FRDX, FRDY, FWRA, FWRB and FINH. Coincidence of current flow in these two drive lines in each plane of the selected 4K stack effects selection of a 25-bit storage location in Core Memory **350** in which a word is to be stored or from which a word is to be read. The logical schematic diagrams for the Y-Axis and X-Axis Address Decode and Driver Circuits are illustrated in FIGS. 37–40. A precharge voltage is applied to the stacks of Core Memory **350** through Address Decode and Driver Circuits **357** and **358** just prior to every read operation. This precharge voltage brings all lines through the selected stack to the same potential and minimizes the effect of stray capacitance. The precharge voltage is supplied by precharge circuit **359** under control of Memory Timing and Control Logic **368**.

M-Register **301** of the Program Processor temporarily stores a 24-bit word that is to be stored in Core Memory **350** or temporarily stores a 24-bit word that is read from Core Memory **350**. The 25th or parity bit WD24 of a word read from Core Memory **350** is applied through line **360** to Parity Check and Generation Logic **316** of the Program Processor while the parity bit DM24 of a word to be stored in Core Memory **350** is provided on line **361** by Parity Check and Generation Logic **316**. The flip-flops of M-Register **301** are cleared to the 0-state by signal DCLM. The logical schematic diagrams for M-Register **301** are illustrated in FIGS. 58–63.

Inhibit Drivers **362** control the storage of the 25-bit word from M-Register **301** and Parity Check and Generation Logic **316** in the addressed location of Core Memory **350**. A separate set of Inhibit Drivers is used for each 4K stack. The bits stored in flip-flops B12 and B13 of B-register **352** determine which set of Inhibit Drivers is enabled by the signal FINH. The twenty-four bits of the word in M-Register **301** and the parity bit from Parity Check and Generation Logic **316** determine whether a "1" or "0" is stored in the corresponding cores of the addressed memory location. The logical schematic diagrams for the Inhibit Drivers **362** are illustrated in FIGS. 41 and 42.

Sense Amplifiers **364**, when enabled by signal CSTR from generator **365**, amplify the signals representing the 25 bits read from the addressed memory location and apply the signals to Data Drivers **367**. Data Drivers **367** serve to further amplify the signals representing the bits of the word read from Core Memory **350**. After amplification in Data Drivers **367**, the twenty-four bits of the extracted word are stored in M-Register **301** and the parity bit WD24 is applied on line **360** to flip-flop MUJ of Parity Check and Generation Logic **316** in the Program Processor. Each 4K stack of Core Memory **350** has a separate set of Sense Amplifiers. The output signals of each set of Sense Amplifiers are applied to an OR-gate prior to application to Data Drivers **367**. The logical schematic diagrams for the Data Drivers **367** are illustrated in FIG. 43.

Memory Timing and Control Logic **368** includes a two-phase memory timing clock generator, a 4-bit counter and

various memory command and control flip-flops. Memory Timing and Control Logic 368 directs, through the signal paths indicated in FIG. 14, the timing sequence of the 16K memory unit for reading words from and for storing words in Core Memory 350. The 16K memory unit is enabled by Memory Timing and Control Logic 368 to perform five types of memory cycles, viz. read/restore, clear/write, read-only, write-only and read/delay/write. Memory Timing and Control Logic 368 also serves to provide control signal communication between the Program Processor and the Memory and to synchronize the timing sequence of the 16K memory unit with that of the Program Processor and the Input/Output Control Unit. This control signal communication and synchronization of timing is effected by signals DICR, $\overline{\text{DAMR}}$, DMWS, FMRD and FMWR provided by the Program Processor on lines 369, 370, 371, 372 and 373 respectively and by signals FWEN, FMAI, $\overline{\text{FMBU}}$, $\overline{\text{DMDA}}$ and DCLM provided by the Memory on lines 374, 375, 376, 377 and 378 respectively. Details of Memory Timing and Control Logic 368 are described in the following subsection entitled "Timing and Control."

CORE MEMORY ADDRESSING

FIG. 15 shows a portion of a memory plane and illustrates how the X-Axis Address Decode and Driver Circuits select a row of cores. A similar arrangement (not shown) is employed in the Y-axis of the plane to select a column of cores. The core at the intersection of the selected row and column is the addressed core of the plane. Twenty-four corresponding cores in the remaining 24 planes of the 4K stack are similarly selected. With reference to FIG. 15, the X-Axis Address Decode and Driver Circuits provide one of the signals $\overline{\text{XX01}}$, $\overline{\text{XX02}}$ . . . $\overline{\text{XX08}}$ if a word is to be read from Core Memory 350 or one of the signals XX01, XX02 . . . XX08 if a word is to be stored in Core Memory 350. This "select X-line" signal selects eight of the 64 rows in the plane, as illustrated. Similarly, the X-Axis Address Decode and Driver Circuits provide one of the signals $\overline{\text{XXB1}}$–$\overline{\text{XXB8}}$. This "select X-group" signal selects one of the eight rows selected by the "select X-line" signal to energize one row of the plane. The Y-Axis Address Decode and Driver Circuits similarly select a column of the plane. The action of the X-Axis and Y-Axis Address Decode and Driver Circuits thus drives one core of each plane to address a memory word.

FIG. 16 illustrates diagrammatically the relationship of the Address Decode and Driver Input Signals of FIGS. 37–40 to the 4K stacks of a 16K memory unit. 4K stack 400 is designated stack #1 or the lower 4K of the lower 8K while stack 401 is designated stack #2 or the upper 4K of the lower 8K. Stack 402 is designated stack #3 or the lower 4K of the upper 8K while stack 403 is designated stack #4 or the upper 4K of the upper 8K. As illustrated, the "select X-line" signals $\overline{\text{XX01}}$–$\overline{\text{XX08}}$ and XX01–XX08 are applied to stacks 400–403. Similarly, the "select Y-line" signals XY01–XY08 and $\overline{\text{XY01}}$–$\overline{\text{XY08}}$ are applied to stacks 400–403. The "select X-group" signals $\overline{\text{XXB1}}$–$\overline{\text{XXB8}}$ are applied to stacks 400 and 402 while the "select X-group" signals XXB1–XXB8 are applied to stacks 401 and 403. The "select Y-group" signals $\overline{\text{XYB1}}$–$\overline{\text{XYB8}}$ are applied to stacks 400 and 401 while the "select Y-group" signals XYB1–XYB8 are applied to stacks 402 and 403. The issuance of predetermined ones of the above line and group signals in response to a given address in B-Register 352 selects the corresponding 25 bit memory storage location during a memory read or write operation.

FIG. 17 illustrates diagrammatically the entire 32K memory comprising an upper 16K memory unit 410 and a lower 16K memory unit 411. The connection of memory units 410 and 411 with the Program Processor is indicated symbolically by lines 412 and 413 respectively. The selection of the upper 16K memory unit or the lower 16K memory unit is effected by the illustrated logical gating circuits which provide the signal DULS to one of the 16K memory units in response to the state of flip-flop B14 of the B-Register of the selected memory unit or in response to the 15th bit provided by B-Gates 317 of the Program Processor. The signal DULS starts the clock generator of the selected 16K memory unit. The 15th bit provided by B-Gates 317 serves to generate the DULS signal at the beginning of a memory cycle while the output of flip-flop B14 generates the DULS signal during a split cycle, e.g. a read/delay/write cycle.

The following table illustrates the relationship of the address in B-Register 352 to the line, group and stack selection signals and the function of each B-register flip-flop in selecting the addressed location of the 32K memory:

MEMORY ADDRESSING FROM B-REGISTER

B-Register Flip-Flops	Function
B00 B01 B02	Cause a particular "select X-line" signal to issue.
B03 B04 B05	Cause a particular "select X-group" signal to issue.
B06 B07 B08	Cause a particular "select Y-line" signal to issue.
B09 B10 B11	Cause a particular "select Y-group" signal to issue.
B12	Selects upper or lower 4K stack of upper or lower 8K unit.
B13	Selects upper or lower 8K unit of upper or lower 16K memory unit.
B14	Selects upper or lower 16K memory unit of 32K memory.

The following table, in conjunction with FIGS. 16 and 17, shows the function of flip-flops B12–B14 in selecting one of the eight 4K stacks in the 32K memory:

STACK SELECTION FROM B-REGISTER

Address Bits			Selected Stack	
DB14 FB14	FB13	FB12	Upper 16K Memory Unit	Lower 16K Memory Unit
0	0	0	#4	----------
0	0	1	#3	----------
0	1	0	#2	----------
0	1	1	#1	----------
1	0	0	----------	#4
1	0	1	----------	#3
1	1	0	----------	#2
1	1	1	----------	#1

Since Memory 10 of the data processing system comprises modular 4K stacks, the data processing system may be provided with memory capacities of 4K, 8K, 16K or 32K. In the event a memory capacity of less than 32K is employed, the Memory Address Invalid signal FMAI is employed to indicate to the Program Processor when a memory address exceeds the capacity of the Memory. The presence of the FMAI signal prevents the starting of the memory cycle. The MAI flip-flop (FIG. 33) is reset at the beginning of the next memory cycle in which a valid address is received by the Memory.

TIMING AND CONTROL

FIG. 18 illustrates diagrammatically in greater detail the Memory Timing and Control Logic of a 16K memory unit, as identified by reference numeral **368** in FIG. 14. The Memory Timing and Control Logic comprises Clock Generator **380**, Gated Clock Amplifiers **384**, **385** and **390**, CC-Counter **394** and Command and Control flip-flops **397**. The generation of memory clock pulses QCPG by Clock Generator **380** is controlled by the memory start-stop clock logic signals on input lines **381**. The memory clock pulses QCPG provided by Clock Generator **380** occur at a repetition rate of approximately 5.7 megacycles. The circuit of Clock Generator **380** is illustrated in FIG. 3.

Gated Clock Amplifiers **384** and **385** provide, in response to clock pulse QCPG and the gated clock amplifier input signals on input lines **387** and **388** respectively, clock pulses QCPA and QCPB which are out-of-phase by 175 nanoseconds. Clock pulses QCPC, provided by Gated Clock Amplifier **390** in response to memory clock pulse QCPG and the gated clock amplifier input signal on line **391**, occur in synchronism with memory clock pulse QCPG. The two-phase clock signals are used in the Memory to obtain finer timing of the 16K memory unit command flip-flops and other logic. The circuit of Gated Clock Amplifiers **384**, **385** and **390** is illustrated in FIG. 5. The logical schematic diagrams for memory Clock Generator **380** and memory Gated Clock Amplifiers **384**, **385** and **390** are illustrated in FIG. 40.

Operation of the 16K memory unit is divided into fourteen time periods, T00–T13. These time periods are defined by the four-stage CC-Counter **394** which comprises flip-flops CCA, CCB, CCC and CCD. The state of CC-Counter **394** is controlled by clock pulse QCPA and the logic input signals on line **395**. The logical schematic diagrams for the CC-Counter flip-flops are illustrated in FIG. 31. The relationship between the states of the flip-flops of CC-Counter **394** and the time periods of the 16K memory unit is illustrated in the following table:

MEMORY TIME PERIODS

Time Period	CC-Counter Flip-Flops			
	FCCD	FCCC	FCCB	FCCA
T00	0	0	0	0
T01	0	0	0	1
T02	0	0	1	0
T03	0	0	1	1
T04	0	1	0	0
T05	0	1	0	1
T06	0	1	1	0
T07	0	1	1	1
T08	1	0	0	0
T09	1	0	0	1
T10	1	0	1	0
T11	1	0	1	1
T12	1	1	0	0
T13	1	1	0	1

The Command and Control flip-flops employed in the Memory Timing and Control Logic are indicated in block **397** of FIG. 18. Flip-flop ARC (FIG. 32) is set to the 1-state during time period T02 to indicate that the address has been set in the B-Register **352**. The 1-output signal of flip-flop ARC sets and triggers the generator which provides the signal CSTR (FIG. 36). Flip-flop CMA (FIG. 32) is alternately set and reset by successive clock pulses QCPA to define one phase of two-phase Clock Generator **380**. Flip-flop CMB (FIG. 32) is alternately set and reset by successive clock pulses QCPB to define the second phase of two-phase Clock Generator **380**. Flip-flop IAR (FIG. 32) is set to the 1-state during a read/delay/write cycle to inhibit the resetting of the B-Register flip-flops at the end of the read cycle and to inhibit the setting of a new address into B-Register **352** at the start of the following write cycle. Flip-flop INH (FIG. 33) is set to the 1-state during time period T07 to enable the set of Inhibit Drivers associated with the selected 4K stack. Flip-flop MAI (FIG. 33) is employed in a system having a memory capacity less than 32K and is set to the 1-state in response to the receipt by the 16K memory unit of an address greater than the memory capacity.

Flip-flop MBD (FIG. 33) is set to the 1-state during time period T00 to terminate a memory cycle. Flip-flop MBU (FIG. 33) is set to the 1-state at the beginning of a memory cycle to indicate to the Program Processor that the 16K memory unit is busy and is reset to the 0-state a predetermined time period before the end of the memory cycle to indicate to the Program Processor that the 16K memory unit is approaching the end of the cycle. Flip-flop MBZ (FIG. 34) is set to the 1-state in response to the receipt of a Memory Read or a Memory Write command from the Program Processor and remains set to the 1-state during the period that the 16K memory unit is busy. Memory Data Available flip-flop MDA (FIG. 34) is set to the 1-state during time period T04 to indicate to the Program Processor that a word being read from the 16K memory unit is available to M-Register **301**. Flip-flops RDX and RDY (FIG. 34) are set to the 1-state during time periods T00 and T01 respectively to enable the X-Axis and Y-Axis Address Decode and Driver Circuits during a read operation. Flip-flops WRA and WRB (FIG. 35) are set to the 1-state during time period T07 to enable the X-Axis and Y-Axis Address Decode and Driver Circuits during a write operation. Write Enable flip-flop WEN (FIG. 35) is set to the 1-state in response to a Memory Write Command from the Program Processor and permits the appropriate Inhibit Drivers to be energized. The logical schematic diagrams for these Command and Control flip-flops are illustrated in FIGS. 32–35. The logical schematic diagrams of the logical combination signals employed in Memory Timing and Control Logic **368** and elsewhere in a 16K memory unit are illustrated in FIGS. 36 and 37.

GENERAL OPERATION AND TIMING

During the read/restore cycle, a word is read from the 16K memory unit, made available to the Program Processor and restored to its original location in the 16K memory unit. In a clear/write cycle, a word is read from the 16K memory unit, clearing the memory location, but is not made available to the Program Processor. A new word is then written into that location in the 16K memory unit. A read-only cycle causes a word to be read from the 16K memory unit and made available to the Program Processor, leaving the location from which the word was extracted cleared to zeros. During a write-only cycle, a word is written into a predetermined location. A read/delay/write cycle is similar to the basic read/restore cycle except that the word is modified in the Program Processor prior to returning it to its original location in the 16K memory unit.

FIG. 19 is a timing diagram which illustrates the timing of the 16K memory unit during a read/restore cycle. The read/restore cycle is initiated by a Memory Read command $\overline{\text{FMRD}}$ from the Program Processor. The leading edge of the $\overline{\text{FMRD}}$ signal starts the memory Clock Generator **380** which begins to generate 2-phase clock pulses QCPA and QCPB and clock pulse QCPC. The time at

which CC-Counter 394 flip-flops CCA–CCD change state is controlled by clock pulse QCPA. Flip-flops CMA and CMB define the phases of the Clock Generator 380. The first clock pulse QCPA starts the 4-bit CC-Counter 394 by setting flip-flop CCA to the 1-state to define time period T01. The Memory Busy flip-flop MBZ and the Memory Busy Level flip-flop MBU are simultaneously set. During time period T01, the address on the address lines from B-Gates 317 of the Program Processor is set into B-Register 352 by signal DBRS. Signal FMBU resets flip-flop MRD in the Program Processor. The X-Axis and Y-Axis Address Decode and Driver Circuits 357 and 358 are enabled by flip-flops RDX and RDY, which are set to the 1-state during time T01, to provide read drive currents to the cores of the selected 4K stack. Allow Memory Read signal DAMR is provided by the Program Processor.

The second clock pulse QCPA advances the CC-Counter 394 by one to define time period T02. Signal DCLM is provided by Memory Timing and Control Logic 368 at this time to clear the M-Register of the Program Processor preparatory to receiving the word being read from the 16K memory unit. During time period T02, the Address Received flip-flop ARC is set. The signal FARC sets and triggers the Generator 365 which provides the signal CSTR. A predetermined delay exists between the time when the Generator 365 is triggered and the generation of the signal CSTR, which will occur during time period T03 or T04 depending upon the delay setting. During time period T04, the signal FMDA is transmitted from Memory Timing and Control Logic 368 to the Program Processor to indicate to the Program Processor that the word is now available from the 16K memory unit. The FMDA signal from the 16K memory unit initiates the generation of clock pulses in the Timing Clock Generator in the Program Processor which then processes the extracted word. The MDA flip-flop is reset to the 0-state during time period T06.

Prior to the beginning of time period T07, the 16K memory unit has received the Memory Write command FMWR from the Program Processor. The Write Enable flip-flop WEN is then set. At the beginning of time period T07, the RDX and RDY flip-flops are reset to the 0-state to turn off the read currents. The Write Enable flip-flop WEN causes the Inhibit flip-flop INH to set and the set of Inhibit Drivers corresponding to the 4K stack in which the word is to be written is energized. The WRA and WRB flip-flops are also set during time period T07 to cause the current flow in the selected cores of the stack to reverse. During time periods T08–T12, the word is restored into the addressed memory location. At the beginning of time period T13, the Memory Busy Level flip-flop MBU is reset to the 0-state to indicate to the Program Processor that the 16K memory unit is completing its cycle. During time period T13, the Inhibit flip-flop INH is reset to the 0-state to de-energize the Inhibit Drivers. Flip-flop WRB is also reset during time period T13.

At the end of time period T13, CC-Counter 394 reverts to a count of zero to define time period T00. During time period T00, the End Memory Cycle flip-flop MBD is set to prevent additional QCPA clock pulses from being generated. The WRA, ARC, WEN and MBZ flip-flops are reset to the 0-state during time period T00. The 0-output signal $\overline{\text{FMBZ}}$ of flip-flops MBZ stops the memory Clock Generator 380. The DBRP signal is generated to reset the B-Register flip-flops to the 0-state. The MBD flip-flop is then reset to the 0-state to complete the read/restore cycle.

The timing sequence of the clear/write cycle is the same as that of the read/restore cycle except that the signal CSTR is not generated since the Program Processor does not furnish an Allow Memory Read signal DAMR to the 16K memory unit. The word in the addressed memory location is therefore not transmitted through Data Drivers 367 to M-Register 301. The portion of the cycle between time periods T07 and T13 is identical to the read/restore cycle, a new word in M-Register 301 being written into the cleared memory location.

The read-only cycle is identical to the read/restore cycle during time periods T01–T07. The read-only cycle is terminated at the end of time period T07 in the same manner that the read/restore cycle is terminated at the end of time period T13. When the Program Processor calls for a write-only cycle, CC-Counter 394 is immediately set to define time period T07 and the sequence from T07–T13 is identical to that of the read/restore cycle.

During a read/delay/write cycle, the Program Processor transmits the Memory Write Special signal DMWS to Memory Timing and Control Logic 368. The DMWS signal sets the Inhibit Address Register flip-flop IAR to inhibit the resetting to the 0-state of the B-Register flip-flops at the end of the read cycle. The IAR flip-flop also inhibits the setting of a new address into the B-Register 352 at the start of the write cycle which follows. Thus, during a read/delay/write cycle, a word is read from a given storage location in the 16K memory unit and transmitted to M-Register 301 of the Program Processor. The memory cycle then terminates without clearing B-Register 352. The extracted word is processed in the Program Processor and a write cycle is then initiated in memory by a Memory Write Command FMWR sent from the Program Processor. At the beginning of the write cycle, the address in B-Register 352 is retained and the word resulting from the processing is written into the same storage location from which the original word was extracted. The Inhibit Address Register flip-flop IAR is reset to the 0-state during time period T08.

Program Processor

The Program Processor exercises operational control over the data processing system and responds to a plurality of distinct instructions which are supplied thereto in the sequential order necessary to perform a particular data processing operation. The data which is processed in the Program Processor is supplied primarily from the Memory. This data is usually transferred into and out of the Memory under control of the Program Processor.

The Program Processor comprises the Central Processor Control Unit, the Arithmetic Unit and the Console. The Central Processor Control Unit and the Arithmetic Unit each comprise registers for temporarily storing data, logic circuits for transferring data through and between registers and flip-flops employed as control signal sources. The Central Processor Control Unit further includes a timing control unit for controlling the timing of the Program Processor. FIGURE 20 illustrates diagrammatically the elements of the Program Processor which store data, the paths of data transfer between these elements and certain control elements.

Central Processor Control Unit

The Central Processor Control Unit comprises the following registers, shown in FIG. 20: P-Register 310, D-Register 311, A-Register 312, I-Register 313, Accumulator Length Indicator Register 314, Accumulator Counter Register 315 and W-Selector Register 470. The Central Processor Control Unit also includes B-Gates 317, I-Register Decode Logic 480, W-Selector Decode Logic 482, Central Processor Control Logic 490, Parity Adder 472, Memory Output Parity flip-flop 474 and Word Parity Error flip-flop 476. W-Selector Register 470, I-Register Decode Logic 480, Central Processor Control Logic 490 and W-Selector Decode Logic 482 comprise the Operations Control Unit 318 of FIG. 12. Parity Adder 472, Memory Output Parity flip-flop 474 and Word Parity Error flip-flop 476 comprise Parity Check and Generation Logic 316 of FIG. 12. The timing of the data processing system is controlled by the Central Processor Control Unit through Start-Stop Clock

Logic 484, Timing Clock Generator 486 and TL-Counter 488 which comprise Timing Control Unit 319 of the block diagram of FIG. 12. The details of the circuit structure of the Central Processor Control Unit are provided in FIG. 21, which illustrates the Timing Clock Generator and Start-Stop Clock Logic; FIG. 23, which illustrates the Parity Check and Generation Logic; FIGS. 45–51, 64–67, 69, 70 and 74–92 which are the flip-flop input logical schematic diagrams of Program Processor Control Unit flip-flops employed in registers, counters and as control signal sources; FIGS. 93–114, which are logical combination signal diagrams; and FIGS. 117 and 118, which contain clock generator input signal diagrams, gated clock distribution driver input signal diagrams, and gated clock amplifier input signal diagrams.

As described in the preceding section "Data Processing System—Details," the operation code which directs the system operation in executing an instruction is stored in I-Register 313. I-Register 313 is a six-bit register for storing bits 18–23 of the instruction word which is to be executed by the Central Processor. The preceding section "Data Processing Operations" describes the operation code of the instruction word as having any one of 53 bit configurations, each configuration directing a fundamentally different data processing operation of the system. These operations were described generally in the preceding section, but are also described in detail in a following section entitled "Instructions." In the "Instructions" section, and in the sections immediately following thereafter, there is provided a detailed description of the operation of the Central Processor in performing each of its possible functions.

I-Register 313 is adapted to receive, by parallel transfer, the contents of flip-flops M18–M23 of the M-Register, upon application of working clock signal QMIX to the input gates which connect the output terminals of the M18–M23 flip-flops to corresponding input terminals of the IR0–IR5 flip-flops. The contents of I-Register 313 are applied to I-Register Decode Logic 480. The logical schematic diagrams of the I-Register are illustrated in FIGS. 69 and 70.

The operation code in I-Register 313 is decoded by I-Register Decode Logic 480 which provides a different output signal for each of the 53 possible bit configurations of the operation code in I-Register 313. The output signals of I-Register Decode Logic 480 are applied to the input gates of W-Selector Register 470. W-Selector Register 470 comprises flip-flops WR0–WR3, the states of which are determined by the output signals of I-Register Decode Logic 480 and by the previous contents of the register. The time at which the flip-flops of the W-Selector Register 470 change state is determined by clock signals received from Timing Clock Generator 486. The contents of W-Selector Register 470 are applied to W-Selector Decode Logic 482. The logical schematic diagrams of the W-Selector Register are illustrated in FIG. 77.

The contents of the W-Selector Register 470 are decoded by W-Selector Decode Logic 482 to provide an input to Central Processor Control Logic 490. W-Selector Decode Logic 482 also transfers signals indicative of the states of the W-Selector Register flip-flops 470 to TL-Counter 488.

Start-Stop Clock Logic 484 controls the initiation and termination of clock pulse generation by Timing Clock Generator 486. Start-Stop Clock Logic 484 receives synchronizing signals from Memory which enable the Start-Stop Clock Logic 484 to synchronize the operations of the Program Processor with the operations of Memory. Start-Stop Clock Logic 484 also receives at its input terminals signals indicative of the state of TL-Counter 488. The logical schematic diagrams of the flip-flops RCK and ISP of Start-Stop Clock Logic 484 are illustrated in FIG. 83.

Timing Clock Generator 486 generates clock signals which are employed to control the timing of the Program Processor and the Input/Output Control Unit of the Central Processor. The output signals of Timing Clock Generator 486 are transmitted to W-Selector Register 470, TL-Counter 488 and Input/Output Control Unit 13. A schematic diagram of Timing Clock Generator 486 is illustrated in FIG. 21.

TL-Counter 488 comprises flip-flops TL0–TL5 and serves as a six-bit ring counter in the Central Processor to control the timing within the blocks of micro-operations defined by the W-Selector Register 470 and the R-Selector Register of Input/Output Control Unit 13. TL-Counter 488 receives inputs from W-Selector Decode Logic 482 and Timing Clock Generator 486. The output signals of TL-Counter 488 are transmitted to Input/Output Control Unit 13, Start-Stop Clock Logic 484 and Central Processor Control Logic 490. The logical schematic diagrams of the TL-Counter flip-flops are given in FIGS. 75 and 76.

Central Processor Control Logic 490 comprises all logic necessary to the control of data transfers and other operations within the Central Processor. Central Processor Control Logic 490 receives inputs from W-Selector Decode Logic 482, TL-Counter 488 and Timing Clock Generator 486. The flip-flops of Central Processor Control Logic 490 which serve as control signal sources are listed in block 492 of FIG. 20. The logical schematic diagrams for these flip-flops are given in FIGS. 78–92.

P-Register 310 is a 15-bit counter, comprising flip-flops P00–P14. The sequence in which successive instructions are executed is controlled by P-Register 310, which serves as the program counter in the Program Processor. The count in P-Register 310 is incremented by one in response to signal DPCU, each time that a P-sequence word is employed in the Program Processor. The count in P-Register 310 is used to provide the address of one of the following types of words in Memory:

(a) Instruction Word,
(b) P-Sequence AMS Word,
(c) P-Sequence SAS Word.

The instruction word or auxiliary word address in P-Register 310 is transferred through Adder 302 to D-Register 311. Memory 10 is addressed from D-Register 311 to extract the instruction word or the auxiliary word from Memory.

P-Register 310 is adapted to receive, by parallel transfer, the bit position 0–14 adder output signals, upon application of working clock signal QSPX to the input gates which connect the appropriate adder output terminals to the corresponding input terminals of the P00–P14 flip-flops. The program can alter the contents of P-Register 310 by making use of one of the eight branch instructions (BRU, SPB, BRE, BRL, BRG, BRC, BRZ and BRM). The contents of P-Register 310 are not affected by any other operation in the Central Processor, except as described above. The logical schematic diagrams of P-Register 310 are given by FIGS. 64–67. The P-Register flip-flops are of the pulse-pedestal type. The parenthetical terms in the logical schematic diagrams of flip-flops P00–P14 serve as the clock signal inputs to the respective flip-flops. The parenthetical terms in the logical schematic diagrams of flip-flops P01–P14 indicate that each of the flip-flops changes state only when the next lower-order flip-flop is reset to the 0-state.

D-Register 311 is a 15-bit register, comprising flip-flops D00–D14, which normally contains the address of a data word, an auxiliary word or an instruction word. D-Register 311 is adapted to receive, by parallel transfer, the bit position 0–14 adder output signals, upon application of working clock signal QSDX to the input gates which connect the appropriate adder output terminals to

the corresponding input terminals of the D00–D14 flip-flops. During the execution of instructions employing a double, triple or quadruple length Accumulator, D-Register 311 is employed so that operand data words are accessed in the proper sequence to match the accumulator words. The address in D-Register 311 is decremented by one, in response to signal DDCD, to provide successive operand word addresses. D-Register 311 also contains shift control information during execution of Instruction 22, Shift (SHG). The contents of D-Register 311 may be applied to either B-Gates 317 or Adder 302. The logical schematic diagrams of the D-Register are given by FIGS. 48–51. The D-Register flip-flops are of the pulse-pedestal type. The parenthetical terms in the logical schematic diagrams of flip-flops D00–D14 serve as the clock signal inputs to the respective flip-flops. The parenthetical terms in the logical schematic diagrams of flip-flops D01–D14 indicate that each of the flip-flops change state only when the next lower-order flip-flop is set to the 1-state.

A-Register 312 is a 13-bit register comprising flip-flops A02–A14. A-Register 312 is employed to store the present address of the 4-word Accumulator in Memory. The actual address contained in A-Register 312 is the address of the most-significant word of the Accumulator. Information is transferred to A-Register 312 from E-Register 303 upon application of working clock signal QEAX to the input gates connecting the appropriate adder output terminals to corresponding input terminals of the A14–A02 flip-flops. The information in A-Register 312 is transferred either to B-Gates 317 or to E-Register 303. The flip-flops of A-Register 312 may be reset to the 0-state by signal DCLA, clearing A-Register 312. The logical schematic diagrams of A-Register 312 are given by FIGS. 44–47.

Accumulator Length Indicator Register 314, comprising flip-flops PL2 and PL1, stores the working length of the Accumulator which may be 1, 2, 3 or 4 words. The relationship between the contents of Accumulator Length Indicator Register 314 and the accumulator working length are indicated in the following table:

Accumulator Working Length	FPL2	FPL1
Quadruple	0	0
Triple	0	1
Double	1	0
Single	1	1

The contents of Accumulator Length Indicator Register 314, in conjunction with the contents of A-Register 312, form a full 15-bit address which is the address of the most-significant word of the working Accumulator. Flip-flops PL2 and PL1 are adapted to receive, by parallel transfer, the contents of either flip-flops IR1 and IR0 of I-Registers 313, flip-flops E01 and E00 of E-Register 303 or flip-flops D10 and D09 of D-Register 311 upon application of the appropriate signal DILX, DEAP or DDLX to the input gates to Accumulator Length Indicator Register 314. The logical schematic diagrams of Accumulator Length Indicator Register 314 are given in FIG. 47.

Accumulator Counter Register 315, comprising flip-flops AC2 and AC1, supplements the address in A-Register 312 to form the address of one of the four words of the Accumulator. The relationship between the contents of Accumulator Counter Register 315 and the accumulator word addressed by the combined contents of Accumulator Counter Register 315 and A-Register 312 is given in the following table:

Accumulator Word Addressed	FAC2	FAC1
Accumulator Word A	0	0
Accumulator Word B	0	1
Accumulator Word C	1	0
Accumulator Word D	1	1

The count in Accumulator Counter Register 315 is incremented or decremented during execution of certain instructions by the Central Processor to address predetermined words of the Accumulator. The logical schematic diagrams of the Accumulator Counter Register 315 are given in FIG. 70.

Parity Adder 472, Memory Output Parity flip-flop 474 and Word Parity Error flip-flop 476 comprise the Parity Check and Generation Logic of the Program Processor. Parity Adder 472 comprises three ranks of full adders, the first rank comprising adders JA0–JA7, the second rank comprising adders JB0–JB2 and the third rank comprising adder JE0. Parity Adder 472 samples the contents of the M-Register to either check or generate parity. Flip-flop MUJ, identified by reference numeral 474, stores the 25th or parity bit, represented by signal WD24 on line 360, of a word read from Memory. The output of flip-flop MUJ is applied to adder JB2 of the second rank during the parity check of a word read from Memory. Flip-flop WJE, identified by reference numeral 476, is set to the 1-state by the output of third rank adder JE0 if a parity error is detected. When the word in the M-Register is to be stored in Memory, Parity Adder 472 generates the correct parity bit, represented by signal DM24 on line 361, and transmits the parity signal DM24 to the Memory Inhibit Drivers. Signal DM24 is stored as the 25th bit of the word in the addressed memory location. The schematic diagram for the Parity Check and Generation Logic is given in FIG. 23.

B-Gates 317 transmit addresses from the Program Processor and the Input/Output Control Unit to the B-Register of Memory. B-Gates 317 receive inputs from D-Register 311, A-Register 312, Accumulator Length Indicator Register 314, Accumulator Counter Register 315, M-Register 301, Q-Register 306 and the Input/Output Control Unit 13. Address information is applied to B-Gates 317 by Input/Output Control Unit 13 during data interrupts and program interrupts. Gating signals DDBX, DMBX, DABX and DQBX are applied to the B-Gates 317 to enable the appropriate address to be transmitted to the B-Register. The logical schematic diagrams of B-Gates 317 are given in FIGS. 94 and 95.

TB-Counter 494, comprising flip-flops TB2 and TB1, serves as a counter during execution of certain instructions by the Central Processor. The logical schematic diagrams of TB-Counter 494 are given in FIG. 74.

INSTRUCTION DECODE

Data processing operations in the Central Processor are performed under the control of instruction words. The operation code of each instruction word which directs the system operation is stored in I-Register 313 of FIG. 20. The operation code comprises bits 18–23 of the instruction word and these bits are stored in flip-flops IR0–IR5 respectively of I-Register 313. The operation code in I-Register 313 controls the type of operation to be executed by the Central Processor. The address field of the instruction word may be modified in the Arithmetic Unit and is normally transmitted to D-Register 311 preparatory to instruction execution.

The operation code of an instruction word may have any one of 53 bit configurations, each configuration representing a different one of 53 instruction codes. Each

instruction code directs a fundamentally different processing operation in the system. The data processing operation corresponding to each instruction code is illustrated in the following table:

INSTRUCTION DECODE

Class	Type FIR2,1,0 / FIR5,4,3	DSGI	DDBI	DTPI	DQDI	DSGI	DDBI	DTPI	DQDI
		000	001	010	011	100	101	110	111
DMIS	000	HLT	CMI	CDA	CAA	CMM	EDT	MOV	GEN
DBCL	001	BRU	BRM	BRG	BRE	BRL	BRZ	BRC	SPB
DSHC	010	EXP	IMP	SHG	RIM	ANM	RXM	VLD	VLM
DMEC	011	MFM	MFI	MTA	AMI	ABM	SBM	LAL	SAL
DLSC	100	LDS	LDD	LDT	LDQ	STS	STD	STT	STQ
DADC	101	ADS	ADD	ADT	ADQ	AMS	AMD	AMT	AMQ
DSDC	110	SDS	SDD	SDT	SDQ				CPO
Group		DIGA	DIGB	DIGC	DIGD	DIGE	DIGF	DIGG	DIGH

In the table, the groupings of three alphanumeric characters represent the mnemonic code assigned to each instruction, as indicated in the section under the heading "Data Processing Operations." The numeral assigned to each instruction in the above-identified section of the description is the binary instruction code of the instruction, as illustrated in the table, expressed as an octal number. On the upper, lower and left-hand sides of the table are listed signals which issue to indicate that the instruction falls into a particular type, class or group. The signals DSGI, DDBI, DTPI and DQDI at the top of the table indicate that the instructions in the corresponding columns employ a single, double, triple or quadruple accumulator, respectively. Similarly, the signals at the bottom of the table indicate that the instructions in the corresponding columns fall into instruction groups A–H. The signals on the left-hand side of the table classify the instructions in the corresponding rows as Miscellaneous (DMIS), Branch (DBCL), Shift (DSHC), Move (DMEC), Load and Store (DLSC), Add Decimal (DADC) and Subtract Decimal (DSDC) instructions.

TIMING CLOCK GENERATOR

Timing Clock Generator **486** of FIG. 20, which provides working and control clock signals to the Central Processor, is shown in greater detail in FIG. 21. FIGURE 21 also shows details of Start-Stop Clock Logic **484** which serves to synchronize the timing of the Program Processor operations with operations in the Memory and in the Input/Output Control Unit. The Start-Stop Clock Logic provides signals DSPG and $\overline{\text{DSGP}}$ (FIG. 107) to Clock Generator **560**. Signal DSPG initiates generation of clock pulses QCCK by Clock Generator **560** while signal $\overline{\text{DSGP}}$ enables an AND-gate in the feedback loop of the Clock Generator **560** to maintain clock pulse generation.

The time at which generation of clock pulses by Clock Generator **560** is initiated is a function of synchronizing signals $\overline{\text{DMBU}}$ and DMDA from Memory. The input signals to Driver **562**, which furnishes logical combination signal DSPG, includes logical combination signal DVRS. Because of the implementation of the logical structure with NAND-Gates **564** and **565**, DVRS is at a voltage level corresponding to a binary 1 only when the output signal of each of the NAND-Gates **564** and **565** is a binary 1. Thus, when synchronizing signal DMBU from Memory is a binary 0 ($\overline{\text{DMBU}}$ is a binary 1), signal DVRS is a binary 1. Alternatively, when synchronizing signal $\overline{\text{DMDA}}$ from Memory is a binary 0 (DMDA is a binary 1), and either signal $\overline{\text{DSRK}}$ or DIRK is a binary 0, signal DVRS is also a binary 1. Assuming the other input signals to Line Driver **562** to be at the appropriate level, Start Pulse Generator signal DSPG is provided to Clock Generator **560** when either of the above conditions of the memory synchronizing signals is satisfied. Synchronization of the Program Processor Clock Generator **560** with the Memory Clock Generator **380** is thus attained.

Generation of clock pulses by Clock Generator **560** is terminated when Run Clock flip-flop RCK is reset to the 0-state. The Start-Stop Clock Logic provides signal $\overline{\text{DSGP}}$ to Clock Generator **560** to control the termination of generation of clock pulses QCCK. Signal $\overline{\text{DSGP}}$ has a voltage level of binary 0 when input signals FISP and $\overline{\text{FRCK}}$ to NAND-Gate **568** correspond to a binary 1 to stop Clock Generator **560**. Input signal FSRE to NAND-Gate **569** is provided to inhibit clock pulse generation upon initial application of power to the system.

Clock Generator **560** generates clock pulses QCCK. The circuit of Clock Generator **560** is illustrated in FIG. 3. In addition to Start Pulse Generator signal DSPG and signal $\overline{\text{DSGP}}$ from the Start-Stop Clock Logic, the 0-output signals of Memory Read flip-flop and Manual Reset flip-flop MRE are applied to Clock Generator **560**. When either signal $\overline{\text{FMRD}}$ or $\overline{\text{FMRE}}$ is a binary 0, i.e. when a Memory Read Command has been issued by the Central Processor or when the Reset Computer switch of the Console is being actuated, Clock Generator **560** is inhibited from generating clock pulses QCCK.

The clock pulses QCCK provided by Clock Generator **560** are applied to Gated Clock Distribution Drivers **570–575**. Gated Clock Distribution Drivers **570–574** are gated by signal FRCK while Distribution Driver **575** is gated by signal $\overline{\text{FRCK}}$. The output signals QCLK of Gated Clock Distribution Drivers **570–574** drive Gated Clock Amplifiers **580–584** to provide the indicated working clock signals to the Central Processor. The output signal QCKC of Gated Clock Distribution Driver **575** is applied to Gated Clock Amplifiers **585** to provide the indicated control clock signals to the Central Processor. In addition to signals QCLK and QCKC, Gated Clock Amplifiers **580–585** receive logic input signals on lines **590–595**, respectively. The logical schematic diagrams for Gated Clock Distribution Drivers **570–575** are illustrated in FIG. 117. The logical schematic diagrams for Gated Clock Amplifiers **580–585** are illustrated in FIGS. 117 and 118. The logical schematic diagram for Clock Generator **560** is illustrated in FIG. 117.

TL-COUNTER

TL-Counter **488** of FIG. 20 is a 6-bit ring counter comprising six flip-flops. The logical schematic diagrams for the TL-Counter flip-flops are illustrated in FIGS. 75 and 76. TL-Counter **488** defines six discrete time periods TL0–TL5. The signals representing the state of the TL-Counter, which defines one of the six time periods, are employed to gate Clock Amplifiers in the Central Processor. The out-

put signals of the Clock Amplifiers in turn cause corresponding micro-operations to be performed in the Central Processor. For example, the Gated Clock Amplifier which provides working clock signal QCNX is enabled by the signal FTL0, defining time period TL0, to effect a transfer of the contents of CC-Register **304** to N-Register **305**.

The count of TL-Counter **488** is advanced by working clock signal QTLP. When started at TL0, TL-Counter **488** stops the generation of pulses by Clock Generator **486** at time period TL2 and remains in that state until Timing Clock Generator **486** is again started; when started at TL2, TL-Counter **488** stops the generation of clock pulses by Timing Clock Generator **486** at TL0 and remains in the TL0 state until Timing Clock Generator **486** is again started. The state of TL-Counter **488** may be changed from TL0 to TL2 by control clock signal QTC0 and may be changed from TL2 to TL0 by control clock signal QTC2, as indicated by the logical schematic diagrams for flip-flops TL0 and TL2. This non-sequential change of state of TL-Counter **488** is employed to alter the normal sequence of performance of micro-operations in the Central Processor, for example, during a write-only sequence.

W-SELECTOR REGISTER

W-Selector Register **470** of FIG. 20 is a 4 bit register comprising flip-flops WR0–WR3. Fifteen of the sixteen possible states of W-Selector Register **470** are decoded by W-Selector Decode Logic **482** to provide corresponding signals W00–W12, W14 and W15. Each state of W-Selector Register **470**, as represented by the the output signal of W-Selector Decode Logic **482**, causes a corresponding block of micro-operations to be performed by the Central Processor. Each of the fifteen blocks of micro-operations performs some basic macro-operation in the Central Processor, such as instruction fetch, etc. The sequence of blocks of macro-operations selected by W-Selector Register **470** (W-sequence) is a function of the instruction being executed by the Ceneral Processor. The logical schematic diagrams for the flip-flops of W-Sequence Register **470** are illustrated in FIG. 77.

At the end of execution of an instruction, the W-Selector Register is normally set to W00. During the W00 block of micro-operations, the next instruction is extracted from Memory. Subsequent sequencing of the W-Selector Register is controlled by the operation code of the extracted instruction.

Thus, the instruction being executed determines the sequence of states of W-Selector Register **470** so that an appropriate sequence of macro-operations (W-sequence) is performed to execute the instruction. Each state of W-Selector Register **470** causes a corresponding macro-operation to be performed by causing a corresponding series or block of micro-operations to be performed in the Central Processor. The timing of the performance of the micro-operations in a block is controlled by TL-Counter **488** and by the working clock and control clock signals generated by Timing Clock Generator **486**. Central Processor Control Logic **490** receives output signals from W-Selector Decode Logic **482**, TL-Counter **488** and Timing Clock Generator **486** to control the execution of instructions by the Central Processor.

Instruction execution during the W-sequence may be halted by a data interrupt initiated by the Input/Output Control Unit **13**. During a data interrupt, the Central Processor enters a data transfer sequence (R-sequence). The W-Selector Register is preset to define the next block to be performed in executing the current instruction. The output signals of the W-Selector Decode Logic **482** are inhibited until the R-sequence is terminated. At this time, the output signals of W-Selector Decode Logic **482** are enabled and instruction execution continues in the W-sequence.

TIMING CLOCK GENERATOR AND TL-COUNTER TIMING

Each block of micro-operations is divided into a first half-sequence, identified by the suffix R, e.g. block W04R, and a second half-sequence, identified by the suffix S, e.g. block W04S. The first half-sequence timing is controlled by states TL0 and TL1 of TL-Counter **488**. The second half-sequence timing is controlled by states TL2, TL3, TL4 and TL5 of TL-Counter **488**. Certain of the blocks of micro-operations, viz. blocks W02, W03, W04, W05, W06, W10, W11, W12 and W15, can be split or divided so that only the first half-sequence or the second-half sequence of the block of micro-operations is performed. Such a split sequence is enabled by changing the state of TL-Counter **488** from TL0 to TL2 or from TL2 to TL0. In the remaining blocks of micro-operations, viz. blocks W00, W01, W07, W08, W09 and W14, both the first and the second half-sequences of the block of micro-operations are performed when the state of W-Selector Register **470** calls for that block.

FIGURE 22 is a timing diagram which illustrates the timing of Timing Clock Generator **486**, TL-Counter **488** and certain control signals during the performance of blocks of micro-operations. FIGURE 22 also illustrates the timing for split sequences wherein only the first half-sequence or the second half-sequence of a block of micro-operations is performed. The diagram of FIG. 22 is divided into portions 22A–22D to illustrate four different timing sequences.

Portion 22A of FIG. 22 illustrates the timing for a full block of micro-operations including both the first and the second half-sequences of the block. During the first half-sequence, the word is read from Memory and during the second half-sequence, the word is restored to Memory to comprise a read/restore sequence which calls for a read/restore cycle in Memory. Assuming that the operation code in I-Register **313** has caused the flip-flops of W-Selector Register **470** to be preset to an appropriate state and that TL-Counter **488** is at TL0, Start-Stop Clock Logic **484** initiates generation of clock pulses QCCK by Timing Clock Generator **486** upon receipt of signal $\overline{\text{FMBU}}$ from Memory. Since flip-flop RCK is reset to the 0-state, control clock signals QTRK, QTSP and QTC0 issue in response to distribution driver clock pulse QCKC. Control clock signal QTRK sets flip-flop RCK to the 1-state. The signal combination QTC0 FTL0 causes certain micro-operations of the block to be performed. With time period TL0 defined by TL-Counter **488**, certain other micro-operations are performed by the Central Processor. The next clock pulse QCCK causes distribution driver clock pulse QCLK to issue to provide working clock signals QTLP and QTCK. The signal combination QTCK FTL0 at the end of time period TL0 causes additional micro-operations of the block to be performed in the Central Processor. Working clock signal QTCK sets flip-flop ISP to the 1-state while working clock signal QTLP advances the TL-Counter to TL1.

During time period TL1, Memory Read Command FMRD is sent to Memory to initiate a read operation in Memory. During time period TL1, additional micro-operations corresponding to the block defined by W-Selector Decode Logic **482** are performed. Upon generation of the next clock pulse QCCK by Timing Clock Generator **486**, working clock signals QTLP and QTCK again issue. The signal combination QTCK FTL1 causes appropriate micro-operations to be performed. The signal combination QTLP FTL1 resets flip-flop RCK to the 0-state. Working clock signal QTLP advances TL-Counter **488** to TL2. The signal $\overline{\text{FRCK}}$ causes the gate in the feedback path of Clock Generator **560** to be disabled, stopping the generation of clock pulses by the Timing Clock Generator **486**. Because of the application of a clock pulse QCCK to the feedback path of Clock Generator **560** prior to disabling of the gate, Timing Clock Generator **486** produces one more pulse at its output terminal. This pulse causes clock pulse QCKC to be generated by the Gated Clock Distribution Driver enabled by $\overline{\text{FRCK}}$. Control clock signal QTSP

then issues to reset flip-flop ISP to the 0-state. Since signal combination FISP FRCK then exists, Start-Stop Clock Logic 484 is ready to again initiate pulse generation in Timing Clock Generator 486 upon receipt of synchronizing signal FMBU or FMDA from Memory. TL-Counter 488 remains at TL2 until Timing Clock Generator 486 is again started. Time periods TL0 and TL1 define the first half-sequence of the block of micro-operations.

Upon completion of the read operation in Memory, signal DMDA is provided by the Memory to start Timing Clock Generator 486. Since flip-flop RCK is reset to the 0-state, control clock signals QTRK and QTSP issue in response to distribution driver clock signal QCKC. The signal combination QTRK FTL2 causes certain micro-operations to be performed. QTRK resets flip-flop RCK to the 1-state. Working clock signals QTLP and QTCK again issue in response to the next clock signal QCCK. Signal combination QTCK FTL2 causes Memory Write command FMWR to issue and certain micro-operations to be performed. QTLP advances TL-Counter 488 to TL3.

Since flip-flop RCK is in the 1-state, working clock signal QTLP is provided in response to each clock pulse QCCK to advance TL-Counter 488 through successive states to TL5. During time periods TL3, TL4 and TL5 and in response to signal combinations QTCK FTL3, QTCK FTL4 and QTCK FTL5, certain other micro-operations are performed to complete the series of micro-operations corresponding to the block. At QTLP FTL5, flip-flop RCK is reset to the 0-state to terminate generation of clock pulses in Timing Clock Generator 486. The last pulse circulating in the feedback delay line causes control clock signal QTSP to issue, resetting flip-flop ISP to the 0-state, readying Start-Stop Clock Logic 484 to again start Timing Clock Generator 486 upon receipt of a synchronizing signal FMBU or DMDA from Memory. TL-Counter 488 remains at TL0 while the Memory performs the write operation. Time periods TL2, TL3, TL4 and TL5 comprise the second half-sequence of a block of micro-operations.

The timing within the first and second half-sequences of a block of micro-operations has been illustrated by the read/restore sequence shown in portion 22A of FIG. 22. During the first half-sequence, TL-Counter 488 starts at TL0 and stops at TL2 to permit the Memory to complete a read operation. Time periods TL0 and TL1 and signal combinations QTC0 FTL0, QTRK FTL0, QTCK FTL0 and QTCK FTL1 control the sequence of micro-operations performed during the first half-sequence of the block. Upon completion of the memory read operation, Timing Clock Generator 486 is started and TL-Counter 488 is advanced from TL2 back to TL0 during the second half-sequence of the block of micro-operations. At TL5, Timing Clock Generator 486 is again stopped to permit the Memory to complete a write operation. Time periods TL2, TL3, TL4 and TL5 and signal combinations QTRK FTL2, QTCK FTL2, QTCK FTL3, QTCK FTL4 and QTCK FTL5 control the sequence of micro-operations performed during the second half-sequence of the block.

Portion 22B of FIG. 22, illustrates a split sequence wherein only the first half-sequence of a block of micro-operations is performed. The timing of this read-only sequence is identical to that of the first half-sequence of the read/restore sequence illustrated in portion 22A. As in portion 22A, TL-Counter 488 stops at TL2 and the generation of clock pulses by Timing Clock Generator 486 is terminated.

Portion 22C of FIG. 22 illustrates the timing of a read/write sequence immediately following a read-only sequence. At the termination of the read-only sequence, TL-Counter 488 is at TL2. Upon receipt of synchronizing signal FMBU from Memory, Timing Clock Generator 486 is started. Control clock signal QTC2 issues to change the state of TL-Counter 488 from TL2 to TL0. The remainder of the read/write sequence of portion 22C of FIG. 22 is the same as that of portion 22A.

Portion 22D of FIG. 22 illustrates the timing of a write-only sequence following a read/write sequence. At the end of the read/write sequence, TL-Counter 488 is at TL0. At the beginning of the write-only sequence, Timing Clock Generator 486 is started by synchronizing signal FMBU from Memory. Control clock signal QTC0 issues to change the state of TL-Counter 488 from TL0 to TL2. The second half-sequence of the block of micro-operations is then performed. The remainder of the write-only sequence of portion 22D of FIG. 22 is identical to the second half-sequence of portion 22A.

As indicated in the sequences of FIG. 22 and in the Start-Stop Clock Logic shown in FIG. 21, the generation of clock pulses QCCK by Timing Clock Generator 486 can be initiated by either of the synchronizing signals FMBU or DMDA from Memory. Memory Busy Level signal FMBU starts Timing Clock Generator 486 for each first half-sequence, for each second half-sequence following another second half-sequence, and for each second half-sequence following a first half-sequence and using a different address than the first half-sequence. Snychronizing signal DMDA is employed to start Timing Clock Generator 486 only for a second half-sequence following a first half-sequence and using the same address as the first half-sequence and for any sequence following a first half-sequence which does not use the Memory.

MEMORY PARITY CHECK AND GENERATION

The basic unit of information in the Central Processor is the word consisting of 24 binary digits plus a 25th binary digit used in Memory for parity checking. Each time that a word is stored in the M-Register and is to be written into Memory, the parity bit is automatically generated and transmitted to Memory as the 25th bit of the word. If the number of 1-bits in the word is odd, a 1-bit is generated for the parity bit position; if the number of 1-bits in the word is even, the parity bit remains a binary 0. Thus, all words in Memory will have an even number of 1-bits.

As words are read from Memory and stored in the M-Register, parity is checked to detect errors in the word. If the number of 1-bits in the 25-bit word extracted from Memory is odd, a parity error signal is generated to set Parity Error flip-flop WJE (FIG. 92) to the 1-state. Provision is also made to store parity during shift, multiply, divide and implode operations.

FIGURE 23 illustrates in detail the Parity Check and Generation Logic shown diagrammatically in FIG. 12. Parity Check Generation Logic 316 serves to check parity of a word read from Memory and to generate the parity bit for a word to be stored in Memory. The Parity Check and Generation Logic 316 includes three ranks of full adders. The circuit for a full adder is illustrated in FIG. 11. The first-rank comprises full adders JA0–JA7. The output signals of the 24 flip-flops of M-Register 301 are applied to the first-rank of adders, each full adder having three input signals from M-Register 301. The second-rank comprises full adders JB0, JB1 and JB2. The $\overline{\text{Sum}}$ ($\overline{S}$) output signals of full adders JA0–JA2 and JA3–JA5 are applied to the input terminals of full adders JB0 and JB1, respectively. Full adder JB2 has as its inputs the $\overline{\text{Sum}}$ ($\overline{S}$) outputs of adders JA6 and JA7 and signal DZB2. Signal DZB2 is a binary 1 when Memory Output Parity flip-flop MUJ is set to the 1-state during a parity check or during certain parity generating operations. Third-rank adder JE0 receives the Sum (S) output signals of second-rank adders JB0–JB2.

During a parity check of a word read from Memory, the Sum (S) outputs of second-rank adders JB0 and JB1 are either both binary 0's or both binary 1's if signals FM00–FM17 contain an even number of binary 0's. The sum output of one of the adders JB0 and JB1

will be a binary 1 and the other output a binary 0 if an odd number of binary 0's are contained in the input signals FM00–FM17 to the first rank of adders.

The Sum (S) output of adder JB2 will be a binary 0 unless an odd number of binary 0's are present in inputs FM18–FM23 to first-rank adders JA6 and JA7 and flip-flop MUJ is set to the 0-state, or unless the inputs FM18–FM23 to adders JA6 and JA7 contain an even number of binary 0's and flip-flop MUJ is set to the 1-state.

The Sum (S) output of third-rank adder JE0 is a binary 0 if an even number of binary 1 signals have been applied to its input terminals, indicating a correct parity check. If an odd number of binary 1 signals are applied to the input terminals of adder JE0, the Sum (S) output signal is a binary 1 and sets Word Parity Error flip-flop WJE to the 1-state to indicate a parity error.

When the word in M-Register **301** is to be stored in Memory, Parity Check and Generation Logic **316** samples the word in M-Register **301** and generates the correct parity bit so that the word stored in Memory has an even number of binary 1's. During generation of a parity bit, Memory Output Parity flip-flop MUJ is set to the 0-state by signal DGEP and the trigger signal to the set-input terminal of Word Parity Error flip-flop WJE is inhibited by signal FMRJ. If an even number of binary 0's are contained in input signals FM00–FM17, the Sum (S) output signals of adders JB0 and JB1 will be both binary 0's or both binary 1's. If the number of binary 0's in the input signals FM18–FM23 are even, the Sum (S) output of adder JB2 will be a binary 0. An even number of binary 1 input signals will thus be provided to adder JE0 and the sum output signal MJE0 of adder JE0 will be a binary 0. The parity signal DM24 to the Memory Inhibit Drivers will thus be a binary 0. If an odd number of binary 0's are contained in first-rank adder input signals FM00–FM17 and an even number of binary 0's are contained in first-rank adder input signals FM18–FM23 or if an even number of binary 0's are contained in input signals FM00–FM17 and an odd number of binary 0's are contained in input signals FM18–FM23, an odd number of binary 1's will be applied to the input terminals of adder JE0 and Sum (S) output signal MJE0 will be a binary 1. Signal DM24 will then be a binary 1 to provide an even number of binary 1's in the 25-bit memory word.

The control signal source flip-flops for Parity Check and Generation Logic **316** include flip-flops MRJ (FIG. 86), WJE (FIG. 92), SP1 (FIG. 76) and SP2 (FIG 76). Word Parity Error flip-flop WJE is set to the 1-state by signal MJE0 to indicate a parity error in a word read from Memory. Flip-flop MRJ is set to the 1-state during a read operation to permit the parity of a word read from Memory to be checked. Flip-flop SP2 is employed to remember parity during right shift, multiply and implode operations. Flip-flops SP1 and SP2 are employed to remember parity during left shift and divide operations.

Arithmetic Unit

The Arithmetic Unit contains the logical elements necessary to execute binary and decimal arithmetic instructions and bit and character shifts. The Arithmetic Unit comprises the following registers, shown in FIG. 20: M-Register **301**, E-Register **303**, CC-Register **304**, N-Register **305** and Q-Register **306**. The Arithmetic Unit also includes the Accumulator in Memory and Adder **302** which comprises two ranks of majority logic adder circuits and decimal carry and correction logic. The details of the circuit structure of the Arithmetic Unit are provided in FIG. 25, which illustrates the Adder; FIGS. 52–63, 68 and 71–74, which are the flip-flop input logical schematic diagrams of Arithmetic Unit flip-flops employed in registers and counters; and FIGS. 115–117, which are full adder input signal diagrams.

M-Register **301** is a 24-bit register, comprising flip-flops M00–M23, which serves as an operating register during instruction execution and input-output operations. M-Register **301** receives signals WD00–WD23, which represent the 24 bits of a word, from the Data Drivers of Memory during a read operation. Flip-flops M00–M05 of M-Register **301** are adapted to receive, by parallel transfer, the bit position 0–5 output signals of Adder **302** upon application of working clock signal QSMA to the input gates of M-Register **301** which connect the appropriate adder output terminals to the corresponding input terminals of the M-register flip-flops.

M-Register flip-flops M06–M14 are adapted to receive, by parallel transfer, the bit position 6–14 output signals of Adder **302** upon application of working clock signal QSMB to the input gates of the M-Register which connect the appropriate adder input terminals to the corresponding input terminals of the M-Register flip-flops. M-Register flip-flops M15–M23 are adapted to receive, by parallel transfer, the bit position 15–23 output signals of Adder **302** upon application of working clock signal QSMC to the input gates which connect the appropriate output terminals of Adder **302** to the corresponding input terminals of the M-Register flip-flops. During an input/output operation, the M-register flip-flops are adapted to receive information from the Input/Output Control Unit **13** upon application of signal DGMM to the input gates which connect Input/Output Control Unit **13** to the corresponding input terminals in the M-Register flip-flops. Signals DCM0–DCM3 are applied to the input terminals of the M-Register flip-flops at predetermined times to reset the flip-flops of M-Register **301** to the 0-state, clearing the M-Register. Working clock signal QSHM causes the contents of flip-flops M23–M01 to be transferred to the next lower-order flip-flop of the M-Register. The contents of flip-flop M00 may be transferred to flip-flop M23. The information contained in M-Register **301** may be transferred to Memory, Input/Output Control Unit **13**, Parity Adder **472**, Q-Register **306**, B-Gates **317** and I-Register **313**. The logical schematic diagrams of M-Register **301** are given by FIGS. 58–63.

Adder **302** provides logic for binary and decimal arithmetic operations during instruction execution and also serves as a bus for most data transfers within the Central Processor. Adder **302** comprises two ranks of full adder circuits. First-rank full adders S00–S23 perform the arithmetic and data transfer operations. Second-rank full adders S51–S53, S57–S59, S63–S65 and S69–S71 correct the result obtained from the first-rank adders during a decimal arithmetic operation. Adder **302** does not provide static storage in the Central Processor. Information from M-Register **301** is applied to the first-rank adders upon application of signal FMSX to the input gates connecting the output terminals of the M-Register flip-flops to corresponding input terminals of the Adder. Application of information in E-Register **303**, D-Register **311**, P-Register **310** and CC-Register **304** to the first-rank adders is similarly effected by gating signals DEST, DDST, DPST, FCS2 and FCS3 applied to the adder input gates. Signal FCAY serves to add one during an arithmetic operation to incorporate a carry from a previous addition in the sum, or to add one to a complement applied to the first-rank full adders. The output signals of Adder **302** may be transmitted to M-Register **301**, CC-Register **304**, P-Register **310**, D-Register **311** and E-Register **303**. The schematic diagram of Adder **302** is illustrated in FIG. 25. The logical schematic diagrams of the adder inputs are given in FIGS. 115 and 116.

E-Register **303** is a 24-bit register which serves as an operating register during instruction execution. E-Register **303** comprises flip-flops E00–E23. The E-Register flip-flops are adapted to receive, by parallel transfer, the output signals of Adder **302** upon application of working clock signal QSEX to the input gates which connect the adder output terminals to the corresponding input termi-

nals of the E-Register flip-flops. E-Register flip-flops E00 and E01 are adapted to receive, by parallel transfer, the contents of the corresponding flip-flops of Accumulator Length Indicator Register **314** and E-Register flip-flops E02–E14 are adapted to receive, by parallel transfer, the contents of the corresponding flip-flops of A-Register **312**, upon application of signal DAEX to the input gates which connect the output terminals of Accumulator Length Indicator Register **314** and A-Register **312** to the corresponding input terminals of the E-Register flip-flops. Similarly, flip-flops E15–E23 are adapted to receive, by parallel transfer, the contents of flip-flops E00–E08, upon application of working clock signal QEEX to the input gates which connect the output terminals of flip-flops E00–E08 to the corresponding input terminals of flip-flops E15–E23. Working clock signal QCER causes the complement of the contents of E-Register **303** to be formed in the E-Register. Working clock signal QSHE causes the contents of flip-flops E23–E01 to be transferred to the next lower-order flip-flop of the E-Register. The contents of flip-flop E00 is transferred out of the E-Register and may, during the execution of certain instructions, be transferred to flip-flop M23 of M-Register **301**. The contents of the E-Register may be transmitted to A-Register **312**, Accumulator Length Indicator Register **314** and Adder **302**. The logical schematic diagrams of E-Register **303** are given in FIGS. 52–57.

CC-Register **304** is a five-bit register comprising flip-flops CC1–CC4 and is employed primarily as a counter. CC-Register **304** counts carries during the execution of Instruction 27, Variable Length Multiply (VLM) and counts subtractions during execution of Instruction 26, Variable Length Divide (VLD). CC-Register **304** counts candidates for suppression during execution of Instruction 05, Edit (EDT) and also stores the number of character or bit positions through which the Accumulator is to be shifted during execution of Instruction 22, Shift (SHG). The CC-Register flip-flops are adapted to receive, by parallel transfer, the contents of the corresponding flip-flops of N-Register **305** upon application of working clock signal QNCX to the input gates which connect the output terminals of the N-Register flip-flops to corresponding input terminals of the CC-Register flip-flops. The CC-Register flip-flops are also adapted to receive the bit position 0–4 adder output signals upon application of working clock signal QSCX to the input gates which connect the appropriate adder output terminals to the corresponding input terminals of the CC-Register flip-flops. CC-Register **304** is cleared by signal DCLC, the count in the CC-Register is advanced by one in response to signal DCCU. The contents of CC-Register **304** may be transmitted to Adder **302** or to N-Register **305**. The logical schematic diagrams of CC-Register **304** are given in FIGS. 71 and 72.

N-Register **305**, comprising flip-flops N00–N04, is a five-bit register which serves as a counter during execution of Instructions 05, Edit (EDT), 22, Shift (SHG), 27, Variable Length Multiply (VLM) and 26, Variable Length Divide (VLD). The N-Register flip-flops are adapted to receive, by parallel transfer, the contents of the corresponding flip-flops of CC-Register **304** upon application of working clock signal QCNX to the input gates which connect the output terminals of the CC-Register flip-flops to corresponding input terminals of the N-Register flip-flops. N-Register flip-flops N00–N03 are also adapted to receive, by parallel transfer, the contents of the corresponding flip-flops of Q-Register **306** upon application of working clock signal QQNX to the input gates which connect the output terminals of the Q-Register flip-flops to the corresponding input terminals of the N-Register flip-flops. The count in N-Register **305** is reduced by one in response to working clock signal QNCD. The contents of the N-Register may be transmitted to CC-Register **304** or to Q-Register **306**. The logical schematic diagrams of N-Register **305** are given in FIGS. 73 and 74.

Q-Register **306**, comprising flip-flops Q00–Q03, is a four-bit register employed as a counter during the shift operations of certain instructions. Q-Register **306** is also used to store the multiplier character during execution of Instruction 27, Variable Length Multiply (VLM), or the trial quotient character during execution of Instruction 26, Variable Length Divide (VLD). It also serves to store the address control field of the instruction word. The Q-Register flip-flops are adapted to receive, by parallel transfer, the contents of the corresponding flip-flops of the N-Register upon application of signal DWT9 to the input gates which connect the output terminals of the N-Register flip-flops to corresponding input terminals of the Q-Register flip-flops. Q-Register flip-flops Q00–Q02 are also adapted to receive, by parallel transfer, the contents of flip-flops M15–M17 of the M-Register upon application of signal DMQX to the input gates which connect the output terminals of flip-flops M15–M17 to the corresponding input terminals of flip-flops Q00–Q02. The flip-flops of Q-Register **306** are reset to the 0-state upon application of signal DCLQ to their respective input terminals, clearing the Q-Register. The contents of Q-Register **306** may be transmitted to N-Register **305** and B-Gates **317**. The logical schematic diagrams of Q-Register **306** are given in FIG. 68.

RELOCATABLE ACCUMULATOR

The Accumulator of the Central Processor comprises four adjacent locations in Memory. The Accumulator locations are assigned by the program and the Accumulator may be relocated in any group of four adjacent memory locations. Relocating the Accumulator does not affect the contents of the memory locations involved. The apparatus for varying the location of the accumulator described herein includes the invention of Messrs. Robert D. Hunter, David E. Keefer, and John E. Wilhite. The four words of the Accumulator are directly addressable, as well as being implicitly addressed during the execution of various instructions.

In addition to relocating the Accumulator in Memory, the working or effective length of the Accumulator can be set to single, double, triple or quadruple word length. The means for varying the length of the accumulator described herein includes the invention of Messrs. Richard A. Boennighausen and Byron F. Burch, Jr. Regardless of the working length assigned the Accumulator, the full Accumulator always corresponds to four adjacent memory locations, i.e. has a length of four words. The setting of the working length of the Accumulator determines the length of the memory field affected by certain operations involving the Accumulator. Instructions are provided for setting the working length alone or in combination with other operations.

The four words of the full Accumulator, designated words D, C, B and A, are illustrated diagrammatically in FIG. 24. Word D is the most-significant word of the Accumulator and word A is the least-significant word of the Accumulator with words C and B being assigned successively decreasing orders of significance in accordance with their positions within the Accumulator. The address of the full Accumulator is the address of the most-significant word, i.e. word D, of the Accumulator. The address of the most-significant word of the Accumulator is chosen, in assigning the location of the Accumulator in Memory, to be evenly divisible by four, i.e. 0, modulo-4. This address is then stored in the 13-bit A-Register **312**, the two least-significant bits of the address being assumed to be zero. Thus, the location of word D and the address of the full Accumulator can be arbitrarily designated as *xxx xxx xxx xxx x*00, the *x*'s representing the bits of the address in the A-Register **312**; the location of the next most-significant word, word C, is *xxx xxx xxx xxx x*01 while the location of words B and A of the full Accumulator are *xxx xxx xxx xxx x*10 and *xxx xxx xxx xxx x*11, respectively, as illustrated in FIG. 24.

The working length of the Accumulator is established by the program. Regardless of the working length, the least-significant word of the Accumulator is always word A and the actual length of the full Accumulator is always four words. As illustrated in FIG. 24, if the accumulator length is single, the working Accumulator comprises word A, if the accumulator length is double, the working Accumulator comprises words A and B, if the accumulator length is triple, the working Accumulator comprises words A, B and C, and if the accumulator working length is quadruple, the working Accumulator comprises words A, B, C and D. The working length of the Accumulator is stored in Accumulator Length Indicator Register 314 which comprises flip-flops PL2 and PL1. The contents of the Working Accumulator Indicator Register 314 establish the working length of the Accumulator as shown in the following table:

ACCUMULATOR LENGTH INDICATOR REGISTER AND ACCUMULATOR WORKING LENGTH

PL2, PL1	Accumulator Working Length
00	quadruple
01	triple
10	double
11	single

As seen in FIG. 24, the combined contents of A-Register 312 and Accumulator Length Indicator Register 314 can be interpreted as specifying the location of the most-significant word of the working Accumulator.

Instruction 36, Load Accumulator Location and Length (LAL), establishes the working length of the Accumulator as well as the location of the Accumulator in Memory. Instruction 22, Shift (SHG), may establish the working length of the Accumulator, in addition to performing the shifting operation. Instruction 37, Store Accumulator Location and Length (SAL), provides for storing into any specified memory location signals representing the current working length of the Accumulator as well as the current accumulator location.

The following instructions each set the working length of the Accumulator to the length specified by the operation code of the instruction:

Instruction 40, Load Single (LDS),
Instruction 41, Load Double (LDD),
Instruction 42, Load Triple (LDT),
Instruction 43, Load Quadruple (LDQ).

The following instructions cause the working length of the Accumulator to be changed to the length specified in the operation code if the previously established working length is less than that specified by the operation code:

Instruction 50, Add Decimal Single (ADS),
Instruction 51, Add Decimal Double (ADD),
Instruction 52, Add Decimal Triple (ADT),
Instruction 53, Add Decimal Quadruple (ADQ),
Instruction 60, Subtract Decimal Single (SDS),
Instruction 61, Subtract Decimal Double (SDD),
Instruction 62, Subtract Decimal Triple (SDT),
Instruction 63, Subtract Decimal Quadruple (SDQ).

In the above instructions, the operation code specifies the length of the memory field, while the previously established accumulator length determines the length of the accumulator field. If the length specified by the operation code is greater than the previously established working length, the working length of the Accumulator is changed to the length specified by the operation code and the accumulator words added to the working Accumulator are set to zero before the arithmetic operation is performed.

In executing the above eight instructions, if the field length specified by the operation code is equal to or less than the working length of the Accumulator, the addition or subtraction of the specified memory field takes place without changing the accumulator working length. Carries propagate across the accumulator words, with any carry out of the existing accumulator length causing Overflow flip-flop FVO to be set to the 1-state. If the field length specified by the operation code is greater than the accumulator working length, the accumulator working length is extended to coincide with the length specified by the operation code. The accumulator words added to the working Accumulator are cleared to 0 before the addition or subtraction takes place. Any arithmetic carry out of the newly established working Accumulator causes Overflow flip-flop FVO to be set to the 1-state.

The following instructions never cause the working length of the Accumulator to be changed:

Instruction 54, Add to Memory Single (AMS),
Instruction 55, Add to Memory Double (AMD),
Instruction 56, Add to Memory Triple (ADT),
Instruction 57, Add to Memory Quadruple (AMQ).

In these instructions, the operation code specifies the length of the memory field while the previously established accumulator length determines the length of the working accumulator. In executing these instructions, if the accumulator length is equal to or less than the field length specified by the operation code, the addition to the memory field takes place, with carries propagating across the memory field. A carry out of the high-order memory field causes Overflow flip-flop FVO to be set to the 1-state. If the working accumulator length is greater than the field length specified by the operation code, the addition does not take place and the Overflow condition automatically results.

The instructions referred to above contain a designation of accumulator working length. Certain other instructions contain no designation of accumulator working length but depend upon the previously established accumulator working length to delimit the operation. The instructions in this category are:

Instruction 02, Compare Decimal Accumulator to Memory (CDA),
Instruction 03, Compare Alphanumeric Accumulator to Memory (CAA),
Instruction 05, Edit (EDT),
Instruction 20, Explode (EXP),
Instruction 21, Implode (IMP),
Instruction 15, Branch if Zero (BRZ).

These instructions never have any effect on the existing accumulator working length. The manner in which each instruction is affected by the previously established accumulator length is described in the portion of the specification containing descriptions of these instructions. In general, the accumulator working length defines how many accumulator words are involved in executing the instruction and also defines how many memory field words, if any, are involved.

ADDER

Adder 302 includes two ranks of majority logic adders, as illustrated in FIG. 25. The first rank, comprising full adder circuits S00–S23, serves to add the contents of M-Register 301 to the contents of either E-Register 303, CC-Register 304, P-Register 310 or D-Register 311. The second rank comprises adder circuits S51–S53, S57–S59, S63–S65 and S69–S71. Each group of six first-rank adder circuits has associated with it a group of three second-rank adder circuits which effect corrections during decimal operations. Second-rank adder circuits S51, S52 and S53 perform decimal correction on the character 3 sum produced by adder circuits S00–S03 and provide the second, third and fourth binary digits of the corrected character 3 sum. Similarly, second-rank adder circuits S57–S59, S63–S65 and S69–S71 perform decimal correction on

the character 2, character 1 and character 0 sums produced by adder circuits S06–S09, S12–S15 and S18–S21, respectively.

Adder **302** is also employed as a bus for data transfers in the Central Processor. For example, a word in M-Register **301** may be transferred to E-Register **303** through Adder **302**, Adder **302** performing no operation on the word being transferred. Adder **302** does not provide static storage in the Central Processor.

FIGURE 25 illustrates schematically the circuit arrangement of Adder **302**. Referring to FIG. 25, each of the first-rank adder circuits S00–S23 has three inputs, termed the X-, Y- and Z-inputs. The Y-inputs to adder circuits S00–S23 are derived from E-Register **303**, CC-Register **304**, P-Register **310** and D-Register **311**. The X-inputs to first-rank adder circuits S00–S23 are provided by M-Register **301**. In addition, input terms for the Edit instruction are provided in the X-inputs for adder circuits S00–S05 and special terms for move, branch-on counter and input/output operations are included in the X-inputs to adder circuits S15 and S16 and in the Y-inputs for adder circuits S15 and S17. The X-inputs to adder circuits S04, S05, S10, S11, S16, S17, S22 and S23 are disabled during a decimal operation. The Z-inputs of first-rank adder circuits S01–S23 include carry inputs from the previous stage. In addition, the Z-inputs of adder circuits S06, S12 and S18 include carry inputs generated during decimal operations. A Z-input to adder circuit S00 is provided for carry and input/output operations. The full adder input signal logical schematic diagrams are illustrated in FIGS. 115 and 116. The Sum (S) output signals of first-rank adder circuits S00–S23 are identified as MS00–MS23, respectively, while the Carry (C) output signals are identified as MC00–MC23, respectively.

The second-rank adder circuits also have X-, Y- and Z-inputs. The X-inputs for the second-rank adder circuits are provided by the Sum (S) outputs of corresponding adder circuits in the first rank. For example, the X-inputs for adder circuits S51, S52 and S53 are MS01, MS02 and MS03, respectively. The Y- and Z-inputs for the second-rank adder circuits are provided by carries and by signals, $\overline{DAD0}$–$\overline{DAD3}$ and $\overline{DSD0}$–$\overline{DSD3}$ which determine whether or not decimal correction of the character sums is to be effected. The Sum (S) output signals of second-rank adder circuits are identified as MS51–MS53, MS57–MS59, MS63–MS65 and MS69–MS71. The carry output signal descriptions employ the letter "C" in place of "S."

The input gates to adder circuits S00–S23 of the first rank provide a binary 0 signal for a binary 1 input condition. Adder **302** thus sums binary 0's. To compensate for this, the output signals from Adder **302** are taken from the $\overline{Sum}$ ($\overline{S}$) output terminals of the adder circuits. For example, flip-flop E00 of E-Register **303** is set to the 1-state in response to output signal $\overline{MS00}$ of Adder **302**. Similarly, flip-flop E01 of E-Register **303** is set to the 1-state in response to output signal $\overline{MS01}$ of Adder **302**. A carry occurs if the $\overline{Carry}$ ($\overline{C}$) output signal of an adder circuit is a binary 1.

The control signal sources for Adder **302** include flip-flops DCE (FIG. 80), CAY (FIG. 79), CRE (FIG. 79), SIN (FIG. 90), LSN (FIG. 83), FVO (FIG. 81) and COR (FIG. 79). Decimal flip-flop DCE is set to the 1-state during a decimal operation. Carry flip-flop CAY is set to the 1-state to increment an address, to add one to a number to produce the 2's complement after the 1's complement has been produced by complementing the contents of the E-Register **303** and to add one to an Accumulator if the previous Accumulator produced a carry during a multiple accumulator operation. Carry flip-flop CAY provides the Z-input to adder circuit S00. Carry Remember flip-flop CRE is used during multiple accumulator operations and is set to the 1-state by signal DDC0 to store a carry out of the most-significant character of Adder **302** until it can be added to the next Accumulator. Flip-flop CRE is also used during other instructions to temporarily hold a carry from Adder **302** until appropriate action can be taken.

Sign flip-flop SIN stores the sign of the contents of the M-Register and is set to the 1-state when the contents of M-Register **301** are minus. Like Sign flip-flop LSN is set to the 1-state if the sign of the E-Register contents is the same as the sign of the M-Register contents, i.e. the contents of M-Register **301** and E-Register **303** are both plus or both minus. Flip-flop LSN is reset to the 0-state if the signs of the contents of the E-Register and the M-Register are different.

Overflow flip-flop FVO is set to the 1-state in response to a carry out of the most-significant character position of Adder **302** during the last accumulator operation of a like-signs addition. Decimal Correct flip-flop COR is set to the 1-state during an unlike-signs addition if there is no carry ($\overline{DDC0}$) from the most-significant character position during the last accumulator operation (DLAO) to indicate that the entire sum must be corrected by taking the 10's complement of the result.

The sign of the result of a decimal addition or subtraction is controlled by NAND-gate **600** and NAND-gate **601** which provide signals $\overline{DS55}$ and $\overline{DS54}$, respectively. $\overline{DS54}$ is always a binary 0 during a decimal operation in Adder **302**. Signal $\overline{DS55}$ is a binary 1 during a decimal operation to make the sign of the result minus if Decimal Correct flip-flop COR has been set to the 1-state and Sign flip-flop SIN is reset to the 0-state. Signal $\overline{DS55}$ is also a binary 1 if Decimal Correct flip-flop COR is reset to the 0-state and Sign flip-flop SIN is set to the 1-state. The following table illustrates the relationship between the sign of the M-Register contents, the state of flip-flop COR and the sign of the result for a decimal addition of numbers having unlike-signs:

SIGN DETERMINATION

Sign of M-Register Contents	FSIN	FCOR	$\overline{DS55}$	$\overline{DS54}$	Sign of Result
minus	1	0	1	0	minus
minus	1	1	0	0	plus
plus	0	0	0	0	plus
plus	0	1	1	0	minus

In operation, the contents of M-Register **301** are added, on a bit by bit basis, to the contents of one of the registers providing the Y-inputs to the first-rank full adders. Carriers are transmitted from the adder circuits adding the least-significant binary digits progressively forward to the adder circuits adding the most-significant binary digits, i.e. from full adder S00 toward full adder S23. During a decimal addition, the adder inputs comprising zone bit information to adder circuits S04, S05, S10, S11, S16, S17, S22 and S23 are disabled. If a binary addition is performed, signals DDUS and DDLS are binary 0's to prevent decimal correction of the sum in the second-rank full adders. The Z-input to adder circuit S00 is high during a normal addition with no carry from a previous addition and does not affect the result since Adder **302** sums binary 0's. The Z-input signal to adder circuit S00 becomes a binary 0 during an input/output operation to increment an address and during certain arithmetic operations to add one to a number.

Addition in Adder **302** can be either like-signs addition or unlike-signs addition. Like-signs addition is defined as an add operation with the signs of the operand words the same, or a subtract operation with the signs of the operand words different. Unlike-signs addition is defined as an add operation with the signs of the operand words different or a subtract operation with the signs of the operand words the same. During a like-signs addition, a correction of six is added to each character of the result which is greater

than nine. In an unlike-signs addition, during which the absolute decimal magnitude of one operand word is added to the absolute magnitude of the 10's complement of the other operand word, decimal correction is effected by subtracting six from each character of the result which is not greater than nine.

During a like-signs addition, if the result of an addition of two characters is in the range 0–9 inclusive, no correction is necessary and the corresponding second-rank adder circuits do not modify the result obtained by the first-rank adder circuits. If the result lies in the range of 10 or over, the appropriate signal DAD0, DAD1, DAD2 or DAD3 is a binary 0 to effect addition of six to the character result obtained by the first-rank adder circuits. For example, if the sum of the characters added in first rank adder circuits S00–S05 is greater than nine, the signal $\overline{\text{DAD3}}$, which is the Y-input signal to adder circuits S51 and S52, is a binary 0, effecting an addition of six to the sum produced by first-rank adder circuits S00–S05.

During an unlike-signs addition, flip-flop LSN is reset to the 0-state. Working clock signal QCER issues to complement the contents of E-Register **303**, forming the 1's complement of the contents of E-Register **303**. The Z-input to first-rank adder circuit S00 becomes a binary 0 to add one to the contents of the E-Register, forming the 2's complement of the contents of E-Register **303**. If a carry is generated in a character result during an unlike-signs decimal addition, the character sum is correct and no correction is performed by the corresponding second-rank adder circuits. If no carry is generated in the result produced by the addition of two characters, the appropriate signal DSB0, DSB1, DSB2 or DSB3 is a binary 0 to effect the subtraction of six from the character result obtained by the first-rank adder circuits. For example, if the sum of the characters added in first-rank adder circuits S00–S05 is less than nine, signal $\overline{\text{DSB3}}$ is a binary 0. Signal $\overline{\text{DSB3}}$, which is the Y-input for second-rank adder circuit S53 and the Z-input for second-rank adder circuit S51, effects the subtraction of six from the sum produced by the first-rank adder circuits to provide the correct decimal sum.

The details of the operation of Adder **302** are provided in the descriptions of the various arithmetic instructions which employ Adder **302**, such as in the section entitled "Instruction 60: Subtract Decimal Single (SDS)."

Console

The Console provides an indicating and control station for the operator whereby he has access to the system for control and observation of operation. The Console enables the operator to manually control the clearing of the system, the initiation of execution of instructions, interruption of program execution, program loading and the halting of system operation. FIGURE 26 illustrates diagrammatically the switches of the Console and the signals which are provided to the system upon actuation of the switches. Not shown in FIG. 26 are indicators which may be employed to show the states of the various register, alarm and control flip-flops of the system by indicating with illuminated lamps the flip-flops which are in their 1-states. Also not shown in FIG. 26 are a plurality of switches which may be employed to control typewriter operation and others which may be employed in diagnosing malfunctions in the system.

CLEAR CONTROLS

Selected flip-flops of the data processing system may be cleared by closing appropriate switches on the Console. Initial actuation of the power supplies and application of power to the data processing system is effected by moving the contacts of switch **420** of FIG. 26 to the Power On position. Signal SPSD on line **421** is thus provided to the data processing system to generate Initial Clear signal DICR and to set the System Reset flip-flop SRE (FIG. 90) to the 1-state. Initial Clear signal DICR resets Start-Stop Clock Logic flip-flops RCK (FIG. 83) and ISP (FIG. 83) to the 0-state and also causes the flip-flops of the B-Register and the CC-Register and certain other control flip-flops of the Memory to be reset to the 0-state. The System Reset flip-flop SRE controls the state of other flip-flops in the system to ready the Central Processor for operation.

When the Power On switch **420** of the Console is closed and the data processing system is in the manual mode of operation, i.e. halted, closure of the Reset Computer switch **424** provides signal SMRS on line **425** to the data processing system to cause certain flip-flops of the system to assume predetermined states. Signal SMRS sets Manual Reset flip-flop MRE (FIG. 85) to the 1-state. Signal DMLR issues to preset the TL-Counter to TL2 and to reset certain flip-flops of the Program Processor to the 0-state. After actuation of the Reset Computer switch **424**, the W-Selector Register **470** of the Program Processor assumes the W10 state and the system enters an idling cycle until further instruction execution is initiated by the operator. Actuation of switch **424** when the system is in the run mode has no effect.

Actuation of Memory Parity Alert switch **426** on the Console, when the system is in the manual mode of operation, provides signal SMPR on line **427** to the Program Processor. Signal SMPR resets Word Parity Error flip-flop WJE (FIG. 92) to the 0-state. Flip-flop WJE is set to the 1-state in response to the detection of a parity error upon extraction of a word from Memory.

The closure of Instruction Alert switch **430**, when the system is in the manual mode, provides signal SPES on line **431**. Signal SPES resets to the 0-state Program Error flip-flop PEF (FIG. 87) which had previously been set to the 1-state in response to a program error. Similarly, actuntion of Control Word Alert switch **434** causes signal SIWR to appear on line **435**. Signal SIWR resets Invalid Control Word flip-flop ICW to the 0-state. Flip-flop ICW is set to the 1-state in response to the detection of an invalid control word in the program.

START CONTROLS

Continuous sequential processing of instructions is initiated, when the system is in the manual mode, by actuation of Run switch **438** to provide to the Program Processor signal SSSW on line **439**. Signal SSSW causes Start flip-flop STT (FIG. 91) to be set to the 1-state. The 1-output signal of flip-flop STT in turn sets Start Synchronizing flip-flop SSS (FIG. 91) to the 1-state. The 1-output signal of flip-flop SSS resets Manual flip-flop MAN (FIG. 84) to the 0-state and permits the W-Selector Register to change state and control further instruction execution. Actuation of Run switch **438** when the system is in the run mode has no effect.

Closure of Single Instruction switch **440** provides signal SSTS to the Program Processor on line **441**. Signal SSTS sets Step flip-flop STE (FIG. 91) to the 1-state. The 1-output signal of flip-flop STE in turn sets Start Synchronizing flip-flop SSS (FIG. 91) and Stop flip-flop STP (FIG. 91) to the 1-state. The 1-output signal of flip-flop SSS resets Manual flip-flop MAN to the 0-state to permit execution of an instruction to proceed. When the instruction has been executed, signal DEIN issues. Signal DEIN is employed along with the 1-output signal of flip-flop STP to provide a trigger signal setting flip-flop MAN to the 1-state, thereby preventing further instruction execution. Thus, when Single Instruction switch **440** is actuated while the system is in the manual mode of operation, Manual flip-flop MAN is reset to the 0-state to permit execution of one instruction and then set to the 1-state to halt instruction processing. Actuation of Single Instruction switch **440** when the system is in the run mode will cause Manual flip-flop MAN to be reset to the 0-state and instruction processing to be halted at the end of the instruction being processed.

HALT CONTROLS

Termination of program processing is effected by closure of Manual switch **444** to provide signal SSPS

on line 445. Signal SSPS sets Stop flip-flop STP (FIG. 91) to the 1-state. The 1-output signal of flip-flop STP sets Manual flip-flop MAN (FIG. 84) to the 1-state upon completion of execution of the instruction being executed when the Manual switch 444 is actuated. The 1-output signal of flip-flop MAN presets the W-Selector Register flip-flops to W10 at the end of a W00 block to terminate instruction processing. Actuation of Manual switch 444 when the system is in the manual mode has no effect.

PROGRAM INTERRUPT CONTROLS

A request for a program interrupt by the processor channel can be caused by closure of Request Control switch 448 to provide signal SPIS on line 449. Signal SPIS sets Program Interrupt Manual flip-flop PIM (FIG. 88) to the 1-state. Control flip-flop PMI (FIG. 88) is in the 0-state. Manual Program Interrupt flip-flop MPI (FIG. 85) is set in response to the 1-output signal of flip-flop PIM and the 0-output signal of flip-flop PMI. If permitted by Program Interrupt Termination and Inhibit flip-flop PIT (FIG. 88), the 1-output signal of flip-flop MPI causes a program interrupt to the processor channel. Actuation of Request Control switch 448 when the system is in the run mode has the same effect as when the system is in the manual mode.

Actuation of Program Interrupt Reset switch 450, when the system is in the manual mode, causes signal SPIR to be applied to the Program Processor on line 451 to reset flip-flop PIT. Actuation of switch 450 when the system is in the run mode has no effect.

PROGRAM LOADING CONTROLS

Load switch 454, effective only in the manual mode, provides signal SLPS to the Program Processor on line 455 to establish the load mode whereby a program is loaded in Memory from a selected peripheral subsystem. Load Peripheral Manual flip-flop LPM is set to the 1-state in response to the SLPS signal. The 1-output signal of flip-flop LPM causes signal DLPT to issue. The DLPT signal, in conjunction with the Load Channel switch, causes the selected peripheral subsystem to commence data interrupts. The 1-output signal of the LPM flip-flop also sets Data Control Word Manual and P-Register Manual flip-flops DCW (FIG. 80) and PRM (FIG. 89) respectively. The 0-output signal of Manual Transfer flip-flop MXS (FIG. 87), in conjunction with the 1-output signal of either flip-flop DCW or flip-flop PRM, sets flip-flop MNX (FIG. 84) to the 1-state. Flip-flops MNX, DCW and PRM cause the first four characters received from the selected peripheral subsystem to be inserted in the Data Control Word (DCW) fixed address location for the selected channel and in the P-Register. Upon completion of the transfer of the first word of data from the selected peripheral subsystem, flip-flop MNX is reset to the 0-state. The remainder of the program words are stored in the location specified by the Data Control Word for the selected channel. Termination of the loading operation is caused by the detection of physical end of record by the peripheral subsystem. The loaded program may now be processed by actuating the Single Instruction or Run switches.

Load Channel switch 458 receives signal DLPT, which issues upon actuation of Load switch 454, and, according to the position of switch 458, applies one of the signals SLS0–SLS7 to the switch output line corresponding to the selected channel. The Load Channel switch 459 is thus employed to select the peripheral subsystem to which the signal from the Load switch 454 is sent. The selected peripheral subsystem then loads the Memory of the Central Processor with a program which is to be executed by the data processing system.

INSTRUCTIONS

The Central Processor responds to 53 different basic commands or instruction words to execute 53 fundamentally different data processing operations. These operations may be executed solely within the Program Processor, by the Program Processor in cooperation with the Memory or by the Program Processor in cooperation with the Memory and the Input-Output Control Unit. Each data processing operation consists of a sequence of macro-operations controlled and directed by the Timing Clock Generator, the TL-Counter, the W-Selector Register or the R-Selector Register, and the I-Register contents. Each data processing operation is initiated by the operation code of an instruction word stored in the I-Register and is identified by the instruction word by which it is initiated. For example, the data processing operation executed when the contents of the I-Register are octal 40 is identified as "Instruction 40." In addition to the operation code, each instruction word contains an address field. The instruction address field identifies the location in Memory at which the operand is stored, is employed to develop the memory location of the operand or, in the case of certain instructions, contains other information necessary to instruction execution. The apparatus for controlling the sequence of operations performed to execute instructions described herein includes the invention of Messrs. Robert D. Hunter, Robert A. Perrine, and John E. Wilhite.

In the description to follow, the sequence of execution of each command or instruction word is described in detail. Each detailed description is accompanied by a timing diagram to aid in visualizing the steps in the operational sequence. The description of the 53 instruction words comprise a substantially complete functional description of the Central Processor operation and its control of and operation with the Peripheral Subsystems. The order of presentation of the instruction word description has been chosen to provide an orderly and progressive picture of the Central Processor operation.

Initial routine during data processing operations

Prior to execution of any instruction word, predetermined signals normally occur in the Central Processor to cause particular micro-operations to be performed. These predetermined signals effect the transfer of a new instruction word from Memory to the M-Register, the parity check of this instruction word, the transfer of the operation code of the instruction word to the I-Register, the transfer of the address control field (ACF) of the instruction word to the Q-Register, the transfer of the address field of the instruction word to the D-Register, the transfer of the instruction word to the E-Register, and the advancement of the count in the P-Register. These signals and the corresponding micro-operations occur during the W00 block to perform the macro-operation of "instruction fetch."

Other predetermined signals and micro-operations occur as part of the initial routine prior to instruction execution, if the address field of the instruction word is to be developed into a final address. These signals and micro-operations comprise the W01 block. The micro-operations of the W01 block extract from Memory and employ auxiliary words to develop a final operand address which is used during instruction execution. If the instruction is a two-address instruction, the micro-operations of the W01 block similarly develop the second address. A detailed description of the initial routine comprising the W00 and W01 blocks of micro-operations is given in the following sections.

W00 BLOCK OF MICRO-OPERATIONS—INSTRUCTION FETCH

The initial routine which is common to the execution of all instructions in the data processing system includes the W00 block of micro-operations. The W00 block performs the macro-operation of "instruction fetch." Dur-

ing the W00 block of micro-operations, the instruction word which is to control the next data processing operation is read from the memory location designated by the address in the P-Register. During a program interrupt, the memory location from which information is obtained in the W00 block is determined by the states of the program interrupt scanner flip-flops PS0–PS2. The parity of the instruction word is checked. The operation code of the instruction word is stored in the I-Register and the address field of the instruction word is stored in the D-Register preparatory to instruction execution. The address control field (ACF) of the instruction word is transferred to the Q-Register. The instruction word is restored to its memory location. The count in the P-Register is advanced by one to form the address of the next instruction or the next P-sequence word, if the address field of the present instruction is to be developed, or if the instruction is a two-address instruction. At the end of the W00 block, the Program Processor performs the other blocks of micro-operations necessary to the execution of the particular instruction unless Manual flip-flop MAN is set to the 1-state. In this event, the Program Processor goes to block W10S and instruction execution terminates.

The sequence of occurrence of the micro-operations of the W00 block is illustrated in the timing diagram of FIG. 119. In the timing diagrams to be described, the symbol Δ interposed between signal designations indicates that, if the conditions to the left of the symbol are satisfied, the signal or signals to the right of the symbol issue. At the beginning of the W00 block of micro-operations during time period TL0, signal DDBX issues. Signal DDBX is a binary 1 during the entire W00 block and serves to enable the B-Gates so that the address of the next instruction is transferred from the D-Register to the B-Register of Memory, as soon as this address has been transmitted to the D-Register. When working clock signal QTCK issues during time period TL0, flip-flop MRD is set to the 1-state to initiate a read operation in Memory. The operation of the Memory in response to the Memory Read command signal FMRD is described in the section "Memory—General Operation and Timing." Working clock signal QSDX also issues at this time to transfer the address of the next instruction from the P-Register through the Adder to the D-Register.

During time period TL1, the M-, Q- and CC-Registers are cleared. Upon issuance of working clock signal QTCK during time period TL1, Like-Signs flip-flop LSN is set to the 1-state under the assumption that the operand and the accumulator contents will have the same sign if an arithmetic operation is to be performed. Run Clock flip-flop RCK is reset to the 0-state at this time to stop generation of clock pulses by the Program Processor Clock Generator. The TL-Counter advances to TL2 and the Program Processor operation stops to wait for synchronizing signal DMDA from Memory. This sequence of micro-operations comprises the first half-sequence of the W00 block. As indicated in FIG. 119, certain flip-flops are also reset to the 0-state during this first half-sequence preliminary to execution of the instruction.

The second half-sequence of the W00 block of micro-operations is initiated when synchronizing signal DMDA from Memory issues to start the Program Processor Clock Generator. At this time, the instruction word is transferred from the Memory Data Drivers to M-Register 301. When working clock signal QTCK issues during time period TL2, the Memory Write command signal FMWR issues to initiate a restore operation in Memory to restore the instruction word to its former location. The operation of the Memory in response to the command signal FMWR is described in the section "Memory—General Operation and Timing." At this time flip-flop MSX is set to the 1-state to apply the instruction word to the X-inputs of the Adder. Working clock signal QMIX also issues to transfer the operation code of the instruction word to the flip-flops of the I-Register. Signal DAMS issues if the ACF of the instruction word has a value in the range of 1–7 while signal DNWX issues if the ACF of the instruction word equals 7.

During time period TL3 of the second half-sequence, signal DPCU issues to advance the count in the P-Register by one. The contents of the P-Register then represent the address of the next P-sequence word, which may be an instruction word, a P-Sequence AMS Word or a P-Sequence SAS Word. Signal DMQX also issues during time period TL3 to gate the address control field (ACF) of the instruction word from flip-flops M15, M16 and M17 of the M-Register to the Q-Register.

When working clock signal QTCK issues during time period TL4, working clock signals QSEX and QSDX issue to transfer the instruction word and the address field of the instruction word to the E-Register and the D-Register respectively. During time period TL5, flip-flop NXF is set to the 1-state if the operation code in the I-Register indicates that the instruction is an Instruction Group 3 instruction, i.e., a two-address instruction, and if the address control field (ACF) of the instruction word is not equal to seven. When working clock signal QTCK issues during time period TL5, flip-flop PSX is set to the 1-state to cause the address in the P-Register to be applied to the Y-input gates of the Adder if the address control field (ACF) of the instruction word is equal to seven or if the instruction word is a two-address instruction of Instruction Group 5. The address in the P-Register is then employed to obtain a P-Sequence AMS Word or a P-Sequence SAS Word from Memory to develop the first or second address respectively of the instruction word. At this time, Run Clock flip-flop RCK is reset to the 0-state to terminate generation of clock pulses in the Program Processor Clock Generator.

When control clock signal QTC0 issues during the next time period TL0, Indexing Sequence Register flip-flop XS1 is set to the 1-state if the address control field (ACF) of the instruction word, as sampled in the Q-Register, represents a value from 1–6, calling for the use of a Fixed Location Index Word (FLIW). If the instruction is a single-address instruction, the Fixed Location Index Word is used during the W01 block of micro-operations to modify the address field of the instruction word, whereas if the instruction is a two-address instruction, the address of the Fixed Location Index Word is employed as the second address. If the ACF of the instruction word is not equal to seven, flip-flop XS1 is reset to the 0-state. Control clock signal QTC0 also resets flip-flops AOI and XS2 as well as the flip-flops of the Accumulator Counter Register to the 0-state. The states of Indexing Sequence Register flip-flops XS2 and XS1 determine which sequence of the W01 block will be performed if address development is required. When synchronizing signal $\overline{\text{DMBU}}$ from Memory issues to start the Program Processor Clock Generator, flip-flop RCK is set to the 1-state when QTRK issues during time period TL0. The second half-sequence of the W00 block of micro-operations is then complete and the Program Processor is then ready to enter another block of micro-operations, defined by the W-Selector Register. In the above description, the micro-operations employed in the initial routine of all instructions have been noted. Micro-operations of the W00 block applicable only to particular instructions will be indicated in the detailed description of those instructions.

FIGURE 120 is a flow diagram of the W00 block illustrating the micro-operations which are performed in the W00 block prior to the execution of each instruction, as described above. The flow chart also illustrates the conditions under which address development is initiated in the W01 block or instruction execution is initiated, after performance of the W00 block is completed. Referring to FIG. 120, if the address control field of the instruction word, as represented by the states of flip-flops M17–M15 of the M-Register, equals 7, signal DNWX issues and sequence 0 of the W01 block of micro-operations is then

performed to develop a final address. If the address control field of the instruction word has a value in the range of 1–6, signal DFXW issues and sequence 1–DFXW of the W01 block of micro-operations is then performed.

If the address control field of the instruction word has a value of 0 and if signal $\overline{DIG3}$ has issued to indicate that the instruction is a single-address instruction, the address field of the instruction word is the final address of the operand and the W01 block is not performed. Rather, the blocks of micro-operation appropriate to execution of the particular instruction are performed. If the address control field of the instruction word has a value of 0 and if signal DIG3 has issued to indicate that the instruction is a two-address instruction, flip-flop NXF is set to the 1-state and sequence 0 of the W01 block is performed to develop the second address.

If the instruction is one of the instructions CMM, RIM, ANM, RXM, MFM, ABM or SBM, the W02 block of micro-operations is performed prior to developing the second address in sequence 0 of the W01 block. If the instruction is MOV, both the W15 and the W06R blocks of micro-operations are first performed before going to sequence 0 of the W01 block to develop the second address. A similar sequence of blocks occurs for the above-identified instructions prior to development of the second address in sequence 1–DFXW of the W01 block, if the ACF of the instruction word has a value in the range of 1–6.

W01 BLOCK OF MICRO-OPERATIONS—ADDRESS DEVELOPMENT

In address development, three kinds of address are employed: (a) tentative, (b) effective, and (c) final. A tentative address is an address that must be modified by addition of an index or indices thereto to develop an effective or final address. An effective address is one that is used to define the location of an Indirect Address Word in Memory. The final address is an address that is used with the operation code of an instruction word to execute the instruction. The apparatus described in this section includes the invention of Messrs. Thomas J. Beatson, Frank J. Boyle, Byron F. Burch, Jr., Robert D. Hunter, and Daniel W. Scott.

The W01 block of micro-operations is part of the initial routine for all instructions which require address development. Single-address instructions require address development if the address field of the instruction word is not a final address. All two-address instructions require development of the second address in the W01 block of micro-operations, and may also require development of the first address. During the W01 block of micro-operations, auxiliary words are extracted from Memory and used to develop final addresses.

The W01 block of micro-operations includes five address development sequences defined by the states of Indexing Sequence Register flip-flops XS2, XS1 and signal DFXW, as shown in the following table:

BLOCK W01 SEQUENCES

FXS2	FXS1	DFXW	Sequence
0	0	.....	0
0	1	1	1-DFXW
0	1	0	1-$\overline{DFXW}$
1	0	.....	2
1	1	.....	3

Either sequence 0 or sequence 1–DFXW of the W01 block may be entered after the instruction fetch macro-operation of the W00 block is performed. If a final address is not obtained from a performance of a sequence of the W01 block, the W01 block is again entered in the same or another sequence. Re-entry of the W01 block with performance of an appropriate sequence is repeated until the address field of the instruction word is developed into a final address or until the second address of certain two-address instruction is developed. The following table indicates the sequences of the W01 block which can be entered from a given sequence to continue address development:

BLOCK W01 SEQUENCE SUCCESSION

From Sequence	To Sequence
0	0, 1-$\overline{DFXW}$, 2.
1-DFXW	Address development completed.
1-$\overline{DFXW}$	0, 1-$\overline{DFXW}$, 2.
2	0, 3.
3	0, 1-$\overline{DFXW}$, 2.

Each of the sequences of the W01 block may result in the development of a final address. Therefore, the appropriate blocks W02–W12, W14 and W15 can be entered from any of the sequences of the W01 block to execute a given instruction. Signal DXIM issues to indicate that address development of a single-address instruction is complete and the blocks of micro-operations appropriate to execution of the particular instruction may be entered. Signal DXIM also issues to indicate that development of the first address of a two-address is complete and development of the second address may be initiated. Signal DVCA issues when development of the second address of a two-address instruction is complete to indicate that the blocks of micro-operations appropriate to execution of the particular instruction may be entered.

The sequence of occurrence of the micro-operations of the W01 block is illustrated in the timing diagram of FIG. 121. At the beginning of the W01 block of micro-operations during time period TL0, signal DDBX issues if one of the sequences 0, 1–DFXW, 2 or 3 is to be performed. Signal DDBX is a binary 1 during the entire W01 block and enables the B-Gates so that the address of the auxiliary word is transferred from the D-Register to the B-Register of Memory, as soon as this address appears in the D-Register. If sequence 1–DFXW is being performed, signal DQBX issues to enable the B-Gates so that the address of a Fixed Location Index Word is transferred from the Q-Register to the B-Register of Memory. During a program interrupt, the memory location from which information is obtained in the W01 block is determined by the state of the program interrupt scanner flip-flops PS0–PS2. When working clock signal QTCK issues during time period TL0, flip-flop MRD is set to the 1-state to initiate a read operation in Memory. The operation of the Memory in response to the Memory Read command signal FMRD is described in the section "Memory—General Operation and Timing." If either sequence 0 or sequence 3 is being performed, working clock signal QSDX also issues at this time to transfer the address of a P-Sequence AMS or SAS Word or the address of an Indirect AMS Word respectively from the Adder to the D-Register, for transfer to the B-Gates by signal DDBX.

During time period TL1, flip-flop DSX is reset to the 0-state and signals DCM0–DCM3 issue to reset the flip-flops of the M-Register to the 0-state. Upon issuance of working clock signal QTCK during time period TL1, flip-flops ESX, CAY, PSX and MSX are reset to the

0-state. Run Clock flip-flop RCK is also reset to the 0-state at this time to stop the Program Processor Clock Generator. After the count in the TL-Counter is advanced to TL2, the Program Processor operation stops to wait for synchronizing signal DMDA from Memory. This sequence of micro-operations comprises the first half-sequence of the W01 block.

The second half-sequence of the W01 block of micro-operations is initiated when synchronizing signal DMDA from Memory issues to start the Program Processor Clock Generator. At this time, the auxiliary word is transferred from the memory Data Drivers to the M-Register. When working clock signal QTCK issues during time period TL2, the Memory Write command signal FMWR issues to initiate a restore operation in Memory to restore the auxiliary word to its former location. The operation of the Memory in response to the command signal FMWR is described in the section "Memory—General Operation and Timing." At this time, flip-flop MSX is set to the 1-state to apply the auxiliary word to the X-inputs of the Adder. If signal DIMM has issued to indicate that the auxiliary word is an index, the word stored in the E-Register is applied to the Y-inputs of the Adder so that the index is added to the tentative address.

During time period TL3 of the second half-sequence, flip-flop CL1 is set to the 1-state during the performance of a sequence 0 if the address control field of a P-Sequence AMS Word has a value of seven, indicating that address development is to be continued with another P-Sequence AMS Word. Flip-Flop CL2 is also set to the 1-state during time period TL3 during performance of sequence 0 or sequence 3 if bit 21 of a P-Sequence AMS Word or an Indirect AMS Word respectively is a binary 1. Flip-flop CL2 indicates that the address being developed is an effective address which will be used to obtain an Indirect Address Word from Memory. Signal DPCU also issues at this time to advance the count in the P-Register by one if a sequence 0 is being performed. The contents of the P-Register then represent the address of the next P-sequence word, which may be an instruction word, a P-Sequence AMS Word or a P-Sequence SAS Word.

When working clock signal QTCK issues during time period TL4, working clock signal QSEX issues to transfer the address from the Adder to the E-Register, if signal DXNW is present to indicate continuation of indexing. Working clock signal QSEX also issues during sequence 2 to transfer the Indirect Address Word from the adder to the E-Register.

During time period TL5, when sequence 2 is being performed, the address of the Indirect Address Word is transferred from the D-Register to the Adder if the address control field of the Indirect Address Word has a value of seven, indicating that an Indirect AMS Word occupies the next sequential memory location. Upon issuance of working clock signal QTCK, during the performance of a sequence 2, flip-flop CAY is set to the 1-state if the address control field of the Indirect Address Word has a value of seven, causing one to be added to the address of the Indirect Address Word to form the address of an Indirect AMS Word. If signal DXNW, indicating that indexing is to be continued with a P-Sequence AMS Word, is present at this time, flip-flop PSX is set to the 1-state to transfer the count in the P-Register to the Adder. Flip-flop PSX is also set to the 1-state during sequence 2 if flip-flop ECL1 is set to indicate continuation of indexing and signal $\overline{\text{DNWX}}$ is present to indicate that indirect addressing is not to be continued with an indirect AMS Word. Flip-flop ESX is also reset to the 0-state by this condition. In all sequences except sequence 2 with signal DNWX present to indicate that an Indirect AMS Word is to be employed, working clock signal QSDX issues at this time to effect a transfer from the Adder to the D-Register so long as the instruction is not a two-address instruction or, if a two-address instruction, so long as the second address has not yet been obtained.

When control clock signal QTC0 issues during time period TL0, the Indexing Sequence Register flip-flops XS2 and XS1 are set or reset depending upon the sequence of the W01 block which is to be performed next, assuming that address development has not yet been completed. Thus, flip-flop XS1 is set to the 1-state during sequences 0, 1–$\overline{\text{DXFW}}$ or 3 if indexing is not yet completed and during sequence 2 if indexing is to be continued with an Indirect AMS Word. Flip-flop XS1 is reset to the 0-state if the instruction is a two-address instruction with development of the first address completed and development of the second address to be initiated or if indexing is to be continued with a P-Sequence AMS Word. Flip-flop XS1 is also reset to the 0-state if an index is to be obtained by indirect addressing.

Flip-flop XS2 is set to the 1-state if address development is to be continued with an Indirect Address Word. Flip-flop XS2 is reset to the 0-state if the instruction is a two-address instruction with development of the first address completed and development of the second address to be initiated or when address development is to be continued with a P-Sequence AMS Word. Flip-flop XS2 is also reset to the 0-state during the performance of sequence 2 if indexing is to be continued with P-Sequence AMS Word or if development of the first address of a two-address instruction is completed and development of the second address is to be initiated. In addition, flip-flop XS2 is reset to the 0-state during sequences 0, 1–$\overline{\text{DFXW}}$ and 3 if address development is not yet completed.

Flip-flop NXF is set to the 1-state if the instruction is a two-address instruction with development of the first address complete and development of the second address to be initiated or, during sequence 2, if the instruction is a two-address instruction and development of the first address is not to be continued with an Indirect AMS Word. Flip-flop AOI is set to the 1-state during sequences 0, 1–$\overline{\text{DFXW}}$ or 3 if the auxiliary word is classified as an index pointer, indicating that the next word read from Memory will be an index. Flip-flop AOI is reset to the 0-state when signal DIMM issues. When synchronizing signal $\overline{\text{DMBU}}$ from Memory issues to start the Program Processor Clock Generator, flip-flop RCK is set to the 1-state. The second half-sequence of the W01 block of micro-operations is then complete and the Program Processor is then ready to re-enter the W01 block of micro-operations to complete address development or to enter another block of micro-operations, as defined by the W-Selector Register, to execute the instruction.

The above description takes note of micro-operations of the W01 block which occur generally during the initial routine for all single-address and two-address instructions. Micro-operations of the W01 block applicable to only particular instructions will be indicated in the detailed description of these instructions.

FIGURE 122 is a flow diagram of sequence 0 of the W01 block, illustrating the micro-operations which are performed in sequence 0. The flow chart also illustrates the conditions under which the W01 block is re-entered to continue address development and the sequences performed upon re-entry. Referring to FIG. 122, sequence 0 may be entered from the W00 block or from sequences 1–$\overline{\text{DFXW}}$, 2 or 3, or may be re-entered from sequence 0. Sequence 0 of the W01 block employs either a P-Sequence AMS Word to develop the address of a single-address instruction or the first address of a two-address instruction or a P-Sequence SAS Word to develop the second address of a two-address instruction.

After the P-Sequence AMS or SAS Word has been applied to the Adder, the class of the word, as represented by bits 18 and 19, is checked. If the word is a link, its address field is transferred from the Adder to the D-Register and the W01 block is re-entered in sequence

1-$\overline{\text{DFXW}}$ to develop the index or the operand. If the word is a pointer, the address field of the word is transferred from the Adder to the D-Register, flip-flop AOI is set to the 1-state and block W01 is re-entered in sequence 1-$\overline{\text{DFXW}}$ to develop the index or operand. If the word is an operand, the development of the second address is completed and the appropriate blocks of micro-operations, as defined by the W-Selector Register, are then performed to execute the instruction. If the word is an index, the tentative address in the E-Register is gated to the Adder and modified by the index. The address control field of the P-Sequence AMS Word is then checked and flip-flop CL1 is set to the 1-state if the ACF has a value of seven. Bit 21 of the P-Sequence AMS Word is then checked and flip-flop CL2 is set to the 1-state if bit 21 is a binary 1. The contents of the P-Register are advanced by one to form the address of the next P-sequence word.

At this point, if flip-flop CL2 has been set to the 1-state, the developed address is an effective address and is transferred from the Adder to the D-Register. Sequence 2 of the W01 block is then entered to develop a new effective address, another tentative address or a final address through indirect addressing. If flip-flop CL2 is not set to the 1-state, flip-flop CL1 is checked. If flip-flop CL1 is set to the 1-state, the developed address is transferred to the E-Register, the address in the P-Register is transferred to the Adder and sequence 0 of the W01 block is entered to continue indexing with another P-Sequence AMS Word.

If flip-flop CL1 is not set to the 1-state, and if the address is a single-address instruction, the developed address is a final address and is transferred from the Adder to the D-Register. Address development is then complete and the Central Processor performs the appropriate blocks of micro-operations, as directed by the W-Selector Register, to execute the instruction. If flip-flop CL1 is not set to the 1-state and if the instruction is a two-address instruction, the final first address is transferred from the Adder to the D-Register, flip-flop NXF is set to the 1-state, flip-flop AOI is reset to the 0-state, and sequence 0 of the W01 block is eventually entered, depending upon the instruction, to develop the second address with a P-Sequence SAS Word.

FIGURE 123 is a flow diagram illustrating the micro-operations which are performed in sequence 1-DFXW of the W01 block. Sequence 1-DFXW is entered from the W00 block when the address control field of an instruction word has a value in the range from 1-6. The address control field then defines the location of a Fixed Location Index Word in Memory and is transferred from the Q-Register of the Program Processor to the B-Register of Memory to obtain the Fixed Location Index Word.

If the instruction defined by the operation code in the I-Register is a single-address instruction, the Fixed Location Index Word is an index and is added to the address field of the instruction word in the Adder to form a final address which is transferred to the D-Register. Address development is then completed and the instruction is executed. If the instruction is a two-address instruction, the Fixed Location Index Word is an operand, address development is complete and the instruction is executed by performance of appropriate blocks of micro-operations.

FIGURE 124 is a flow diagram illustrating the micro-operations which are performed in sequence 1-$\overline{\text{DFXW}}$ of the W01 block. The flow diagram also illustrates the conditions under which the W01 block is re-entered after completion of sequence 1-$\overline{\text{DFXW}}$ and the sequences performed on re-entry. Referring to FIG. 124, sequence 1-$\overline{\text{DFXW}}$ may be entered from sequences 0 or 3 or may be re-entered from itself. Remote AMS and Remote SAS Words, in addition to indices, are employed during sequence 1-$\overline{\text{DFXW}}$ to effect address development.

Upon extraction of the auxiliary word from Memory, flip-flop AOI is checked and, if set to the 1-state, indicates that the auxiliary word is an index or an operand. If flip-flop AOI is reset to the 0-state, the auxiliary word is a Remote AMS or a Remote SAS Word and its class is checked. If the word is a pointer or a link, its address field is transferred from the Adder to the D-Register. If the word is a pointer, flip-flop AOI is then set. Subsequent address development is then effected by re-entering sequence 1-$\overline{\text{DFXW}}$ of the W01 block.

If the auxiliary word is indicated to be an index or an operand, either because of self-identification as such or because flip-flop AOI is set to the 1-state, flip-flop NXF is checked. If flip-flop NXF is set to the 1-state, the auxiliary word is an operand, address development is completed and the instruction is executed by performing appropriate blocks of micro-operations. If flip-flop NXF is reset to the 0-state the auxiliary word is an index and is added to the tentative address in the Adder.

The state of flip-flop CL2 is then checked. If flip-flop CL2 is set to the 1-state, this indicates that the developed address is an effective address which is to be employed to obtain an Indirect Address Word. In this event, the effective address is transferred from the Adder to the D-Register and sequence 2 of the W01 block is then entered to develop a new effective address, another tentative address or a final address through indirect addressing. If flip-flop CL2 is reset to the 0-state, flip-flop CL1 is checked and, if set to the 1-state, indicates that indexing is to be continued with another P-Sequence AMS Word. In this event, the address of the next P-sequence word is transferred from the P-Register through the Adder to the D-Register and sequence 0 of the W01 block is entered to continue address development. If flip-flop CL1 is reset to the 0-state, and if the address is a single-address instruction, the developed address is a final address which is transferred to the D-Register and the instruction is executed by performance of appropriate blocks of micro-operations. If the instruction is a two-address instruction, the final first address is stored in the E-Register, flip-flop NXF is set to the 1-state, flip-flop AOI is reset to the 0-state and sequence 0 of the W01 block is eventually entered to develop the second address.

FIGURE 125 is a flow diagram illustrating the micro-operations which are performed in sequence 2 of the W01 block. The flow chart also illustrates the conditions under which the W01 block is re-entered to effect further address development and the sequences in which it is re-entered. Sequence 2 of the W01 block may be entered from sequences 0, 1-$\overline{\text{DFXW}}$ or 3. During performance of sequence 2, an Indirect Address Word is obtained from Memory and employed to further develop the address of a single-address instruction or the first address of the two-address instruction.

Referring to FIG. 125, the Indirect Address Word is transferred to the E-Register and its address control field is checked. If the address control field has a value of seven, signal DNWX issues, indicating that indirect addressing is to be continued with an Indirect AMS Word. In this event, the address at which the Indirect Address Word was stored in Memory is transferred from the D-Register to the Adder where it is incremented by one. The new address is then transferred to the D-Register and sequence 3 of the W01 block is entered to effect further address development with an Indirect AMS Word.

If the address control field of the Indirect Address Word is not equal to seven, flip-flop CL1 is checked and, if set to the 1-state, indicates that indexing is to be continued with the next P-Sequence AMS Word. In this event, the address in the P-Register is transferred through the Adder to the D-Register and sequence 0 is entered. If flip-flop CL1 is reset to the 0-state, and if the instruction is a single-address instruction, address development is completed and the instruction is executed by perform-

ing appropriate blocks as defined by the W-Selector Register. If the address is a two-address instruction, sequence 0 is eventually entered to develop the second address.

FIGURE 126 is a flow diagram illustrating the micro-operations which are performed during sequence 3 of the W01 block. The flow diagram also illustrates the conditions under which the W01 block is re-entered and the sequences in which it is re-entered to effect further address development. Sequence 3 of the W01 block is entered only from sequence 2. During sequence 3, an Indirect AMS Word is obtained from Memory and employed to further develop the address.

Referring to FIG. 126, the Indirect AMS Word is obtained from Memory and its class is checked. If the Indirect AMS Word is a pointer or a link, its address field is transferred to the D-Register and sequence 1–$\overline{\text{DFXW}}$ is entered to develop an index or operand. If the Indirect AMS Word is an index, it is added to the tentative address in the Adder and bit 21 is then checked to determine if further indirect addressing is required. If bit 21 is a binary 1, the developed address is an effective address which is to be used to obtain another Indirect Address Word. In this event, the effective address is transferred from the Adder to the D-Register and sequence 2 is entered to develop a new effective address, another tentative address or a final address through indirect addressing.

If bit 21 is a binary 0, flip-flop CL1 is checked and, if set to the 1-state, indicates that indexing is to be continued with the next P-Sequence AMS Word. In this event, the developed address is transferred to the E-Register and the address in the P-Register is transferred to the D-Register. Sequence 0 is then entered to effect further address development. If flip-flop CL1 is reset to the 0-state, and if the address is a single-address instruction, the developed address is a final address and is transferred to the D-Register. Address development is then completed and the appropriate blocks of micro-operations are performed to execute the instruction. If the instruction is a two-address instruction, sequence 0 of the W01 block is eventually entered to develop the second address.

The following table indicates the sequences of the W01 block which are entered from the W00, W02 and W06R blocks, if address development is required, and the conditions under which the sequences are entered:

W01 BLOCK ENTRY

From—	To—
W00 Block ------	Sequence 0—if ACF of instruction word=7 or if ACF of instruction word=0 and instruction is two-address instruction.
	Sequence 1–DFXW—if ACF of instruction word=1–6.
	Execute—if ACF of instruction word=0 and instruction is single-address instruction.
W02 Block -------	Sequence 0—if ACF of instruction word=0 and instruction is CMM, RIM, ANM, RXM, MFM, ABM or SBM.
	Sequence 1–DFXW—if ACF of instruction word=1–6 and instruction is CMM, RIM, ANM, RXM, MFM, ABM or SBM.
W06R Block ------	Sequence 0—ACF of instruction word=0 and instruction is MOV.
	Sequence 1–DFXW—if ACF of instruction word=1–6 and instruction is MOV.

The following table indicates the order of performance of the sequences of the W01 block in developing a final address and the conditions controlling the order of performance:

From—	To—
Sequence 0 -------	Sequence 0—if P-Sequence AMS Word is index, does not call for indirect addressing and if ACF =7,
	or if P-Sequence AMS Word is index, does not call for indirect addressing and ACF≠7 and instruction is two-address instruction.
	Sequence 1–$\overline{\text{DFXW}}$—if P-Sequence AMS or SAS Word is pointer or link.
	Sequence 2—if P-Sequence AMS Word is index and calls for indirect addressing.
	Execute—if P-Sequence SAS Word is operand,
	or if P-Sequence AMS Word is index, does not call for indirect addressing and ACF=7, and instruction is single-address instruction.
Sequence 1–DFXW	Execute—always.
Sequence 1–$\overline{\text{DFXW}}$	Sequence 0—if Remote AMS Word is index, P-Sequence AMS Word does not call for indirect addressing but ACF of P-Sequence AMS Word=7,
	or if previous P-sequence or Remote AMS Word was pointer, P-Sequence AMS Word does not call for indirect addressing but ACF of P-Sequence AMS Word =7,
	or if Remote AMS Word is index, P-Sequence AMS Word does not call for indirect addressing, ACF of P-Sequence ASM Word≠7 and instruction is two-address instruction,
	or if previous P-Sequence or Remote AMS Word was pointer, P-Sequence AMS Word does not call for indirect addressing, ACF of P-Sequence AMS Word≠7 and instruction is two-address instruction.
	Sequence 1–$\overline{\text{DFXW}}$—if Remote AMS or SAS Word is pointer or link.
	Sequence 2—if Remote AMS Word is index and P-Sequence AMS Word calls for indirect addressing.
	Execute—if Remote SAS Word is operand or if previous P-sequence or Remote AMS Word was pointer,
	or if Remote AMS Word is index, P-Sequence AMS Word does not call for indirect addressing, ACF of P-Sequence AMS Word≠7 and instruction is single-address instruction,
	or if previous P-sequence or Remote AMS Word was pointer, P-Sequence AMS Word does not call for indirect addressing, ACF of P-Sequence AMS Word≠7 and instruction is single-address instruction.

From—	To—
Sequence 2	Sequence 0—if ACF of Indirect Address Word≠7 and if ACF of P-Sequence AMS Word=7, or if ACF of Indirect Address Word≠7, ACF of P-Sequence AMS Word≠7 and address is two-address instruction. Sequence 3—if ACF of Indirect Address Word=7. Execute—if ACF of Indirect Address Word≠7, ACF of P-Sequence AMS Word≠7 and instruction is single-address instruction.
Sequence 3	Sequence 0—if Indirect AMS Word is index and does not call for further indirect addressing and if ACF of P-Sequence AMS Word=7, or if Indirect AMS Word is index and does not call for further indirect addressing, ACF of P-Sequence AMS Word≠7 and instruction is two-address instruction. Sequence 1—$\overline{\text{DFXW}}$—if Indirect AMS Word is pointer or link. Sequence 2—if Indirect AMS Word is index and calls for further indirect addressing. Execute—if Indirect AMS Word is index and does not call for further indirect addressing, ACF of P-Sequence AMS Word≠7 and instruction is single-address instruction.

Instruction 40: Load single (LDS)

This instruction causes the working length of the Accumulator to be set to single and the contents of the single Accumulator to be replaced by the contents of the memory location specified by the final address of the instruction word. The contents of the specified memory location are not changed.

The W00 block of the initial routine, illustrated in FIG. 119, is altered by the issuance of signal DILX during time period TL5 to cause the contents of I-Register flip-flops IR1 and IR0 to be transferred to Accumulator Length Indicator Register flip-flops PL2 and PL1. The two least-significant bits of the operation code in flip-flops IR1 and IR0 represent the working length of the Accumulator, as specified by the instruction word, and the above transfer sets the accumulator working length into the Accumulator Length Indicator Register. If address development is required, the W01 block illustrated in FIG. 121 is unaltered.

After completion of the initial routine, the W-Selector Register is set to W02 and W03, in that order, and the W02 and W03 blocks of micro-operations are performed to execute the instruction. During the W02 block of micro-operations, the operand word which is to be transferred to the Accumulator is read from the memory location designated by the address in the D-Register. The operand word is stored in the M-Register. During the W03 block of micro-operations, the operand word is transferred into the accumulator word location in Memory.

The sequence of occurrence of the micro-operations required to execute Instruction 40 is illustrated in the timing diagram of FIG. 127. FIGURE 127 illustrates the issuance of signal DILX during time period TL5 of the W00 block to set the working length of the Accumulator. During time period TL0 of the W02 block of micro-operations, signal DDBX in the Accumulator Length Indicator Register issues. Signal DDBX is a binary 1 during the entire W02 block and serves to enable the B-Gates so that the operand address in the D-Register is transferred to the B-Register of Memory. When working clock signal QTCK issues during time period TL0, flip-flop MRD is set to the 1-state to initiate a read operation in Memory.

During time period TL1, the flip-flops of the M-Register are cleared. Run Clock flip-flop RCK is reset to the 0-state at this time to stop generation of clock pulses by the Program Processor Clock Generator. The TL-Counter is advanced to TL2 and the Program Processor operation stops to wait for synchronizing signal DMDA from Memory. This sequence of micro-operations comprises the first half-sequence of the W02 block.

The second half-sequence of the W02 block of micro-operations is initiated when synchronizing signal DMDA from Memory issues to start the Program Processor Clock Generator. At this time, the operand word is transferred from the memory Data Drivers to M-Register 301. When control clock signal QTRK issues during time period TL2, flip-flop RCK is set to the 1-state. Upon issuance of working clock signal QTCK, Memory Write command signal FMWR issues to initiate a write operation in Memory to restore the operand word to its former location.

When working clock signal QTCK issues during time period TL5, flip-flop RCK is reset to the 0-state to stop the Program Processor Clock Generator. When the Clock Generator is again started by signal $\overline{\text{DMBU}}$ from Memory, flip-flop RCK is set to the 1-state. Control clock signal QTC0 during time period TL0 causes W-Selector Register flip-flop WR0 to be set to the 1-state to define block W03. The second half-sequence of the W02 block of micro-operations is then complete and the Program Processor is then ready to enter the W03 block to continue instruction execution.

Signal $\overline{\text{DAMR}}$ issues during time period TL0 of the W03 block to inhibit the transfer of a word from Memory to the M-Register during the first half-sequence of the W03 block. Signal DABX issues at this time to transfer the accumulator address from the A-Register and the Accumulator Counter Register to the B-Register of Memory. Working clock signal QTCK causes signal FMRD to issue during time period TL0 to initiate a read operation in Memory which clears the addressed accumulator location. During time period TL1, signals $\overline{\text{DCM0}}$–$\overline{\text{DCM8}}$ are provided to inhibit clearing of the operand from the M-Register. During time period TL1, the Program Processor Clock Generator is again stopped. Flip-flop RCK is set during time period TL2 when the Program Processor Clock Generator is again started by synchronizing signal DMDA.

When signal QTCK issues during time period TL2, a write operation is initiated to store the operand in the addressed accumulator location in Memory. During time period TL5, flip-flop PSX is set to the 1-state to cause the next instruction address to be transferred from the P-Register to the Adder. Flip-flop RCK is reset at this time to stop the Program Processor Clock Generator. During time period TL0, flip-flops WR1 and WR0 of the W-Selector Register are reset to the 0-state, preparatory to execution of the next instruction. The Clock Generator of the Program Processor is started upon receipt of signal $\overline{\text{DMBU}}$ from Memory.

Instruction 41: Load double (LDD)

This instruction causes the accumulator working length to be set to double and the contents of the double Accumulator to be replaced by two-word memory field from specified adjacent memory locations. The contents of the specified memory locations are not changed.

The execution of Instruction 41 is the same as that for Instruction 40, described above, with the following exceptions. During time period TL5 of the W03 block which transfers the first operand word to the accumulator word A location, as illustrated in FIG. 127, flip-flop PSX is not set since the last accumulator operation has not occurred.

The address of the next instruction is therefore not transferred from the P-Register to the Adder. Rather, the address in the D-Register is decreased by one to address the second operand word of the memory field which is to be transferred to accumulator word B location in Memory. During time period TL0, only flip-flop WR0 of the W-Selector Register is reset to the 0-state. Flip-flop WR1 remains set to define block W02. At this time, the count in the Accumulator Counter Register is advanced by one to address accumulator word B.

The Program Processor then returns to the W02 block of micro-operations to extract the second operand word of the two-word memory field. Following performance of block W02, block W03 is again repeated. During the second half-sequence of block W03, the second operand word of the memory field is stored in the accumulator word B location in Memory. Signal DLAO issues since the last accumulator operation required by Instruction 41 is being performed. Flip-flop PSX is then set to the 1-state to transfer the address of the next instruction from the P-Register to the Adder. The flip-flops of the W-Selector Register are reset to the 0-state to return the Program Processor to the W00 block of micro-operations, preparatory to execution of the next instruction.

Instruction 42: Load triple (LDT)

This instruction causes the working length of the Accumulator to be set to triple and the contents of the triple Accumulator to be replaced by a three-word memory field from specified adjacent memory locations. The contents of the specified memory locations are not changed.

The execution of Instruction 42 is identical to execution of Instruction 41 except that the micro-operations of the W02 and W03 blocks, illustrated in FIG. 127, are repeated three times, once for each word of the three-word memory field to be transferred to the Accumulator. During the third performance of the W03 block, signal DLAO issues to indicate the last accumulator operation. Instruction execution is then terminated by the issuance of signal FPSX and the resetting of the flip-flops of the W-Selector Register to the 0-state, preparatory to execution of the next instruction.

Instruction 43: Load quadruple (LDQ)

This instruction causes the working length of the Accumulator to be set to quadruple and the contents of the quadruple Accumulator to be replaced by a four-word memory field from specified adjacent memory locations. The contents of the specified memory locations are not changed.

Execution of Instruction 43 is identical to execution of Instruction 42 except that the micro-operations of the W02 and W03 blocks, illustrated in FIG. 127, are repeated four times, once for each word of the four-word memory field to be transferred to the Accumulator. During the fourth performance of the W03 block, signal DLAO issues to indicate the last accumulator operation. Termination of execution of Instruction 43 is the same as that for Instructions 40–42, as described above.

Instruction 44: Store single (STS)

This instruction causes the contents of a specified memory location to be replaced by the contents of the single Accumulator, regardless of the working length of the Accumulator. The working length and the contents of the Accumulator are not changed.

The W00 block of the initial routine, illustrated in FIG. 119 is unaltered. If address development is required, the W01 block illustrated in FIG. 121 is unaltered.

After completion of the initial routine, the W-Selector Register is set to W02 and W03, in that order, and the W02 and W03 blocks of micro-operations are performed to execute the instruction. During the W02 block of micro-operations, the accumulator word which is to be transferred to a specified memory location is read from Memory and stored in the M-Register. During the W03 block of micro-operations, the accumulator word is transferred to the specified storage location in Memory.

The sequence of occurrence of the micro-operations required to execute Instruction 44 is illustrated in the timing diagram of FIG. 128. During time period TL0 of the W02 block of micro-operations, signal DABX issues. Signal DABX is a binary 1 during the entire W02 block and serves to enable the B-Gates so that the address of accumulator word A is transferred from the A-Register and the Accumulator Counter Register to the B-Register of Memory. Working clock signal QTCK causes signal FMRD to issue during time period TL0 to initiate a read operation in Memory.

During time period TL1, the flip-flops of the M-Register are cleared. Run Clock flip-flop RCK is reset to the 0-state at this time to stop generation of clock pulses by the Program Processor Clock Generator. The TL-Counter is advanced to TL2 and the Program Processor operation stops to wait for synchronizing signal DMDA from Memory. This sequence of micro-operations comprises the first half-sequence of the W02 block.

The second half-sequence of the W02 block of micro-operations is initiated when synchronizing signal DMDA from Memory issues to start the Program Processor Clock Generator. At this time, accumulator word A is transferred from the memory Data Drivers to M-Register 301. When control clock signal QTRK issues during time period TL2, flip-flop RCK is set to the 1-state. Upon issuance of working clock signal QTCK, Memory Write command signal FMWR issues to initiate a write operation in Memory to restore accumulator word A to its former location.

When working clock signal QTCK issues during time period TL5, flip-flop RCK is reset to the 0-state to stop the Program Processor Clock Generator. When the Clock Generator is again started by signal $\overline{\text{DMBU}}$ from Memory, flip-flop RCK is set to the 1-state. Control clock signal QTC0 during time period TL0 causes W-Selector Register flip-flop WR0 to be set to the 1-state to define block W03. The second half-sequence of the W02 block of micro-operations is then complete and the Program Processor then enters the W03 block to continue instruction execution.

Signal $\overline{\text{DAMR}}$ issues during time period TL0 of the W03 block to inhibit the transfer of a word from Memory to the M-Register during the first half-sequence of the W03 block. Signal DDBX issues at this time to transfer the address of the specified memory location from the D-Register to the B-Register of Memory. Working clock signal QTCK causes signal FMRD to issue during time period TL0 to initiate a read operation in Memory which clears the addressed memory location. During time period TL1, signals $\overline{\text{DCM0}}$–$\overline{\text{DCM8}}$ are provided to inhibit clearing of accumulator word A from the M-Register. During time period TL1, the Program Processor Clock Generator is again stopped. Flip-flop RCK is reset during time period TL2 when the Program Processor Clock Generator is again started by synchronizing signal DMDA.

When signal QTCK issues during time period TL2, a write operation is initiated to store accumulator word A in the addressed memory location. During time period TL5, flip-flop PSX is set to the 1-state to cause the next instruction address to be transferred from the P-Register to the Adder. Flip-flop RCK is reset at this time to stop the Program Processor Clock Generator. During time period TL0, flip-flops WR1 and WR0 are reset to the 0-state, preparatory to execution of the next instruction. The Clock Generator of the Program Processor is started upon receipt of signal $\overline{\text{DMBU}}$ from Memory.

Instruction 45: Store double (STD)

This instruction causes the contents of two specified adjacent memory locations to be replaced by accumulator

words A and B, regardless of the working length of the Accumulator. The accumulator length and contents are not changed.

The execution of Instruction 45 is identical to the execution of Instruction 44 with the following exceptions. During time period TL5 of the W03 block which transfers accumulator word A to the specified memory location, as illustrated in FIG. 128, flip-flop PSX is not set since the last accumulator operation has not occurred. The address of the next instruction is therefore not transferred from the P-Register to the Adder. Rather the address in the D-Register is decreased by one to address the second memory location, the contents of which are to be replaced by accumulator word B. During time period TL0, only flip-flop WR0 of the W-Selector Register is reset to the 0-state. Flip-flop WR1 remains set to define block W02. At this time, the count in the Accumulator Counter Register is advanced by signal DACU to address accumulator word B. The Program Processor then returns to the W02 block of micro-operations to extract accumulator word B from Memory.

Following the performance of block W02, block W03 is again repeated. During the second half-sequence of block W03, accumulator word B is stored in the second specified memory location. Signal DLAO issues since the last accumulator operation required by Instruction 45 is being performed. Flip-flop PSX is then set to the 1-state to transfer the address of the next instruction from the P-Register to the Adder. The flip-flops of the W-Selector Register are reset to the 0-state to return the Program Processor to the W00 block of micro-operations, preparatory to execution of the next instruction.

Instruction 46: Store triple (STT)

This instruction causes the contents of three specified adjacent memory locations to be replaced by accumulator words, A, B and C, regardless of the working length of the Accumulator. The working length of the Accumulator and the contents of the Accumulator are not changed.

The execution of Instruction 46 is identical to execution of Instruction 45 except that the micro-operation of the W02 and W03 blocks, illustrated in FIG. 128, are repeated three times, once for each accumulator word to be transferred to a specified memory location. During the third performance of the W03 block, signal DLAO issues to indicate the last accumulator operation. Instruction execution is then terminated by the issuance of signal FPSX and the resetting of the flip-flops of the W-Selector Register to the 0-state, preparatory to execution of the next instruction.

Instruction 47: Store quadruple (STQ)

This instruction causes the contents of four specified adjacent memory locations to be replaced by accumulator words A, B, C and D, regardless of the working length of the Accumulator. The working length of the Accumulator and the contents of the Accumulator are not changed.

Execution of Instruction 47 is identical to execution of Instruction 46 except that the micro-operations of the W02 and the W03 blocks, illustrated in FIG. 128, are repeated four times, one for each accumulator word to be transferred to a specified memory location. During the fourth performance of the W03 block, signal DLAO issues to indicate the last accumulator operation. Termination of execution of Instruction 47 is the same as that for Instructions 44–46, as described above.

Instruction 30: Move from first memory (MFM)

This instruction causes the contents of a first specified memory location to be moved to a second specified memory location, leaving the contents of the first specified memory location unchanged. The address field of the instruction word may provide the address of the first specified memory location or this address may be developed by employing auxiliary words. The address of the second specified memory location is specified to be the address of a Fixed Location Index Word or is developed by employing SAS auxiliary words.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. If development of the address of the first specified memory location is required, the W01 block illustrated in FIG. 121 is unaltered.

After completion of the W00 block and development of the address of the first specified memory location in the W01 block, if required, the W-Selector Register is set to W02, W01 and W15S, in that order. The micro-operations corresponding to these blocks are performed to develop the second address of the instruction and to execute the instruction. During the W02 block of micro-operations, the operand word in the first specified memory location is extracted from Memory and stored in the E-Register of the Program Processor. During the W01 block, the address of the second specified memory location is developed and stored in the D-Register. The contents of the memory location specified by the second address are cleared and the operand word from the first specified memory location in the E-Register is applied to the Adder. During the subsequent W15S block, the operand word is transferred to the second specified memory location to complete execution of the instruction.

The sequence of occurrence of the micro-operations required to execute Instruction 30 is illustrated in the timing diagrams of FIG. 129. During time period TL0 of the W02 block of micro-operations, signal DDBX issues to enable the B-Gates so that the operand address in the D-Register is transferred to the B-Register of Memory. When working clock signal QTCK issues during time period TL0, flip-flop MRD is set to the 1-state to initiate a read operation in Memory.

During time period TL1, the flip-flops of the M-Register are cleared, preparatory to receiving the operand from Memory. Upon issuance of working clock signal QTCK, flip-flop MSX is set to the 1-state. Signal FMSX causes the operand word to be transferred from the M-Register to the Adder when the operand word is received in the M-Register from Memory during the read operation. Run Clock flip-flop RCK is reset to the 0-state at this time to stop generation of clock pulses by the Program Processor Clock Generator. The TL-Counter is advanced to TL2 and the Program Processor operation stops to wait for synchronizing signal DMDA from Memory. This sequence of micro-operations comprises the first half-sequence of the W02 block.

The second half-sequence of the W02 block of micro-operations is initiated when synchronizing signal DMDA from Memory issues to start the Program Processor Clock Generator. At this time, the operand word is transferred from the memory Data Drivers into the M-Register. When control clock signal QTRK issues during time period TL2, flip-flop RCK is set to the 1-state. Upon issuance of working clock signal QTCK, Memory Write command signal FMWR issues to initiate a write operation in Memory to restore the operand word to its former location.

During time period TL4, working clock signal QSEX issues to transfer the operand word through the Adder to the E-Register. During time period TL5, flip-flop PSX is set to the 1-state to transfer the address of the next P-sequence word from the P-Register to the Adder. This is the address of the P-Sequence SAS Word which will be used to develop the second address of the instruction. At this time flip-flop RCK is reset to the 0-state to stop the Program Processor Clock Generator. When control clock signal QTC0 issues during time period TL0, W-Selector Register flip-flops WR1 and WR0 are set and reset respectively, causing the Program Processor to return to the W01 block to develop the second address of the instruction. The second half-sequence of the W02 block is complete and the Program Processor enters the W01 block.

The micro-operations of the W01 block are the same as those illustrated in FIG. 121 with the following exceptions. During time period TL2 of the W01 block when the second address has been developed and the contents of the second specified memory location have been read into the M-Register from Memory, restoration of the contents of the second specified memory location to Memory is inhibited by signal DVCA, which prevents flip-flop MWR from being set to the 1-state. The memory location corresponding to the second address is thus cleared. Flip-flop ESX is set at this time to transfer the operand word in the E-Register to the Adder. During time period TL5 of the previous sequence of the W01 block, flip-flop MSX was reset to the 0-state to prevent the word read from the second specified memory location from being transferred to the Adder. During time period TL0, W-Selector Register flip-flops WR3, WR2 and WR1 and TL-Counter flip-flop TL2 are set to the 1-state, while TL-Counter flip-flop TL0 is reset to the 0-state, to define the second half-sequence of the W15 block.

During time period TL2 of block W15S, either signal DDBX or signal DQBX issues, depending upon the address control field of the instruction word, to transfer the address of the second location from either the D-Register or the Q-Register to the B-Register of Memory. Flip-flop MWR is set to the 1-state to initiate a write operation in Memory to store the operand word in the second specified memory location. Working clock signals QSMA, QSMB and QSMC issue at this time to transfer the operand word through the Adder to the M-Register, preparatory to storing the operand in Memory during the write operation. This transfer clears the contents of the second specified memory location from the M-Register. During time period TL5, flip-flop PSX is set to the 1-state to transfer the address of the next instruction to the Adder. Flip-flop RCK is reset to the 0-state to stop the Program Processor Clock Generator until synchronizing signal $\overline{\text{DMBU}}$ is provided by Memory. During time period TL0, flip-flops WR3, WR2, WR1 and WR0 are reset to the 0-state to define the W00 block, preparatory to execution of the next instruction.

Instruction 31: Move from immediate (MFI)

This instruction causes the address field of the instruction word to be moved to the address field of an operand word in a second specified memory location. The operation code and the address control field of the operand word in the second specified memory location are not changed. The address field of the instruction word may be developed by employing auxiliary words. The address of the second memory location is specified to be the address of a Fixed Location Index Word or is defined by employing SAS auxiliary words.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. If development of the address field of the instruction word is required, the W01 block illustrated in FIG. 121 is unaltered. During development of the address of the second specified memory location, the W01 block of FIG. 121 is altered as follows. During time period TL2 signal DVCA, indicating that the address of the second specified memory location has been obtained, inhibits the restoration of the operand in the second specified memory location by preventing flip-flop MWR from being set to the 1-state, as illustrated in FIG. 130. At this time, the address field in the E-Register is transferred to the Adder by signal FESX. Flip-flop MSX is reset during the previous W01 sequence to prevent the operand in the M-Register from being applied to the Adder.

After completion of the initial routine, the W-Selector Register is preset to W15 and the W15S block of micro-operations is performed to complete execution of the instruction. During the W15S block, the address field of the operand in the M-Register is replaced by the address field of the instruction word and the operand is stored in the second specified memory location.

The sequence of occurrence of the micro-operations required to execute Instruction 31 is illustrated in the timing diagram of FIG. 130. During time period TL2 of block W15S, the address of the second specified memory location is transferred from the D-Register or the Q-Register, depending upon the value of the address control field of the instruction word, to the B-Register of Memory. Flip-flop MWR is set to the 1-state at this time to initiate a write operation to store the operand word with its new address field in the second specified memory location. Working clock signals QSMA and QSMB issue at this time to transfer the address field of the instruction word through the Adder into bit positions 0–14 of the M-Register. This transfer clears the old address field of the operand from the M-Register. The operand with the address field of the instruction word is then stored in the second specified memory location during the write operation. During time period TL5, flip-flop PSX is set to the 1-state to transfer the address of the next instruction in the P-sequence to the Adder. Flip-flop RCK is reset to the 0-state to stop the Program Processor Clock Generator. Flip-flops WR3, WR2, WR1 and WR0 are reset to the 0-state to define the W00 block, preparatory to execution of the next instruction.

Instruction 32: Move to first address field (MTA)

This instruction causes the address field of the operand word in a first specified memory location to be replaced by the address field of an operand word in a second specified memory location. The contents of the second specified memory location are not changed. The address of the first specified memory location may be developed by employing auxiliary words while the address of the second specified memory location is specified to be the address of a Fixed Location Index Word or is developed by employing SAS auxiliary words.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine, illustrated in FIG. 121, is unaltered during development of the address of the first specified memory location, if required. The W01 block of the initial routine is altered during development of the address of the second specified memory location as follows. During time period TL2, when the address of the second specified memory location has been developed, the transfer of the operand word read from the second specified memory location from the M-Register to the Adder is inhibited by signal $\overline{\text{DVCA}}$, as illustrated in FIG. 131. At this time, the address of the first specified memory location stored in the E-Register is transferred to the Adder by signal FESX. During time period TL4, signal QSDX causes the address of the first specified memory location to be transferred from the Adder to the D-Register. During time period TL5, the operand word read from the second specified memory location is transferred from the M-Register to the Adder while flip-flop ESX is reset to the 0-state to inhibit the transfer of the address of the first specified memory location from the E-Register to the S-Register. During time period TL0, W-Selector Register flip-flop WR2 is set to define the W05 block.

After completion of the initial routine, the W-Selector Register is set to W05 and W15S, in that order, and the W05R and W15S blocks of micro-operations are performed to execute the instruction. During the W05R block, the operand word in the first specified memory location is stored in the M-Register and the operand word in the second specified memory location is stored in the E-Register. During the subsequent W15S block, the address field of the operand in the M-Register is replaced by the address field of the operand word in the E-Register and the operand word in the M-Register is stored in the first specified memory location.

The sequence of occurrence of the micro-operations required to execute Instruction 32 is illustrated in the timing diagram of FIG. 131. During time period TL0 of the W05 block, signal DDBX issues to transfer the address of the first specified memory location through the B-Gates to the B-Register of Memory. Flip-flop MRD is set to the 1-state to initiate a read operation which extracts the contents of the first specified memory location and stores them in the M-Register. Working clock signal QSEX issues at this time to transfer the operand word read from the second specified memory location through the Adder to the E-Register. During time period TL1, the M-Register is cleared, preparatory to receiving the operand word from the first specified memory location during the read operation, and the operand word read from the second specified memory location is transferred from the E-Register to the Adder. Flip-flop MSX is reset to the 0-state and the first half-sequence of the W05 block is terminated in a conventional manner. During time period TL2, flip-flops WR3 and WR1 are set to the 1-state to define block W15. The second half-sequence of the W05 block is not performed. The count TL2 of the TL-Counter remains the same and the micro-operations of the first half-sequence of the W15 block are therefore not performed.

When working clock signal QTCK issues during time period TL2 of block W15S, a write operation is initiated in Memory. Working clock signals QSMA and QSMB issue at this time to transfer the address field of the operand word read from the second specified memory location through the Adder to the M-Register. This transfer clears the old address field of the operand from the M-Register. With the address of the first specified memory location in the B-Register of Memory, the operand from the first specified memory location with the address field of the operand from the second specified memory location is written into the first specified memory location.

During time period TL5 of the W15S block, the address of the next instruction is transferred to the Adder. The second half-sequence of block W15S is terminated in a conventional manner with flip-flops WR3–WR0 of the W-Selector Register being reset to the 0-state to define the W00 block.

Instruction 06: Move (MOV)

This instruction causes the contents of a first memory field to be replaced by words moved from a second memory field. The second memory field is not changed. The address of the most-significant word of the first memory field is specified by or is developed from the address field of the instruction word. The address of the most-significant word of the second memory field is defined by bits 1–14 of a control word. Bit positions 15–23 of the control word contains a count in the form of 512–N, expressed in binary, where N is the number of words to be moved from the second memory field to the first memory field. The location of the control word is specified by the address of a Fixed Location Index Word or is defined by employing a P-Sequence SAS Word and, if necessary, Remote SAS Words. The organization of the control word is illustrated in FIG. 2*k*.

The W00 block of the initial routine, illustrated in FIG. 119 is unaltered. The W01 block of the initial routine illustrated in FIG. 121, if necessary to develop the first address of the instruction, is unaltered.

After development of the first address of the instruction, if required, the W-Selector Register is set to W15 and W06, in that order, and the W15 and W06R blocks of micro-operations are performed by the Program Processor to initiate execution of the instruction. The Program Processor then returns to block W01 to develop the second address of the instruction. The W-Selector Register is then set to W06, W02, W03, W04 and W06, in that order, and the W06R, W02, W03, W04R and W06S blocks of micro-operations are performed to complete execution of the instruction. The W02 and W03 blocks are repeated once for each word to be moved from the second memory field to the first memory field prior to performance of blocks W04R and W06S.

During the W15 block of micro-operations, after the initial routine is completed, the address and working length of the Accumulator are stored in a fixed location in Memory. During block W06R, the address of the most-significant word of the first memory field is transferred to the E-Register. After development of the second address in the W01 block, block W06S causes the address of the most-significant word of the first memory field to be transferred back to the D-Register while the control word, whose location is defined by the developed second address, is transferred to the E-Register. The address of the most-significant word of the second memory field, contained in bit positions 0–14 of the control word, is transferred to the A-Register and to flip-flops PL2 and PL1. During blocks W02 and W03, the most-significant word of the second memory field is transferred to the most-significant word position of the first memory field. The count and address field of the control word, as well as the address of the first memory field in the D-Register, are each incremented by one. Blocks W02 and W03 are repeated until the count field of the control word becomes equal to zero, setting Carry Remember flip-flop CRE to the 1-state. Blocks W04R and W06S are then performed to restore the accumulator location and length to the A-Register and to flip-flops PL2 and PL1 respectively.

The sequence of occurrence of the micro-operations required to execute Instruction 06 is illustrated in the timing diagram of FIG. 132. During time period TL0 of the W15 block of micro-operations, signal DFBS issues, enabling B-Gate B03 to address fixed location 10 (octal) in Memory. During time period TL1, the address and the working length of the Accumulator are transferred from the A-Register and the Accumulator Length Indicator Register respectively to the E-Register. When working clock signal QTCK issues, the address and working length of the Accumulator are transferred from the E-Register to the Adder. Synchronization of memory operation with the initiation of the second half-sequence of the W15 block is accomplished as previously described.

During time period TL2, flip-flop MWR is set to the 1-state to initiate a write operation in Memory to store the address and working length of the Accumulator in fixed location 10 (octal) in Memory. Working clock signals QSMA and QSMB issue at this time to transfer the address and working length of the Accumulator from the Adder to the M-Register, preparatory to effecting the write operation. During time period TL3, the address of the most-significant word of the first memory field is transferred from the D-Register to the Adder. During time period TL5, flip-flop RCK is reset to the 0-state to stop the Clock Generator of the Program Processor. During time period TL0, flip-flops WR3 and WR0 are reset to the 0-state to define block W06 and the Clock Generator of the Program Processor is again started in the conventional manner.

During time period TL0 of the W06R block, working clock signal QSEX issues to transfer the address of the most-significant word of the first memory field to the E-Register. During time period TL1, the address of the P-Sequence SAS Word which is to be used in the development of the second address is transferred from the P-Register to the Adder by signal FPSX. Flip-flop RCK is reset and set, as previously described, to synchronize the Program Processor Clock Generator with the memory operation. During time period TL2 of the W06 block, the flip-flops of the W-Selector Register are preset to define block W01. Flip-flops TL0 and TL2 of the TL-Counter are set and reset respectively to define the first half-sequence of the W01 block.

During the W01 block, the second address of Instruction 06 is developed employing a P-Sequence SAS Word and Remote SAS Word, as necessary. The timing of the W01 block and the development of the second address is essentially the same as that illustrated in FIG. 121. After development of the second address during the W01 block, the control word at the memory location corresponding to the second address is placed in the M-Register. The address of the most-significant word of the first memory field in the E-Register is applied to the Adder. At the end of the W01 block during time period TL0, the flip-flops of the W-Selector Register are preset to define block W06 and the TL-Counter is preset to time period TL2 to define the second half-sequence of the W06 block.

During time period TL2 of block W06S, flip-flop MSX is set to the 1-state to transfer the control word from the M-Register to the Adder. Simultaneously, the address of the most-significant word of the first memory field is transferred from the Adder to the D-Register. The A-Register is cleared during time period TL3 by signal DCLA. The control word is transferred to the E-Register during time period TL4 by working clock signal QSEX. During time period TL5, bits 2–14 of the control word are transferred to the A-Register while bits 0 and 1 of the control word are stored in the Accumulator Length Indicator Register. During time period, TL0, flip-flop WR2 of the W-Selector Register is reset to the 0-state so that the W-Selector Register defines time period W02.

Signal DABX, issuing during time period TL0 of block W02, transfers the address of the most-significant word of the second memory field from the A-Register and the Accumulator Length Indicator Register to the B-Register of Memory. A read operation is then initiated. During time period TL1, flip-flop IRE is set to the 1-state to provide input DX15 to the Adder. The control word in the E-Register is then applied to the Adder, adder input DX15 serving to increment the count field of the control word by one. Flip-flop CAY is also set to the 1-state at this time to increment the address field of the control word by one.

During the second half-sequence of the W02 block, the addressed word of the second memory field is restored. The A-Register is cleared by signal DCLA and the the control word is returned to the E-Register during time period TL4. The incremented address field of the control word, which represents the address of the next most-significant word of the second memory field, is again stored in the A-Register and the Accumulator Length Indicator Register. Carry Remember flip-flop CRE is set to the 1-state during the second half-sequence of the W02 block if a carry from first rank adder circuit S23 is detected, i.e., if bits 15–23 of the control word have become binary 0's. During time period TL0 of the W02 block, flip-flop WR0 is set to the 1-state so that the W-Selector Register defines block W03.

During time period TL0 of block W03, signal DDBX issues to transfer the address of the most-significant word of the first memory field to the B-Gates. A memory read cycle is initiated and the transfer of the word in the addressed memory location to the M-Register is inhibited by signal DASTR, the read operation serving to clear the addressed location of the first memory field. During the second half-sequence of the W03 block, a write operation is initiated to store the most-significant word of the second memory field in the most-significant word location of the first memory field. The address of the most-significant word of the first memory field is transferred to the Adder by signal FDSX during time period TL3, is incremented by one and is restored to the D-Register by signal QSDX during time period TL5 to address the next most-significant word location of the first memory field.

The Program Processor returns to the W02 block to derive the next most-significant word of the second memory field from Memory. The W03 block of micro-operations then serves to store the second most-significant word of the second memory field in the second most-significant word location of the first memory field. The number of repetitions of the W02 and W03 blocks is equal to the number of words in the second memory field to be transferred to the first memory field, as defined by N in the count 512–N in the count field of the control word. At the end of the W03 block transferring the last word of the second memory field to the last word location of the first memory field, the W-Selector Register is preset to define the W04 block instead of the W02 block.

Signal DFBS again issues during time period TL0 of block W04R to address fixed memory location 10 (octal). The read operation initiated during time period TL0 reads the accumulator location and length from location 10 (octal) into the M-Register. At the end of the first half-sequence of the W04 block, i.e. block W04R, flip-flop WR1 is set to the 1-state so that the W-Selector Register defines block W06. The state of the TL-Counter is not changed at this time so that the second half-sequence of block W06, i.e. block W06S, is defined.

During block W06S, flip-flop MSX is set to the 1-state during time period TL0 to transfer the accumulator location and length to the Adder. The A-Register is cleared during time period TL3 by signal DCLA. During time period TL4, working clock signal QSEX issues to transfer the accumulator location and length to the E-Register. During time period TL5, the accumulator location and length is restored to the A-Register and to flip-flops PL2 and PL1 respectively. Flip-flop PSX is set to the 1-state to transfer the address of the next instruction to the Adder. At the end of block W06S, the W-Selector Register is preset to define the W00 block, preparatory to execution of the next instruction.

Instruction 20: Explode (EXP)

This instruction causes the working Accumulator to be cleared to zero and a character from a word in a specified memory location to be stored in character 3 of each word of the working Accumulator as follows: character 3 of the word in the specified memory location to the character 3 position of accumulator word A; character 2 of the word in the specified memory location to the character 3 position of accumulator word B, if the accumulator working length is double; character 1 of the word in the specified memory location to the character 3 position of accumulator word C, if the accumulator working length is triple; and character 0 of the word in the specified memory location to the character 3 positon of accumulator word D, if the accumulator working length is quadruple. Accumulator words outside the working Accumulator and the contents of the specified memory location are not changed. The apparatus described in this section includes the invention of David E. Keefer.

The W00 block of the initial routine, illustrated in FIG 119, is unaltered. If development of the instruction address field is acquired, the W01 block of the initial routine illustrated in FIG. 121 is unaltered.

After completion of the initial routine, the W-Selector Register is set to W02 and W03, in that order, and the W02 and W03 blocks of micro-operations are performed to execute the instruction, if the accumulator working length is single. If the accumulator working length is greater than single, block W14S follows block W03 and the W03 block is again performed following block W14S. The series of blocks W14S and W03 is repeated once if the accumulator working length is double, twice if the accumulator working length is triple and three times if the accumulator working length is quadruple.

During the W02 block of micro-operations following the initial routine, the operand word in the specified memory location is stored in the D-Register. During block W03, character 3 of the operand word in the E-Register is stored in bit positions 0–5 of accumulator word A.

If the accumulator working length is greater than single, the W14S block is performed. During the W14S block, the contents of the E-Register are shifted right six bit positions to place character 2 of the operand word in bit positions 0–5 of the E-Register. The Accumulator Counter Register is advanced by one. During the subsequent W03 block, character 2 of the operand word in the E-Register is stored in bit positions 0–5 of accumulator word B. Blocks W14S and W03 are repeated for each additional word of the working Accumulator. After the last accumulator operation, execution of the instruction is terminated in block W03. The sequence of occurrence of the micro-operations required to execute Instruction 20 is illustrated in the timing diagram of FIG. 133. During time period TL0 of the W02 block, signal DDBX issues to transfer the operand address in the D-Register to the B-Register of Memory. Signal DDBX is present during the entire W02 block. Working clock signal QTCK causes a read operation to be initiated in Memory. During time period TL1, the M-Register is cleared preparatory to receiving the operand word from Memory during the read operation. The Q-Register is also cleared. Working clock signal QTCK sets flip-flop MSX during time period TL1, causing the operand word in the M-Register to be transferred to the Adder as soon as it is read from Memory. During the second half-sequence of the W02 block, the operand word is restored to Memory and is transferred from the Adder to the E-Register by working clock signal QSEX. The W02 block is terminated and flip-flop WR0 of the W-Selector Register is set to the 1-state to define block W03.

During time period TL0 of the W03 block, the address of accumulator word A is transferred from the A-Register and the Accumulator Counter Register to the B-Register of Memory. Signal $\overline{\text{DAMR}}$ issues at this time to prevent the contents of accumulator word A from being read out of the Memory into the M-Register during the subsequent read operation initiated by signal FMRD. The accumulator word A location in Memory is thus effectively cleared. During time period TL1, the M-Register is cleared. At this time, working clock signal QTCK sets flip-flop ESX, causing the operand to be transferred to the Adder. During the second half-sequence of the W03 block, working clock signal QSMA issues during time period TL2 to cause bits 0–5 of the operand word, i.e. character 3, to be transferred from the Adder to the M-Register. During the subsequent memory write operation initiated by signal FMWR during time period TL2, bits 0–5 of the operand word are stored in bit positions 0–5 of the accumulator word A location in Memory.

If the accumulator working length is single, the address of the next instruction is transferred from the P-Register to the Adder by signal FPSX during time period TL5 and the Program Processor goes to the W00 block, preparatory to execution of the next instruction. If the accumulator working length is greater than single, the TL-Counter and W-Selector Register flip-flops are preset to W14S and the W14S block of micro-operations is next performed.

During the time period TL2 of the W14S block, Shift Right flip-flop SRI is set to the 1-state. During time period TL3, flip-flop Q00 of the Q-register is set to the 1-state to establish the count of one in the Q-Register. Signal QSHE issues at this time to shift the contents of the E-Register right one bit position. During time period TL4, the contents of the E-Register are again shifted right one bit position and the count in the Q-Register is advanced to two. During time period TL5, signal FSRI inhibits QTLP and the advance of the TL-Counter while working clock signal QTCK issues four times. The TL-Counter remains at TL5 during this period. As each signal QTCK issues, the contents of the E-Register are shifted right one bit position and the count in the Q-Register is advanced by one. When the count in the Q-Register is five, working clock signal QTCK causes flip-flop SRI to be reset to the 0-state, the count in the Q-Register to be returned to zero, and one more shift to be performed in the E-Register. A total of six shifts are thus performed to place character 2 of the operand word in bit positions 0–5 of the E-Register. The bits shifted out of the E-Register are lost. QTLP again issues to advance the TL-Counter. During time period TL0 of the W14S block, the W-Selector Register flip-flops are preset to define the W03 block and the count in the Accumulator Counter Register is advanced by one by signal DACU to define, in conjunction with the contents of the A-Register, the address of the accumulator word B location in Memory.

During the subsequent W03 block, character 2 of the operand word in bit positions 0–5 of the E-Register is stored in bit positions 0–5 of the accumulator word B location in Memory. The W-Selector Register is preset to define block W00 at the end of the W03 block, if the accumulator working length is double. If the accumulator working length is triple, blocks W14S and W03 are again performed to store character 1 of the operand word in the positions 0–5 of the accumulator word C location in Memory. If the accumulator working length is quadruple, blocks W14S and W03 are performed for the third time to store character 0 of the operand word in bit positions 0–5 of the accumulator word D location in Memory.

Instruction 21: Implode (IMP)

This instruction causes the least-significant character, i.e. character 3, of each working accumulator word to be gathered into a specified memory location as follows: character 3 of accumulator word A is stored in the character 3 position of the specified memory location; character 3 of accumulator word B, if the working accumulator length is greater than single, is stored in the character 2 position of the specified memory location; character 3 of accumulator word C, if the working accumulator length is greater than double, is stored in the character 1 position of the specified memory location; and character 3 of accumulator word D, if the accumulator working length is quadruple, is stored in the character 0 position of the specified memory location. If any accumulator words are excluded because the working length of the accumulator is less than quadruple, the corresponding character positions in the specified memory location are cleared to zero. The accumulator contents and length are not changed. The apparatus described in this section includes the invention of David E. Keefer.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine, illustrated in FIG. 121, is unaltered if development of the address field of the instruction word is required.

After completion of the initial routine, the W-Selector Register is set to W02R, W03, W14S and W02S, in that order, and the corresponding micro-operations of these blocks are performed to execute the instruction. If the working length of the Accumulator is greater than single, the sequence of blocks W03, W14S and W02S is repeated once for each additional word of the working Accumulator.

During the W02R block of micro-operations, the memory location specified by the instruction word is cleared. During the W03 block, accumulator word A is read from Memory into the M-Register. During the W14S block, bit positions 6–23 of the M-Register are cleared, leaving character 3 of accumulator word A in the M-Register. During the subsequent W02S block, character 3 of accumulator word A is written into the character 3 position of the specified memory location. If the accumulator working length is greater than single, the Program Processor returns to block W03 to derive accumulator word B and store it in the M-Register. During the subsequent block W14S, the accumulator word undergoes a circular shift in the M-Register, under control of the N-

and Q-Registers, so that character 3 of accumulator word B is correctly positioned in the M-Register for storage in the character 2 position of the specified memory location. During the W02S cycle, the write operation for effecting the storage is completed. If the accumulator working length is triple, the sequence of blocks W03, W14S and W02S is repeated once and if the accumulator working length is quadruple, the sequence is repeated twice, to complete execution of the instruction.

The sequence of occurrence of the micro-operations required to execute Instruction 21 is illustrated in the timing diagram of FIG. 134. During time period TL0 of the W02R block of micro-operations, signal DDBX issues to transfer the address of the specified memory location through the B-Gates to the B-Register of Memory. Upon issuance of working clock signal QTCK, a read operation is initiated by signal FMRD to transfer the contents of the specified memory location into the M-Register. During time period TL1, the Q-Register is cleared. The M-Register is also cleared at this time, preparatory to receiving the operand word from Memory during the read operation. The second half-sequence of the W02 block is not performed and the operand in the specified memory location is not restored, leaving the specified memory location cleared. Upon completion of the first half-sequence of block W02, the Program Processor goes to block W03.

During time period TL0 of block W03, the address of accumulator word A is transferred from the A-Register and the Accumulator Counter Register to the B-Register of Memory by signal DABX. Flip-Flop MRD is set to the 1-state to initiate a read operation in Memory. During time period TL1, the contents of the specified memory location are cleared from the M-Register by signals DCM0–DCM3, preparatory to receiving accumulator word A from Memory. The first half-sequence of the W03 block is terminated and the second half-sequence is initiated. During the second half-sequence of the W03 block, accumulator word A is restored to Memory by a write operation initiated by signal FMWR during time period TL2. Upon completion of the second half-sequence of the W03 block, the TL-Counter and the W-Selector Register are preset to define the second half-sequence of the W14 block, i.e. block W14S.

In processing accumulator word A, circular shifting is not required during block W14S since character 3 of accumulator word A is in the correct position for storage into the character 3 position of the specified memory location, i.e. character 3 of accumulator word A is in bit positions 0–5 of the M-Register. During time period TL0 of block W14S, bit positions 6–23 of the M-Register are cleared by signals DCM0, DCM1 and DCM2. The TL-Counter and the W-Selector Register are then preset to define block W02S.

During time period TL2 of block W02S, flip-flop MWR is set to the 1-state to initiate a write operation in Memory. During time period TL4 of block W02S, the Parity Check and Generation Logic accumulates the parity of the character being stored. If the last accumulator operation is not occurring, the state of flip-flop SP2 is changed each time that the character in the M-Register contains an odd number of binary 1's. Flip-flop SP2 thus stores parity for the characters being written into the specified memory location prior to the last accumulator operation. During the last accumulator operation, the Parity Check and Generation Logic generates a parity bit based on the last character to be stored in the specified memory location and the state of flip-flop SP2. The generated parity bit is stored in the 25th bit position of the specified memory location.

During time period TL5 of the W02S block, flip-flop PSX is set to the 1-state during the last accumulator operation to transfer the address of the next instruction to the Adder. If the accumulator working length is single, the W-Selector Register is preset to define the W00 block, preparatory to execution of the next instruction. If the working length of the accumulator is greater than single, the W-Selector Register is preset to define block W03 to repeat the sequence of blocks W03, W14S and W02S. Signal DACU issues during time period TL0 to advance the count in the Accumulator Counter Register by one, to address the next accumulator word in Memory.

Assuming that the working length of the accumulator is greater than single, the Program Processor again enters block W03 to read accumulator word B from Memory into the M-Register. During the subsequent W14S block, as illustrated in FIG. 134, signal DNS0 issues during time period TL2 to reset the flip-flops of the N-Register to the 0-state. Shift Right flip-flop SRI is set to the 1-state at this time. During time period TL3, the count in the N-Register is adjusted in accordance with the accumulator word being processed. Since accumulator word B is being processed, flip-flops N00 and N01 are set to the 1-state to define the number of character position shifts required in the M-Register to put character 3 of accumulator word B in bit positions 6–11 of the M-Register for storage into the character 2 position of the specified memory location.

When working clock signal QTCK issues during time period TL3, the count in the Q-Register is advanced to one and the contents of the M-Register experience a circular right shift through one bit position. In a circular right shift, the bit shifted out of flip-flop M00 is shifted into flip-flop M23. When working clock signal QTCK issues during time period TL4, flip-flop Q01 is set to the 1-state and flip-flop Q00 is reset to the 0-state to advance the count in the Q-Register to two. The contents of the M-Register are again circular right shifted one bit position. During time period TL5, signal FSRI inhibits working clock signal QTLP to prevent the TL-Counter from advancing. The TL-Counter thus remains at TL5. With every subsequent working clock signal QTCK, the count in the Q-Register is advanced and the contents of the M-Register are circular shifted one more bit position to the right. When the Q-Register reaches a count of five, working clock signal QNCD issues to reduce by one the count in the N-Register. The count in the Q-Register is then reduced to zero and the circular right shift operation continues until the count in the N-Register equals zero. For accumulator word B, the initial count in the N-Register is three, indicating that the accumulator word B in the M-Register must be circular right shifted three character positions to put character 3 of accumulator word B in bit positions 6–11 of the M-Register. When this occurs, flip-flop SRI is reset to the 0-state, signal QTLP issues and the TL-Counter advances to TL0.

During time period TL0, bit positions 0–5 and 12–23 of the M-Register are cleared by signals DCM0, DCM1 and DCM3, leaving character 3 of accumulator word B in bit positions 6–11 of the M-Register. The Program Processor then goes to the W02S block to store character 3 of accumulator word B in the character 2 position of the specified memory location.

If the working length of the Accumulator is triple or quadruple, block W03 is repeated to read accumulator word C from Memory into the M-Register. Block W14S is then performed. Flip-flop SRI is set to the 1-state and a count of two is established in the N-Register, indicating that accumulator word C must be circular right shifted two character positions to bring character 3 of accumulator word C into bit positions 12–17 of the M-Register. The Q-Counter is advanced from a count of zero to a count of five two times and working clock signal QSHM issues twelve times to effect the circular shift. Upon completion of the shift operation, bit positions 0–11 and 18–23 of the N-Register are cleared by signals DCM0, DCM2 and DCM3, leaving character 3 of accumulator word C in bit positions 12–17. Block W02S is then performed to transfer character 3 of accumulator word C into the character 1 position of the specified memory location.

If the accumulator working length is quadruple, the sequence of blocks W03, W14S and W02S is again repeated. During block W03, accumulator word D is read from Memory into the M-Register. During block W14S, flip-flop SRI is set to the 1-state and a count of one is established in the N-Register. The Q-Register advances from a count of zero to a count of five and the word in the M-Register is circular right shifted one character position so that character 3 of accumulator word D is in bit positions 18–23 of the M-Register. Flip-flop SRI is then reset to the 0-state, and bit positions 0–17 of the M-Register are cleared by signals DCM1, DCM2 and DCM3. During the subsequent W02S block, character 3 of accumulator word D is stored in the character 0 position of the specified memory location to complete execution of Instruction 21.

Instruction 22: Shift (SHG)

This instruction causes all or part of the contents of the Accumulator to undergo a shift, the type of shift, the length of the Accumulator affected, the direction of shift and the number of positions shifted being determined by the binary digits of the instruction word address field. The shift apparatus described herein includes the invention of Messrs. Robert D. Hunter, John E. Wilhite, and Edwin W. Herron. The address field of the Shift instruction word does not refer to a memory location. The address field of the instruction word can be developed by employing auxiliary words.

A shift may be open or circular. During an open shift, characters or bits shifted out of one end of the Accumulator are lost. Binary or decimal 0's enter the other end of the Accumulator. During a circular shift, characters or bits shifted out of one end of the Accumulator are shifted into the other end and no information is lost. The length of the Accumulator affected by the shift can be single, double, triple or quadruple and is independent of the accumulator working length setting. The contents of the accumulator word locations not affected by the shift are unchanged.

Shifting of information in the Accumulator during a shift can be either by characters or by bits. Shifting by characters is effected when the Accumulator contains alphanumeric data words while shifting by bits is effected when the Accumulator contains binary data words. When the characters of alphanumeric data words in the Accumulator represent decimal numerals and the Accumulator has a sign of plus or minus, the shift is termed a decimal shift whereas if the characters of the alphanumeric data words in the Accumulator represent symbols other than decimal numerals, the shift is termed an alpha shift. During an alpha shift, all character positions are treated identically by the Shift instruction. The count field of the Shift instruction controls the number of character positions that the accumulator contents are shifted. During a decimal shift, the accumulator sign is remembered outside the Accumulator and the sign bits in the Accumulator are set to zero before shifting occurs. After the shift is completed, the remembered sign is restored (00 for plus and 10 for minus) to the accumulator sign bit positions. The count field of the instruction word controls the number of character positions that the accumulator contents are shifted. During a binary shift, all bit positions of the Accumulator are treated identically by the Shift instruction. The count field of the instruction word controls the number of bit positions that the accumulator contents are moved.

Data movement within the Accumulator during a shift can be either to the left or to the right. A right shift can be either by characters or by bits whereas a left shift can only be by characters.

Organization of the Shift instruction word is as follows:

Bit Positions:	Function
0–4	Constitute the count field, expressing in binary the number of character or bit positions through which the Accumulator is to be shifted.
5, 6	Have no effect.
7	Controls direction of shift.
8	Controls test of least-significant accumulator bit position after shift completed.
9, 10	Specify length of Accumulator affected by the shift.
11	Controls the setting of the accumulator working length to correspond to the length of the Accumulator affected by the shift.
12	Controls whether the shift is to be open or circular.
13, 14	Designates the type of shift, i.e. alpha, decimal or binary.
15–17	Address control field.
18–23	Operation code.

The following table indicates the state of each of the bits 7–14 required to specify a particular type of shift, length of Accumulator affected, direction of shift, type of data and the performance of the test and set length functions. A blank at the intersection of a column and a row indicates that either a binary zero or one may be used, without affecting the indicated function.

SHIFT INSTRUCTION WORD AND FUNCTIONS

	Alpha Decimal Binary		Shift Rotate	Set	Single Double Triple Quad.		Test	Right Left
Bit position	14	13	12	11	10	9	8	7
Open Shift			0					
Circular Shift			1					
Right								0
Left		0					0	1
Single					1	1		
Double					1	0		
Triple		0			0	1	0	
Quadruple		0			0	0	0	
Alpha	0	0					0	
Decimal	1	0					0	
Binary	0	1		0	1			0
Test		1		0	1		1	0
Set		0		1			0	

The above table indicates that, for certain functions, specific bits, other than those defining the function, must have predetermined states. For example, during a left shift, test bit 8 must be a binary zero and bit 13, defining a binary shift, must be a binary zero.

If bit position 11 of the instruction word is a binary 1, the flip-flops of the Accumulator Length Indicator Register are preset in accordance with the length of the Accumulator affected by the shift, as specified by bit positions 9 and 10. Bit position 8 of the instruction word, if it contains a binary 1, causes the least-significant bit position of the Accumulator to be tested after completion of the shift. If the least-significant bit position of the Accumulator contains a binary 0, the comparison indicator flip-flops are set to the "less than" condition; if the least-significant bit position contains a binary 1, the comparison indicator flip-flops are set to the "equal" condition. The comparison indicator flip-flops can then be tested in the

program by using either Instruction 13 (BRE) or Instruction 14 (BRL).

If the number of character or bit positions to be shifted exceeds the length of the Accumulator affected by the shift, the result is a function of the type of shift being performed. If the shift is a circular shift, the specified number of positions are shifted. If the shift is an open shift, the portion of the Accumulator affected by the shift operation will contain all zeros, except in a decimal shift when the sign is preserved if it is minus.

The shift instruction is executed by separately shifting each accumulator word affected by the shift. Shifting is accomplished in the Program Processor primarily by two registers, the E- and M-Registers, aided by other registers and logic circuits. If the number of bit or character positions through which the accumulator is to be shifted is greater than the length of a word, the M-Register is employed to shift entire accumulator words between accumulator word locations and a temporary memory location until the number of character or bit positions through which the Accumulator is to be shifted is either zero or less than the length of a word. If the number of character or bit positions through which the Accumulator is to be shifted is less than the length of a word, both the E- and M-Registers are employed to shift each accumulator word through the required number of bit or character positions. Partial words are shifted between accumulator word locations and the temporary memory location to complete the shift. In both of the above cases, if the shift is specified as open, the information in the temporary memory location is cleared; if the shift is specified to be circular, the information in the temporary memory location is stored in the last accumulator word location to be shifted.

The following chart illustrates the blocks of micro-operations which are performed to execute Instruction 22 and the sequence of performance of these blocks:

W-SEQUENCE—SHIFT INSTRUCTION

W00	$\overline{\text{DAMS}}\Delta$W15R DAMS$\Delta$W01
W01	DXIMΔW15S $\overline{\text{DXIM}}\Delta$W01
W02 R	DNEZΔW04S $\overline{\text{DNEZ}}\Delta$W15S
W02 S	
W03 R	
W03 S	W04S
W04 R	$\overline{\text{DNL4}}\Delta$W11S DNL4$\Delta$W04S
W04 S	DNEZΔW00 $\overline{\text{DVVL}}\cdot\overline{\text{DNEZ}}\Delta$W04R
W06 R	
W06 S	W00
W11 R	
W11 S	W03S
W15 R	DCCZ·DLFTΔW06S $\overline{\text{DCCZ}}$+DLFTΔW04R
W15 S	W04R

In the table, the first column identifies the blocks of micro-operations. As indicated, in certain instances only the first or the second half-sequence of a given block of micro-operations is performed. The right column, starting with the W00 block, indicates the block to be performed upon completion of the block in the left-hand column and the conditions under which the sequence of blocks occurs.

The sequence of the blocks of micro-operations for both left and right shifts is as follows. After completion of the initial routine of block W00 and block W01, if development of the instruction address field is required, the Program Processor performs block W15R. During block W15R, the contents of the flip-flops of the Accumulator Length Indicator Register are adjusted, if bit 11 of the instruction word is a binary 1, so that the working length of the Accumulator corresponds to the length of the Accumulator affected by the shift. The contents of the Accumulator Counter Register are also adjusted during block W15R to address the accumulator word A location in Memory if the shift is a right shift or to address the highest-order accumulator word affected by the shift, if the shift is a left shift. The Program Processor then goes to block W06S if the shift is a left shift and if the contents of the CC-Register are zero, indicating that no characters or bits are to be shifted. During block W06S, the address of the next instruction is applied to the Adder, preparatory to execution of the next instruction. If the shift is a right shift or if the contents of the CC-Register are not zero, the Program Processor performs the micro-operations of block W04R after completion of block W15R.

During block W04R, the count in the CC-Register is transferred to the N-Register. If the shift is a right shift, accumulator word A is read from Memory into the M-Register whereas, if the shift is a left shift, the highest accumulator word affected by the shift is read into the M-Register. At this point, the Program Processor goes to block W11S if signal DNL4 is present. Signal DNL4 indicates that the count in the N-Register is less than four if the shift is an alpha or a decimal shift, or less than twenty-four, if the shift is a binary shift.

Assuming that DNL4 issues, the Program Processor then performs the sequence of blocks W11S, W03S and W04S. During block W11S, the accumulator word in the M-Register is transferred to the E-Register and the M-Register is cleared. The contents of the E-Register are shifted to the right, the information bits shifted out of flip-flop E00 of the E-Register being shifted into flip-flop M23 of the M-Register. The contents of the M-Register are shifted in synchronism with the E-Register. The shift continues until the contents of the E-Register has been shifted through the required number of character or bit positions, as specified by the count in the N-Register. During block W03S, the count in the CC-Register is restored to the N-Register. During a right shift, the contents of the M-Register are stored in memory location 10 (octal) if accumulator word A was shifted, or is stored in the next lower-order accumulator word location relative to the accumulator word that was shifted, if the accumulator word shifted was other than accumulator word A. During a left shift, the contents of the M-Register are stored in the accumulator word location corresponding to the accumulator word shifted. During block W04S, the contents of the E-Register are stored in the accumulator word location corresponding to the accumulator word shifted, if a right shift. If a left shift, the contents of the E-Register are stored in memory location 10 (octal), if the highest-order accumulator word affected by the shift was shifted during the previous W11S block, or is stored in the next higher-order accumulator word location relative to the accumulator word shifted during the previous W11S block, if an accumulator word other than the highest accumulator word affected by the instruction was shifted during the previous W11S block.

The sequence of block W04R, W11S, W03S and W04S is repeated once for each accumulator word affected by the shift. The Program Processor then goes to block W02R to reduce the count in the N-Register to zero and to read the contents of memory location 10 (octal) into the M-Register. Block W04S is then again performed to store the

contents of the M-Register into the highest-order accumulator word affected by the shift, if the shift is a right shift, or to store the contents of the M-Register into the accumulator word A location, if the shift is a left shift. The Program Processor then goes to block W00, preparatory to execution of the next instruction.

If, during the first execution of block W04R, signal $\overline{DNL4}$ issues, the Program Processor next performs block W04S to change the location of the accumulator word read from Memory during block W04R. If the shift is a right shift, accumulator word A read from Memory into the M-Register during the previous block W04R is stored in memory location 10 (octal). If the shift is a left shift, the word read from the highest-order accumulator word location affected by the instruction is stored in memory location 10 (octal). Assuming that the length of the Accumulator affected by the shift is greater than single, the Program Processor again performs the sequence of blocks W04R and W04S to extract accumulator word B and store it in the accumulator word A location in Memory, if the shift is a right shift, or to extract the next-lower order accumulator word and store it in the next-higher accumulator word location in Memory, if the shift is a left shift. When the location of each accumulator word affected by the instruction has been changed by the performance of blocks W04R and W04S, the Program Processor goes to block W02R to read the contents of memory location 10 (octal) into the M-Register. If the count in the N-Register is zero at this time, the Program Processor then performs block W04S to store the contents of the M-Register into the highest-order accumulator word location affected by the shift, if a right shift, or into the accumulator word A location, if a left shift. The Program Processor then goes to block W00. If signal $\overline{DNEZ}$ issues to indicate that the count in the N-Register is not zero, the Program Processor goes to block W15S to store the contents of the M-Register in the highest-order accumulator word location affected by the shift, if a right shift, or into the accumulator word A location, if a left shift. The Program Processor then returns to block W04R. The sequence of blocks performed by the Program Processor following block W04R is dependent upon whether signal DNL4 or signal $\overline{DNL4}$ issues, as described above.

To summarize, three conditions may exist prior to initiating execution of Instruction 22:

(a) DNL4 issues, indicating that the total number of character or bit positions through which the Accumulator is to be shifted is less than the length of a word.

(b) signal $\overline{DNL4}$ issues, indicating that the number of character or bit positions through which the Accumulator is to be shifted is equal to or greater than the length of a word, and the number of character or bit positions equals an integral number of word lengths.

(c) signal $\overline{DNL4}$ issues and the number of character or bit positions through which the Accumulator is to be shifted does not equal an integral number of word lengths.

If condition (a) exists, execution of Instruction 22 requires the following sequence of blocks:

W00
W01 (if development of instruction address field required)
W15R
W04R
W11S
W03S
W04S
W02R
W04S

The sequence of blocks W04R, W11S, W03S and W04S is repeated once for each accumulator word affected by the shift.

If condition (b) exists, the sequence of blocks required to execute Instruction 22 is as follows:

W00
W01 (if development of instruction address field required)
W15R
W04R
W04S
W02R
W15S (if length of Accumulator affected by shift is greater than single)
W04S (if length of Accumultor affected by shift is single)

The sequence of blocks W04R and W04S is repeated once for each accumulator word affected by the instruction per integral word length to be shifted, as specified by the count in the N-Register. For example, if the length of the Accumulator affected by the shift is triple and if the shift is a decimal shift through eight character positions, the sequence of blocks W04R and W04S will be repeated six times. The connecting sequence of blocks W02R and W15S is repeated once for each integral word length, greater than one, to be shifted.

If condition (c) exists, the sequence of blocks required to execute Instruction 22 is a combination of the sequences specified for conditions (a) and (b) above. The sequence for condition (b) is followed until signal DNL4 issues, i.e. until the count in the N-Register becomes less than four, if an alpha or decimal shift, or less than twenty-four, if a binary shift. The sequence specified for condition (a) is then followed until the count in the N-Register is reduced to zero.

Illustrated below are examples of typical shift operations, each example showing the successive changes in the contents of the Accumulator and memory location 10 (octal) from the beginning of the shift operation to the end of the shift operation; also shown are the blocks of micro-operations which effect the changes.

Example 1

Decimal circular right shift through two character positions ($N=2$). Accumulator length affected by instruction=double.

Blocks	Accumulator Contents	Temporary Memory Location
Start	XXXX XXXX 1234 5678	XXXX
W04R, W11S, W03S, W04S	XXXX XXXX 1234 0056	7800
W04R, W11S, W03S, W04S	XXXX XXXX 0012 3456	7800
W02R, W04S	XXXX XXXX 7812 3456	0000

Example 2

Decimal open left shift through five character positions ($N=5$). Accumulator length affected by instruction=double.

Blocks	Accumulator Contents	Temporary Memory Location
Start	XXXX XXXX 1234 5678	XXXX
W04R, W04S	XXXX XXXX 0000 5678	1234
W04R, W04S	XXXX XXXX 5678 0000	1234
W02R, W15S	XXXX XXXX 5678 0000	0000
W04R, W11S, W03S, W04S	XXXX XXXX 6780 0000	0005
W04R, W11S, W03S, W04S	XXXX XXXX 6780 0000	0005
W02R, W04S	XXXX XXXX 6780 0000	0000

Example 3

Right binary circular shift through nine bit positions ($N=9$). Accumulator length affected=single.

Blocks	Accumulator (octal)	Temporary Memory Location (octal)
Start	XXXXXXXX XXXXXXXX XXXXXXXX 12345670	XXXXXXXX
W04R, W11S, W03S, W04S	XXXXXXXX XXXXXXXX XXXXXXXX 00012345	670XXXXX
W02R, W04S	XXXXXXXX XXXXXXXX XXXXXXXX 67012345	00000000

In Example 1, the contents of the accumulator word A and B locations are to be circular shifted right two character positions. The first sequence of blocks W04R, W11S, W03S and W04S shifts the contents of the accumulator word A location right two character positions, storing the information shifted out of the Accumulator in a temporary memory location. During the second sequence of blocks W04R, W11S, W03S and W04S, the contents of accumulator word B are shifted right two character positions, the information shifted out of the accumulator word B location being stored into the accumulator word A location. Blocks W02R, W04S are next performed to transfer the information stored in the temporary memory location into the accumulator word B location in Memory, completing the right circular shift.

In Example 2, the contents of the accumulator word A and B locations undergo an open left decimal shift through five character positions. During the first sequence of blocks W04R and W04S, accumulator word B is read from Memory and stored in a specified temporary memory location. During the second sequence of blocks W04R and W04S, accumulator word A is read from Memory and stored in the accumulator word B location. During blocks W02R and W15S, accumulator word B in the temporary memory location is cleared from the Program Processor, since the shift is an open shift. During the subsequent sequence of blocks W04R, W11S, W03S and W04S, the contents of the accumulator word B location are right shifted three character positions, which is the equivalent of the one character position left shift required by the instruction. The information shifted out of the accumulator word B location is stored in the specified temporary location. During the subsequent sequence of blocks W04R, W11S, W03S and W04S, the contents of the accumulator word A location, which are now zeros, experience a left shift through one character position. During the W02R and W04S blocks, the contents of the specified temporary memory location are cleared from the Program Processor, since the shift is an open shift, to complete execution of the shift instruction.

In Example 3, the binary information in the accumulator word A location is circular right shifted through nine bit positions. During the sequence of blocks W04R, W11S, W03S and W04S, accumulator word A is read from Memory and right shifted nine bit positions, the information shifted out of the Accumulator being stored in a temporary memory location. During the subsequent W02R and W04S blocks, the information in the temporary memory location is stored in the accumulator word A location, completing the binary right circular shift.

Preservation of the sign of the Accumulator during execution of the Shift instruction is necessary only if the shift is a decimal shift. Upon completion of the shift, the sign is restored to the Accumulator. When accumulator word A has been read from Memory into the M-Register during either a right or a left decimal shift in performing the micro-operations of block W04R, flip-flop SIN is set to the 1-state if the sign of the Accumulator is minus and if the number of character positions through which the Accumulator is to be shifted is greater than four. Signal DRE4 also issues at this time, time to force adder output signals $\overline{DS53}$ and $\overline{DS54}$ to zero, clearing the sign information from accumulator word A prior to its being shifted. If the number of character positions through which the Accumulator is to be shifted is less than four, the state of Sign flip-flop SIN is determined and signal DRE4 issues during block W11S.

The sign of the Accumulator is restored during block W04S at the point in a right shift operation when information is being stored in the accumulator word A location for the first time or at the point in a left shift operation when information is being stored in the accumulator word A location for the last time during execution of Shift instruction. At this time, signal DSS4 issues to force adder output signal DS54 to a binary 0, forcing bit 4 of character 3 in the accumulator word A location to a binary 0. If the Sign flip-flop SIN is set to the 1-state, signal $\overline{DS53}$ issues to insert a binary 1 in bit position 5 of character 3. If the flip-flop SIN is reset to the 0-state, signal

$\overline{DS55}$ is a binary 0, rendering the sign of the Accumulator plus.

Whenever a full word of twenty-four bits is stored in a memory location, a parity bit is generated and transmitted to Memory as the twenty-fifth bit of the word. During the shift operation, partial words are often stored in Memory. The parity of a partial word stored in a memory location must be remembered so that the correct parity bit may be generated when the remainder of the word is stored in Memory. The parity full adders and flip-flops SP1 and SP2 of the Parity Check and Generation Logic are employed to generate correct parity during execution of the Shift instruction.

During a right shift, flip-flop SP2 is set to the 1-state if signal MJE0 has issued from the Parity Check and Generation Logic, indicating that the partial word being stored in a memory location contains an odd number of binary 1's. When the remainder of the word is stored in the memory location, the parity information stored in the SP2 flip-flop and the number of binary 1's in the remainder of the word determines the state of the signal DM24 which provides the parity signal to Memory. Signal DM24 is a binary 1 if the number of binary 1's in the full twenty-four bit word is odd.

During a left shift, both flip-flops SP1 and SP2 are employed to store partial parity information. When a partial word is stored in a given accumulator word location, the parity of the partial word is remembered in flip-flop SP1. The state of flip-flop SP1 is then transferred to flip-flop SP2 and flip-flop SP1 is employed to remember parity for the partial word stored in the next lower-order accumulator word location. The state of flip-flop SP2 is then used, as the remainder of the word is stored in the given accumulator word location, to generate parity for the word in the given accumulator word location. When the contents of the temporary memory location are read from Memory into the M-Register during the W02R block, preparatory to storing them in the accumulator word A location if the shift is circular, the state of flip-flop SP1 is transferred to flip-flop SP2. The state of flip-flop SP2 and the information in the temporary memory location are then employed to determine the parity bit for accumulator word A, if the shift is circular. If the shift is open, only the state of flip-flop SP2 is employed to determine the parity bit for the accumulator word A location.

The sequence of occurrence of the micro-operations of each block contributing to the execution of Instruction 22 is illustrated in the timing diagram of FIG. 135. Referring to FIG. 135, block W00 is essentially the same as that illustrated in FIG. 119, with the following exceptions. During time period TL4, working clock signal QSCX issues to transfer bits 0–4 of the instruction word address field through the Adder to the CC-Register. Bits 0–4 comprise the count field specifying the number of character positions, if an alpha or decimal shift, or the number of bit positions, if a binary shift, through which the Accumulator is to be shifted. Upon completion of the W00 block, the operation code of the instruction word is stored in the I-Register, the address control field is stored in the Q-Register, and the bits 0–4 and 7–14, which control the shift, are stored in the CC- and D-Registers respectively. If development of the address field of the instruction word is required, block W01, illustrated in FIG. 121, is next performed to effect the desired modification of bits 0–4 and 7–14.

During time period TL1 of the W15R block, signal DDLX issues, if bit 11 of the instruction word is a binary 1. Signal DDLX transfers the contents of flip-flops D10 and D09 of the D-register, which specify the length of the Accumulator affected by the shift, to flip-flops PL2 and PL1 of the Accumulator Length Indicator Register, setting the working length of the Accumulator to correspond to the length of the Accumulator affected by the shift. If bit position 13 is a binary 1, indicating that the shift is a binary shift, signal DDLX cannot issue. During time period TL2 of the W15R block, signal DCAD issues, if the direction of the shift is left, to change the count in the Accumulator Counter Register to correspond to the highest-order accumulator word affected by the shift. If the direction of the shift is left and if the count in the CC-Register is zero, the Program Processor next performs block W06S to apply the address of the next instruction in the P-Register to the Adder, preparatory to performance of block W00 and execution of the next instruction. Except for this condition, the Program Processor next goes to block W04R to continue execution of Instruction 22.

During time period TL0 of block W04R, signal DABX issues during the first performance of block W04R to transfer the address of the first accumulator word to be shifted from the A-Register and the Accumulator Counter Register through the B-Gates into the B-Register of Memory. During subsequent performances of block W04R either signal DABX or signal DFBS issues to address either an accumulator word location or memory location 10 (octal). During a left shift, signal DABX issues if an accumulator word other than accumulator word A is being addressed whereas signal DFBS issues if accumulator word A is being addressed. During a right shift, signal DFBS issues only if signal $\overline{DNL4}$ has issued indicating that the count in the N-Register is greater than three, if an alpha or decimal shift, or is greater than twenty-three if a binary shift, and the last accumulator word is being shifted. Signal DABX issues under all other conditions during a right shift. Signal DRE4 issues during time period TL0 only when accumulator word A is to be read from Memory into the M-Register during a decimal shift to detect the sign of the Accumulator. Signal DRE4 forces adder input signals $\overline{DS55}$ and $\overline{DS54}$ to binary 0's so that sign information is not contained in accumulator word A after it is shifted.

Flip-flop MRD is set in response to working clock signal QTCK during time period TL0 to initiate a read operation which transfers the contents of the addressed memory location into the M-Register. Working clock signal QCNX also issues at this time to transfer the count in the CC-Register to the N-Register. During time period TL1, signals DCM0–DCM3 issue to clear the M-Register, preparatory to receiving the word from the addressed memory location during the read operation. Flip-flop MSX is also set to the 1-state during time period TL1 to apply the accumulator word in the M-Register to the adder inputs, as soon as it is received in the M-Register during the read operation. During time period TL2, flip-flop AOI is set to the 1-state to indicate that the first performance of block W04R has been completed. Signal DCAD issues during a right shift to reduce the count in the Accumulator Counter Register by one. During a left shift, signal DCAU issues to advance the count in the Accumulator Counter Register by one if the count in the N-Register is greater than three, if an alpha or decimal shift, or greater than tweny-three, if a binary shift. Block W04R is terminated and the Program Processor goes to block W11S, if signal DNL4 is present or to block W04S, if signal $\overline{DNL4}$ is present.

Considering block W04S next, during time period TL2, signal $\overline{DS55}$ issues if the sign of the Accumulator before the decimal shift was effected was minus, if the shift is a decimal shift, if the accumulator word A location in Memory is being addressed, and if signal DALS issues indicating the end of the shift operation. Signal DSS4 also issues, if the time is appropriate for storage of sign information during a decimal shift, to force bit 4 of character 3 of the word to be stored in the accumulator word A location to a binary 0. Flip-flop MWR is set to the 1-state in response to working clock signal QTCK to initiate a write operation which stores the information

in the M-Register into the correct accumulator word location in Memory. Working clock signals QSMA, QSMB and QSMC issue at this time to transfer the output signals from the Adder into the M-Register, preparatory to storing the information represented by the adder output signals in Memory during the write operation. Flip-flop SIN is set to the 1-state at this time to store the sign of a minus Accumulator during a decimal shift, if the characters of the Accumulator are to be shifted at least four character positions, if accumulator word A is in the M-Register, and if flip-flop LSN is set to the 1-state indicating that the sign of the Accumulator has not previously been stored.

During time period TL3 of the W04S block, the test function of the Shift instruction is performed if the shift has been completed and if bit position 8 of the instruction word contains a binary 1. Comparison Indicator flip-flop GRE is reset to the 0-state, if it was previously in the 1-state, if the shift is a binary right shift and if accumulator word A is being addressed. Comparison Indicator flip-flop LES is set to the 1-state under the same conditions if flip-flop M00 of the M-Register is reset to the 0-state, indicating a "less than" condition. Comparison Indicator flip-flop LES is reset to the 0-state if flip-flop M00 of the M-Register is set to the 1-state, indicating an "equal" condition.

In response to working clock signal QTCK during time period TL3, partial parity flip-flop SP2 is reset to the 0-state during a right shift if the count in the N-Register is 1–3, if an alpha or a decimal shift, or is 1–23, if a binary shift. Flip-flop LSN is also reset to the 0-state at this time if the sign of the Accumulator was stored during the previous TL2 time period. Flip-flops MSX, ESX, and CAY are reset to the 0-state. During time period TL4, partial parity flip-flop SP2 is set to the 1-state during a right shift when the count in the N-Register is 1–3, if an alpha or a decimal shift, or 1–23, if a binary shift, if signal MJE0 issues from the Parity Check Generation Logic, indicating an odd number of binary 1's in the partial word being stored in the Accumulator.

During time period TL5, flip-flop PSX is set to the 1-state, if the count in the N-Register is zero, to apply the address of the next instruction in the P-sequence to the Adder. Flip-flop RCK is reset to the 0-state stopping generation of clock pulses in the Program Processor Clock Generator. During time period TL0, signal DACU issues during a right shift if the count in the N-Register is less than four, if an alpha or decimal shift, or is less than 24, if a binary shift, to advance the count in the Accumulator Counter Register, addressing the next-higher order accumulator word to be shifted. If the shift is a left shift, signal DBYD issues to reduce the count in the Accumulator Counter Register by two, to address the next accumulator word to be shifted. If the shift is a right shift, and if the count in the N-Register is greater than three, if an alpha or a decimal shift, or is greater than 23, if a binary shift, signal DBYU issues to advance the count in the Accumulator Counter Register by two, addressing the next accumulator word to be shifted. Flip-flop A0I is reset to the 0-state at this time and the Program Processor goes to block W00, if the count in the N-Register is zero, or goes to block W02R, if signal DVVL issues, indicating that the last accumulator word has been shifted and the information stored in memory location 10 (octal) must be transferred to the Accumulator. If neither of the above conditions obtain, the Program Processor goes to block W04R to repeat performance of the W04 block.

In the W02R block, signal DFBS issues during time period TL0 to enable gate B03 of the B-Gates, addressing memory location 10 (octal). Flip-flop MRD is set to the 1-state to initiate a read operation which transfers the information stored in memory location 10 (octal) into the M-Register. Working clock signal QNCD issues at this time to reduce the count in the N-Register by four. If the count in the N-Register is less than four, if an alpha or decimal shift, or is less than 24, if a binary shift, signal QNCD reduces the count in the N-Register to zero. Partial parity flip-flop SP2 is reset to the 0-state at this time, if a left shift.

During time period TL1 of the W02R block, signals DCM0–DCM3 issue to clear the M-Register, preparatory to receiving the information from memory location 10 (octal) during the read operation. If bit position 12 of the instruction field stored in the D-Register is a binary 1, indicating that the shift is circular, flip-flop MSX is set to the 1-state at this time to apply the information from memory location 10 (octal) in the M-Register to the Adder, as soon as it is stored in the M-Register during the read operation. If the shift is a left shift, and if partial parity flip-flop SP1 was set to the 1-state during a previous W03S block, partial parity flip-flop SP2 is set to the 1-state at this time. Working clock signal QNCX issues to transfer the count in the N-Register to the CC-Register. Flip-flops ESX and RCK are reset to the 0-state, signal $\overline{\text{FRUK}}$ stopping generation of clock pulses in the Program Processor Clock Generator. Signal DCAD issues during time period TL2, if the shift is a right shift or if the count in the N-Register is zero, to reduce the count in the Accumulator Counter Register by one, addressing the highest-order accumulator word affected by the shift. If the shift is a left shift, signal DCAU issues to advance the count in the Accumulator Counter Register by one to render the count zero, addressing accumulator word A. If the count in the N-Register is zero, the Program Processor next performs block W04S, whereas if the count in the N-Register is not zero, the Program Processor goes to the W15S block.

During time period TL2 of the W15S block, signal DABX issues to transfer the address of the appropriate accumulator word from the A-Register and the Accumulator Counter Register through the B-Gates into the B-Register of Memory. Flip-flop MWR is set to the 1-state to initiate a write operation which stores the information read from memory location 10 (octal) during the previous W02R block into the appropriate accumulator word location in Memory. Working clock signals QSMA, QSMB and QSMC issue at this time to transfer the adder output signals to the M-Register, preparatory to storing the information during the write operation. During time period TL3, flip-flops ESX, MSX, PSX and CAY are reset to the 0-state while flip-flop RCK is reset to the 0-state during time period TL5. Signal DACU issues during time period TL0, if a right shift, to advance the count in the Accumulator Counter Register by one, addressing accumulator word A. If the shift is a left shift, signal DACD issues to reduce the count in the Accumulator Counter Register by one, addressing the highest-order accumulator word affected by the shift. The Program Processor then goes to the W04R block to continue the shift operation.

Block W11S, which is performed by the Program Processor after block W04R if signal DNL4 has issued indicating that the count in the N-Register is less than four, if an alpha or decimal shift, or is less than 24, if a binary shift, will next be described. Signal DRE4 issues during time period TL2 to force adder output signals $\overline{\text{D8S5}}$ and $\overline{\text{D8S4}}$ to zero during a decimal shift, if accumulator word A has been read from Memory and the sign of the Accumulator is to be stored. Sign flip-flop SIN is to set to the 1-state at this time during a decimal shift if the sign of the Accumulator is minus, if accumulator word A has been read from Memory and if the sign has not previously been stored. Working clock signal QSEX issues to transfer the accumulator word in the M-Register to the E-Register.

During time period TL3, signal DCLQ issues to clear the Q-Register and signals DCM0–DCM3 issue to clear the M-Register. Flip-flop SRI is set to the 1-state to enable the contents of the M- and E-Registers to be shifted. Flip-flop LSN is reset to the 0-state at this time to indicate that the sign of the Accumulator has been stored. Partial

parity flip-flop SP2 is reset to the 0-state if the shift is a left shift. Flip-flop MSX is reset to the 0-state, preparatory to shifting the information in the M- and E-Registers.

During time period TL4, working clock signals QSHM and QSHE issue to right shift the contents of the M- and E-Registers. Flip-flop M23 of the M-Register, which was previously cleared, is set to the 1-state if flip-flop E00 of the E-Register is set to the 1-state. In this manner, information shifted out of the least-significant bit position of the E-Register is shifted into the most-significant bit position of the M-Register. If the shift is a left shift, partial parity flip-flop SP2 is set to the 1-state at this time, if partial parity flip-flop SP1 was set to the 1-state during a previous W03S block. Flip-flop Q00 of the Q-Register is set to the 1-state to advance the count in the Q-Register to one.

During time period TL5, working clock signal QTLP is inhibited to prevent the advance of the TL-Counter. Working clock signals QSHM and QSHE continue to issue in response to each clock signal QTCK to shift the contents of the M- and E-Registers. The count in the Q-Register is advanced by one each time signals QSHM and QSHE issue, the count in the Q-Register thus indicating the number of bit positions shifted. The information shifted out of the least-significant bit position of the E-Register continues to be shifted into the most-significant bit position of the M-Register. Working clock signal QNCD issues to reduce the count in the N-Register by one for each six bit positions shifted, if the shift is an alpha or decimal shift, or to reduce the count in the N-Register by one for each bit position shifted, if the shift is a binary shift. Flip-flop SRI is reset to the 0-state during a binary shift when the count in the N-Register is reduced to one, or during an alpha or a decimal shift when the count in the N-Register is reduced to one and the count in the Q-Register is five. The delay in the output of flip-flop SRI permits one more bit position to be shifted, reducing the count in the N-Register and the Q-Register to zero. Block W11S is terminated and the Program Processor goes to block W03S to continue the shift operation.

During time period TL2 of the W03S block, signal DABX issues if the shift is a left shift, or if the shift is a right shift and an accumulator word location other than the accumulator word A location is being addressed, to transfer the address of the appropriate accumulator word location to the B-Register of Memory. If the shift is a right shift, and if the accumulator word A location is being addressed, signal DFBS issues to transfer the address of memory location 10 (octal) to the B-Register of Memory. Flip-flop MWR is set to the 1-state at this time to initiate a write operation which transfers the information in the M-Register to the addressed memory location. During time period TL3 of the W03S block, flip-flop ESX is set to the 1-state to apply the information in the E-Register to the Adder. Flip-flop SP1 is reset to the 0-state at this time, if the shift is a left shift. Working clock signal QCNX issues to transfer the count in the CC-Register to the N-Register, preparatory to the next shift operation in the M- and E-Registers. During time period TL4, partial parity flip-flop SP1 is set to the 1-state if the shift is a left shift and if signal MJE0 issues from the Parity Check and Generation Logic, indicating that the partial word in the M-Register to be stored in the Accumulator contains an odd number of binary 1's. Flip-flop RCK is reset to the 0-state and the Program Processor goes to the W04S block.

FIGURE 136 is a flow diagram of Instruction 22 illustrating the sequence of operations performed in executing Instruction 22. FIGURE 136 contains a number of shift sequences and indicates the conditions under which each shift sequence is performed. Referring to FIG. 136, the shift instruction word is read from Memory and the operation code is transferred to the I-Register. The address field which specifies the type, direction and length of the shift and which contains other information concerning the shift operation is transferred to the D-Register. Bits 1–4 of the instruction word, which represent a count defining the number of characters or bit positions to be shifted, are stored in the CC- and N-Registers. If the shift is a left shift, the contents of bit positions 0 and 1 are subtracted from four and the remainder is stored in bit positions 0 and 1 of the CC- and N-Registers. If bit positions 0 and 1 of the instruction word contain other than binary 0's, and if the shift is a left decimal shift, the accumulator is to be shifted through a number of character positions less than a full word length. The subtraction of bits and 1 from four establishes the correct number of positions through which the accumulator is right shifted in the M- and E-Registers. The direction of shift is next checked to determine a sequence to be followed. If signal DLFT has issued indicating that the shift is a left shift, and if the count in the CC-Register is not zero, the Program Processor goes to shift sequence 1 to execute the instruction. If signal $\overline{\text{DLFT}}$ has issued, indicating that the shift is a right shift, a read operation is performed transferring accumulator word A into the M-Register.

If signal $\overline{\text{DNL4}}$ has issued, indicating that the number of character or bit positions through which the Accumulator is to be right shifted is less than a word length, the Program Processor then goes to sequence 2 to execute the instruction. If the number of character or bit positions through which the Accumulator is to be shifted is greater than a word length, accumulator word A, less the sign bits if the shift is a decimal shift, is transferred to the E-Register and the M-Register is cleared. If the shift is a decimal shift, the sign of the Accumulator is stored at this time. The contents of the M- and E-Registers are then shifted right with the information shifted out of the E-Register being shifted into the M-Register. The shift continues until the information is shifted through the number of character or bit positions specified by the count in the N-Register. The partial word in the M-Register is then stored in memory location 10 (octal) and the partial word in the E-Register is stored in the accumulator word A location in Memory. Partial parity for the accumulator word A location is stored at this time. If the shift is a decimal shift, the sign is stored with the partial word in the accumulator word A location. The count in the N-Register is restored, preparatory to another shift operation in the M- and E-Registers.

If only accumulator word A is affected by the shift, and if the shift is a circular shift, the partial word from memory location 10 (octal) and the parity bit are stored in the empty character positions of the accumulator word A location in Memory to complete execution of the shift instruction. If more than one accumulator word is affected by the shift, the Program Processor goes to sequence 3. Referring to sequence 3, accumulator word B is next read from Memory and stored in the E-Register, the M-Register being cleared preparatory to the shift operation. The contents of the M- and E-Registers are shifted through the number of character or bit positions specified by the count in the N-Register. The partial word in the M-Register and the parity bit are stored in the empty character positions of the accumulator word A location and the partial word in the E-Register is restored to the accumulator word B location. Partial parity for the accumulator word B location is stored. If the accumulator length affected by the shift is triple, accumulator word C is read from Memory and shifted in the M- and E-Registers as described for accumulator words A and B. After the shift operation, the partial word in the M-Register and the parity bit are stored in the empty character positions of the accumulator word B location and the partial word in the E-Register is restored to the accumulator word C location. Partial parity for the accumulator word C location is stored. If the length of the Accumulator affected by the shift is quadruple, a similar shift operation is performed with accumulator word D. When the highest-order accumulator word affected by the shift has been

operated on, as described above, the partial word in memory location 10 (octal) is read into the M-Register. If the shift is an open shift, execution of the instruction is terminated and the Program Processor goes to block W00, preparatory to execution of the next instruction in the P-sequence. If the shift is circular, the partial word from memory location 10 (octal) and the parity bit are stored in the empty character positions of the highest-order accumulator word affected by the shift, to complete execution of the shift instruction.

Assuming that the count in the N-Register indicates that the number of character or bit positions to be shifted is greater than a full word, the Program Processor performs sequence 2. During sequence 2, accumulator word A is stored in memory location 10 (octal). If the length of the Accumulator affected by the instruction is greater than single, sequence 4 is next performed to read accumulator word B from Memory and store it in the accumulator word A location in Memory. If the length of the Accumulator affected by the shift is greater than double, accumulator word C is next read from Memory and stored in the accumulator word B location in Memory. Similarly, if the length of the Accumulator affected by the shift is quadruple accumulator word D is stored in the accumulator word C location in Memory.

In either sequence 2 or sequence 4, when the last accumulator word affected by the shift has been operated upon, the contents of memory location 10 (octal) are read into the M-Register and the count in the N-Register is reduced by four. If the shift is a circular shift, the accumulator word from memory location 10 (octal) is stored in the highest-order accumulator word location affected by the shift. If the shift is an open shift, the accumulator word from memory location 10 (octal) is cleared from the M-Register. If the count in the N-Register is zero, instruction execution is complete and the Program Processor goes to the W00 block, preparatory to execution of the next instruction. If the count in the N-Register is not zero, the Program Processor re-enters the sequence at five as indicated in FIG. 136. If the count in the N-Register still requires that the Accumulator be shifted through a number of character or bit positions greater than the length of a word, sequence 2 is again repeated. If the count in the N-Register indicates that the number of character or bit positions through which the Accumulator is yet to be shifted is less than the length of a word, the shift operation in the M- and E-Registers is performed, as previously described.

Assuming that the shift is a left shift, sequence 1 shown in FIG. 136, is performed. During sequence 1, the highest-order accumulator word affected by the shift is read into the M-Register. If the count in the N-Register indicates that the number of character positions through which the Accumulator is to be shifted is less than four, the accumulator word is transferred to the E-Register and cleared from the M-Register. The M- and E-Registers are shifted right, with the information shifted out of the least-significant bit position of the E-Register being shifted into the most-significant bit position of the M-Register. Because of the adjustment of the count in the two least-significant bits of the instruction word prior to transferring the count to the CC-Register, the number of character positions that the accumulator word is shifted right in the M- and E-Registers leaves the accumulator word in the position it would have been had it been shifted left the number of character positions specified in the original character shift count of the instruction word. The partial word in the M-Register is then stored in the highest-order accumulator word location affected by the shift while the partial word in the E-Register is stored in memory location 10 (octal). Partial parity for the highest-order accumulator word is stored. The count in the CC-Register is again transferred to the N-Register, preparatory to shifting the next lower-order accumulator word.

If the word shifted in the M- and E-Registers was not accumulator word A, sequence 7 is performed. During sequence 7, the next lower order accumulator word is read from Memory and shifted in the M- and E-Registers. Upon completion of the shift operation, the partial word in the M-Register is stored in the accumulator word location from which the shifted word was extracted while the contents of the E-Register is stored in the empty character positions of the next higher-order accumulator word location. Partial and full parity are stored, as previously described. If the word shifted was not accumulator word A, the next lower-order accumulator word is read from Memory and shifted and stored, as described above.

After accumulator word A has been shifted and the resulting partial words stored in the accumulator word A location and in the empty character positions of the accumulator word B location, the partial word in memory location 10 (octal) is read from Memory into the M-Register. If the shift is an open shift, the M-Register is cleared and the Program Processor goes to block W00 preparatory to execution of the next instruction in the P-sequence. If the shift is a circular shift, the partial word is stored in the empty character positions of the accumulator word A location. If the shift is a decimal shift, the sign is remembered and stored in the character 3 zone bits in the accumulator word A location.

Assuming, during sequence 1, that the count in the N-Register indicates that the Accumulator is to be shifted through at least four character positions, sequence 6 illustrated in FIG. 136 is performed. Left shift sequence 6 is parallel to right shift sequence 2. In sequence 6, the highest-order accumulator word affected by the shift is stored in memory location 10 (octal). If the accumulator word stored in memory location 10 (octal) is not accumulator word A, the next lower-order accumulator word is read from Memory and stored in the highest-order accumulator word location affected by the shift. If that word is not accumulator word A, the next-lower accumulator word is read from Memory and stored in the next higher-order accumulator word location.

After accumulator word A has been read from Memory and stored in the accumulator word B location, the highest-order accumulator word affected by the shift is read from memory location 10 (octal) and stored in the M-Register. If the shift is an open shift, the M-Register is cleared. If the shift is a circular shift, the highest-order accumulator word in the M-Register is stored in the accumulator word A location in Memory. The sign of the Accumulator is remembered and stored, if the shift is a decimal shift. At this point, in both sequences 6 and 8, if the count in the N-Register is zero, the Program Processor goes to the W00 block preparatory to execution of the next instruction. If the count in the N-Register is not zero, the Program Processor again goes to sequence 1.

Instruction 50: Add decimal single (ADS)

This instruction causes the contents of a specified memory location to be added to the contents of the working Accumulator. The result is placed in the Accumulator and the contents of the specified memory location are not changed. The sign of the result is placed in the zone bits of the least-significant result character, i.e. character 3; the remaining zone bits of the working Accumulator are set to zero. The accumulator length is not effected by this instruction. A carry out of the working Accumulator causes the Overflow flip-flop to be set to the 1-state. The carry is lost and the initial sign of the Accumulator is not changed.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine illustrated in FIG. 121, if development of the address field of the instruction word is required, is unaltered.

After completion of the initial routine, the W-Selector Register is set to W04R, W05, W06S if correction of

the result is necessary, and W04S, in that order, to perform the micro-operations necessary for execution of the instruction. During the W04R block of micro-operations, accumulator word A is read from Memory into the M-Register. During the W05 block, the accumulator word is stored in the E-Register and the operand word is read from Memory into the M-Register. Both the operand and accumulator word A are applied to the adder inputs to effect the addition. If the addition is an unlike-signs addition, the 10's complement of accumulator word A is added to the operand in the Adder. If the addition is a like-signs addition, or if the addition is an unlike-signs addition and a carry is generated in the character 0 sum, the W04S block is next performed to store the result in the Accumulator. If the addition is an unlike-signs addition and if no carry is propagated from the character 0 result or if the result is a minus zero, the W06S block is performed to correct the result prior to storing the corrected result during the W04S block.

The sequence of occurrence of the micro-operations required to execute Instruction 50 is illustrated in the timing diagram of FIG. 137. During time period TL0 of the W04R block, signal DABX issues to transfer the address of accumulator word A from the A-Register and the Accumulator Counter Register through the B-Gates into the B-Register of Memory. Signal DABX is present during the entire W04 block of micro-operations. Working clock signal QTCK causes flip-flop MRD to be set to the 1-state to initiate a read operation, causing accumulator word A to be transferred from Memory into the M-Register. During time period TL1, accumulator word A is applied to the Adder. Block W04R terminates as previously described, the TL-Counter and W-Selector Register being preset to define block W05.

During time period TL0 of the W05 block, signal DDBX issues to transfer the address of the operand from the D-Register through the B-Gates to the B-Register of Memory. Flip-flop MRD is set to the 1-state to initiate a read operation transferring the operand from Memory into the M-Register. At this time, accumulator word A is stored in the E-Register. During time period TL1, flip-flop DCE is set to the 1-state to enable the decimal correction circuits of the Adder, permitting the Adder to operate in the decimal mode. The M-Register is cleared at this time, preparatory to receiving the operand from Memory, unless the address of the operand and of accumulator word A are identical. If the addresses of the operand and accumulator word A are the same, accumulator word A, previously cleared from Memory, is retained in the M-Register to form the operand. At this time, flip-flop ESX is set to the 1-state to apply accumulator word A in the E-Register to the adder inputs. The first half-sequence is terminated and the second half-sequence is initiated in the conventional manner.

During time period TL2 of the second half-sequence, a memory write operation is initiated to restore the operand to Memory, unless the addresses of the operand and accumulator word A are identical. In this event, the addressed location remains cleared to receive the result of the addition. At this time, flip-flop SIN is set to the 1-state if the sign of the operand in the M-Register is minus. Working clock signal QCER issues to form the complement of accumulator word A in the E-Register if the addition is an unlike-signs addition, i.e. if the contents of the M- and E-Registers have opposite signs. If the addition is an unlike-signs addition, flip-flop LSN is reset to the 0-state and flip-flop CAY is set to the 1-state to add one to the complement of accumulator word A in the E-Register to form the 10's complement of accumulator word A. Carry Remember flip-flop CRE is reset to the 0-state during time period TL3 and the addition occurs.

During time period TL5, Decimal Correction flip-flop COR is set to the 1-state if the addition is an unlike-signs addition and if no carry was propagated from the character 0 result. Flip-flop COR is also set to the 1-state during an unlike-signs addition if the sum is zero and if Sign flip-flop SIN is set to the 1-state, since the result of minus zero is prohibited. Block W05 is terminated and the Program Processor goes to block W06S to correct the result, if flip-flop COR is set to the 1-state, or to block W04S if no correction of the result is required.

Assuming that flip-flop COR was set to the 1-state during time period TL5 of the W05 block, the W06S block of micro-operations is performed to correct the result by forming the 10's complement of the result. During time period TL2, the result of the decimal addition is transferred from the Adder to the E-Register. Flip-flop DCE is set to the 1-state during time period TL3 to cause the Adder to operate in the decimal mode. The complement of the result is then formed by complementing the contents of the E-Register. Flip-flop ESX is set to the 1-state to cause the complement of the result to be transmitted from the E-Register into the Adder and flip-flop CAY is set to the 1-state to add one to the complement in the Adder, forming the 10's complement of the result. The 10's complement constitutes the corrected result of the unlike-signs addition. Flip-flop CRE is reset to the 0-state and block W06S is terminated. The TL-Counter and the W-Selector Register are then preset to define the W04S block.

During time period TL2 of the W04S block, the address of accumulator word A is transferred to the B-Register of Memory. Signal $\overline{DS55}$ issues at this time to insert a binary 1 in bit position 5 of result character 3 if the sign of the operand was minus and no correction of the result is required or if the sign of the operand was plus and correction of the result is required. The binary digit in bit position 4 of result character 3 is always a binary 0. The issuance of signal $\overline{DS55}$ thus renders the sign of the result minus. Flip-flop MWR is set to the 1-state to initiate a write operation which stores the result in the accumulator word A location in Memory. Signals QSMA, QSMB and QSMC issue to transfer the result of the addition from the Adder to the M-Register, preparatory to transferring the result to Memory during the write operation. Flip-flop CRE is set to the 1-state at this time if a carry was propagated from character 0 of the result. During time period TL4, Overflow flip-flop FVO is set to the 1-state if flip-flop CRE is set to the 1-state and if the addition was a like-signs addition. Flip-flop PSX is set to the 1-state to transfer the address of the next instruction to the Adder. The W-Selector Register is preset to define the W00 block, preparatory to execution of the next instruction.

The above description presupposes that the working length of the Accumulator is single. If the working length of the Accumulator is double, triple or quadruple, the sequence of blocks required to execute Instruction 50 is the same as that described for Instructions 51, 52 or 53 respectively.

Instruction 60: Subtract decimal single (SDS)

This instruction causes the contents of a specified memory location to be subtracted from the contents of the working Accumulator. The result is placed in the Accumulator and the contents of the specified memory location are not changed. The sign of the result is placed in the zone bits of the least-significant result character, i.e. character 3; the remaining zone bits of the working Accumulator are set to zero. The accumulator length is not affected by this instruction. If the Accumulator and the contents of the specified memory location have unlike-signs, the result can exceed the capacity of the working Accumulator, setting the Overflow flip-flop to the 1-state. The carry is lost and the initial sign of the Accumulator is not changed.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine, illustrated in FIG. 121, is unaltered if develop-

ment of the address field of the instruction word is required.

After completion of the initial routine, the W-Selector Register is set to define blocks W04R, W05, W06S if correction of the result is necessary, and W04S, in that order, to perform the micro-operations necessary for execution of the instruction. During the W04R block of micro-operations, accumulator word A is read from Memory into the M-Register. During the W05 block, accumulator word A is stored in the E-Register and the operand word is read from Memory into the M-Register. Both the operand and accumulator word A are applied to the adder inputs to effect the subtraction. If the subtraction operation gives rise to an unlike-signs addition, the 10's complement of accumulator word A is added to the operand in the Adder. If the subtraction operation gives rise to a like-signs addition, or if the addition is an unlike-signs addition, and a carry is generated in the character 0 sum, the W04S block is next performed to store the result in the Accumulator. If the addition is an unlike-signs addition and if no carry is propagated from the character 0 result, or if the result is a minus zero, the W06S block is performed to correct the result prior to storing the corrected result during the W04S block.

The sequence of occurrence of the micro-operations required to execute Instruction 50 is illustrated in the timing diagram of FIG. 138. During time period TL0 of the W04R block, signal DABX issues to transfer the address of accumulator word A from the A-Register and the Accumulator Counter Register through the B-Gates to the B-Register of Memory. Signal DABX is present during the entire W04 block of micro-operations. Working clock signal QTCK causes flip-flop MRD to be set to the 1-state to initiate a read operation, causing accumulator word A to be transferred from Memory into the M-Register. During time period TL1, flip-flop MSX is set to the 1-state to cause accumulator word A to be applied to the Adder, as soon as it is transferred from Memory into the M-Register. Block W04R terminates, as previously described, the TL-Counter and the W-Selector Register being preset to define block W05.

During time period TL0 of the W05 block, signal DDBX issues to transfer the address of the operand from the D-Register through the B-Gates to the B-Register of Memory. Flip-flop MRD is set to the 1-state to initiate a read operation which transfers the operand from Memory into the M-Register. At this time, accumulator word A is transferred to the E-Register by signal QSEX. During time period TL1, flip-flop DCE is set to the 1-state to enable the decimal correction circuits of the Adder, permitting the Adder to operate in the decimal mode. The M-Register is cleared at this time, preparatory to receiving the operand from Memory during the read operation, unless the address of the operand and of the accumulator word A are identical. If the addresses of the operand and accumulator word A are the same, accumulator word A, previously cleared from Memory, is retained in the M-Register to form the operand. At this time, flip-flop ESX is set to the 1-state to apply accumulator word A in the E-Register to the adder inputs. The first half-sequence is terminated and the second half-sequence is initiated.

During time period TL2 of the second half-sequence, a memory write operation is initiated to restore the operand to Memory, unless the addresses of the operand and accumulator word A are identical. In this event, the addressed location remains cleared to receive the result of the subtraction. At this time, flip-flop SIN is set to the 1-state if the sign of the operand in the M-Register is plus. Working clock signal QCER issues to form the complement of accumulator word A in the E-Register if the signs of accumulator word A and the operand are the same, i.e. if the subtraction operation requires an unlike-signs addition. If the addition is an unlike-signs addition, flip-flop LSN is reset to the 0-state and flip-flop CAY is set to the 1-state to add one to the complement of accumulator word A in the Adder, forming the 10's complement of accumulator word A. Carry Remember flip-flop CRE is reset to the 0-state during time period TL3 and the subtraction occurs.

During time period TL5, Decimal Correction flip-flop COR is set to the 1-state if the addition required to effect the subtraction is an unlike-signs addition and if no carry was propagated from the character 0 result. Flip-flop COR is also set to the 1-state during an unlike-signs addition if the sum is 0 and if the Sign flip-flop SIN is set to the 1-state, since the result of minus zero is prohibited. Block W05 is terminated and the Program Processor goes to block W06S to correct the result, if flip-flop COR is set to the 1-state, or to block W04S, if no correction of the result is required.

Assuming that flip-flop COR was set to the 1-state during time period TL5 of the W05 block, the W06S block of micro-operations is performed to correct the result by forming the 10's complement of the result. During time period TL2, the result of the subtraction operation is transferred from the Adder to the E-Register. Flip-flop DCE is set to the 1-state during time period TL3 to cause the Adder to operate in the decimal mode. The complement of the result is then formed by complementing the contents of the E-Register in response to signal QCER. Flip-flop ESX is set to the 1-state to cause the complement of the result in the E-Register to be transmitted to the Adder and flip-flop CAY is set to the 1-state to add one to the complement in the Adder, forming the 10's complement of the result. The 10's complement constitutes the corrected result of the subtraction operation. Flip-flop CRE is reset to the 0-state and block W06S is terminated. The TL-Counter and the W-Selector Register are then preset to define the W04S block.

During time period TL2 of the W04S block, the address of the accumulator word A location is transferred to the B-Register of Memory by signal DABX. Signal $\overline{\text{DS55}}$ issues at this time to insert a binary 1 in bit position 5 of result character 3 if the sign of the operand was plus and no correction of the result was required, or if the sign of the operand was minus and correction of the result was required. The binary digit in bit position 4 of result character 3 is always a binary 0. The issuance of signal $\overline{\text{DS55}}$ thus renders the sign of the result minus.

Flip-flop MWR is set to the 1-state during time period TL2 of block W04S to initiate a write operation which stores the result in the accumulator word A location in Memory. Signals QSMA, QSMB and QSMC issue to transfer the result of the subtraction from the Adder to the M-Register, preparatory to transferring the result to Memory during the write operation. Flip-flop CRE is set to the 1-state at this time if a carry was propagated from character 0 of the result. During time period TL4, Overflow flip-flop FVO is set to the 1-state if flip-flop CRE is set to the 1-state and if the addition was a like-signs addition. Flip-flop PSX is set to the 1-state to transfer the address of the next instruction to the Adder. The W-Selector Register is preset to define the W00 block, preparatory to execution of the next instruction.

The above description presupposes that the working length of the Accumulator is single. If the working length of the Accumulator is double, triple or quadruple, the sequence of blocks required to execute Instruction 60 is the same as that described for Instructions 61, 62 or 63 respectively.

Instruction 54: Add to memory single (AMS)

If the working length of the Accumulator is single, this instruction causes the contents of the working Accumulator to be added to the contents of a specified memory location. The result is stored in the specified memory location and the accumulator contents and length are not changed. The sign of the result is placed in the zone bits

of the least-significant result character, i.e. character 3 in the specified memory location; the zone bits of all other characters in the specified memory location are set to zero. A carry out of the most-significant character position of the specified memory location is lost and the Overflow flip-flop is set to the 1-state. If the accumulator working length is double, triple or quadruple, the contents of the specified memory location are not changed and the Overflow flip-flop is set to the 1-state.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine, illustrated in FIG. 121, is unaltered if development of the address field of the instruction word is required.

After completion of the initial routine, the W-selector Register is set to define blocks W04R, W05, W06S if correction of the result is necessary, and W04S, in that order, to perform the micro-operations necessary for execution of the instruction. During the W04R block of micro-operations, the operand word is read from Memory into the M-Register. During the W05 block, the operand word is stored in the E-Register and accumulator word A is read from Memory to the M-Register. Both the operand and accumulator word A are applied to the adder inputs to effect the addition. If the addition is an unlike-signs addition, the 10's complement of the operand is added to accumulator word A in the Adder. If the addition is a like-signs addition, or if the addition is an unlike-signs addition, and a carry is generated in the character 0 sum, the W04S block is next performed to store the result in the Accumulator. If the addition is an unlike-signs addition and if no carry is propagated from the character 0 result, or if the result is a minus zero, the W06S block is performed to correct the result prior to storing the corrected result during the W04S block.

The sequence of occurrence of the micro-operations required to execute Instruction 54 is illustrated in the timing diagram of FIG. 139. During time period TL0 of the W04R block, signal DDBX issues to transfer the address of the operand from the D-Register through the B-Gates into the B-Register of Memory. Signal DDBX is present during the entire W04 block of micro-operations. Working clock signal QTCK causes flip-flop MRD to be set to the 1-state to initiate a read operation, which transfers the operand word from Memory into the M-Register. During time period TL1, flip-flop MSX is set to the 1-state to cause the operand word to be applied to the Adder when it is transferred from Memory into the M-Register during the read operation. Block W04R terminates as previously described, the TL-Counter in the W-Selector Register being preset to define block W05.

During time period TL0 of the W05 block, signal DABX issues to transfer the address of accumulator word A from the A-Register and the Accumulator Counter Register through the B-Gates to the B-Register of Memory. Flip-flop MRD is set to the 1-state to initiate a read operation which transfers accumulator word A to Memory into the M-Register. At this time, the operand word is transferred through the Adder into the E-Register by signal QSEX. During time period TL1, flip-flop DCE is set to the 1-state to enable the decimal correction circuits of the Adder, permitting the Adder to operate in the decimal mode. The M-Register is cleared at this time, preparatory to receiving accumulator word A from Memory during the read operation, unless the address of the operand and of accumulator word A are identical. In this event, the operand word, previously cleared from Memory, is retained in the M-Register. At this time, flip-flop ESX is set to the 1-state to apply the operand word in the E-Register to the adder inputs. The first half-sequence is terminated and the second half-sequence is initiated.

During time period TL2 of the second half-sequence of block W05, a memory write operation is initiated by signal FMWR to restore accumulator word A to Memory, unless the addresses of the operand and accumulator word A are identical. In this event, the addressed location remains cleared to receive the result of the addition. At this time, flip-flop SIN is set to the 1-state if the sign of accumulator word A in the M-Register is minus. Working clock signal QCER issues to form the complement of the operand word in the E-Register if the addition is an unlike-signs addition, i.e. if the contents of the M- and E-Registers have opposite signs. If the addition is an unlike-signs addition, flip-flop LSN is reset to the 0-state and flip-flop CAY is set to the 1-state to add one to the complement of the operand in the Adder to form the 10's complement of the operand. Carry Remember flip-flop CRE is reset to the 0-state during time period TL3 and the addition occurs.

During time period TL5, Decimal Correction flip-flop COR is set to the 1-state if the addition is an unlike-signs addition and if no carry was propagated from the character 0 result. Flip-flop COR is also set to the 1-state during an unlike-signs addition if the sum is zero and if Sign flip-flop SIN is set to the 1-state, since the result of minus zero is prohibited. Block W05 is terminated and the Program Processor goes to block W06S to correct the result, if flip-flop COR is set to the 1-state, or to block W04S if no correction of the result is required.

Assuming that flip-flop COR was set to the 1-state during time period TL5 of the W05 block, the W06S block of micro-operations is performed to correct the result by forming the 10's complement of the result. During time period TL2 of block W06S, the result of the decimal addition is transferred from the Adder to the E-Register. Flip-flop DCE is set to the 1-state during time period TL3 to cause the Adder to operate in the decimal mode. The complement of the result is then formed by signal QCER which causes the contents of the E-Register to be complemented. Flip-flop ESX is set to the 1-state to cause the complement to be transmitted from the E-Register into the Adder and flip-flop CAY is set to the 1-state to add one to the complement of the result in the Adder, forming the 10's complement of the result. The 10's complement constitutes the corrected result of the unlike-signs addition. Flip-flop CRE is reset to the 0-state and block W06S is terminated. The TL-Counter and the W-Selector Register are then preset to define the W04S block.

During time period TL2 of the W04S block, the address of the operand word is transferred to the B-Register of Memory. Signal $\overline{\text{DBSS}}$ issues at this time to insert a binary 1 in bit position 5 of result character 3, if the sign of accumulator word A was minus and no correction of the result was required, or if the sign of accumulator word A was plus and correction of the result was required. The binary digit in bit position 4 of result character 3 is always a binary 0. The issuance of signal $\overline{\text{DBSS}}$ thus renders the sign of the result minus.

Flip-flop MWR is set to the 1-state during time period TL2 to initiate the write operation which stores the result in the operand word location in Memory. Signals QSMA, QSMB and QSMC issue at this time to transfer the result of the addition from the Adder to the M-Register, preparatory to transferring the result to Memory during the write operation. Flip-flop CRE is set to the 1-state at this time if a carry was propagated from character 0 of the result. During time period TL4, Overflow flip-flop FVO is set to the 1-state if flip-flop CRE is set to the 1-state and if the addition was a like-signs addition. Flip-flop PSX is set to the 1-state during time period TL5 to transfer the address of the next instruction to the Adder. The W-Selector Register is preset to define the W00 block, preparatory to execution of the next instruction.

The above description presupposes that the working length of the Accumulator is single. If the working length of the Accumulator is double, triple or quadruple, the Overflow flip-flop FVO is set to the 1-state and the instruction is not executed.

Instruction 51: Add decimal double (ADD)

This instruction causes the contents of a two-word memory field in specified adjacent memory locations to be added to the contents of the working Accumulator. The result is placed in the Accumulator and the contents of the specified memory locations are not changed. The sign of the result is placed in the zone bits of the least-significant result character, i.e. character 3; the remaining zone bits of the working Accumulator are set to zero. If the working length of the Accumulator is double, triple or quadruple, the working length is not changed. If the working length of the Accumulator is single, the working length is set to double and the added working accumulator word is cleared to zeros before addition occurs. A carry out of the working Accumulator sets the Overflow flip-flop to the 1-state. The carry is lost and the intial sign of the Accumulator is not changed.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine illustrated in FIG. 121 is unaltered, if development of the address field of the instruction word is required.

After completion of the initial routine, the Program Processor performs the sequence of blocks W04R, W05 and W04S, in that order, twice to perform the addition. If correction of the result is necessary, the Program Processor performs the sequence of blocks W04R, W06S and W04S, in that order, twice to effect correction of the result.

During the W04 block of each addition sequence, the appropriate accumulator word is read from Memory into the M-Register. During the subsequent W05 block, the accumulator word is stored in the E-Register and the corresponding operand word is read from Memory into the M-Register. Both the accumulator word and the operand word are applied to the adder inputs to effect the addition. The W04S block is then performed to store the result in the appropriate accumulator word location in Memory. If the addition is an unlike-signs addition and if no carry is propagated from the character 0 result during the last accumulator operation, or if the result is a minus zero, the correction sequences are performed. During the W04R block of each correction sequence, the result stored in the appropriate accumulator word location is read from Memory. During the subsequent W06S block, that result is corrected and then restored to the accumulator word location during the W04S block.

The sequence of occurrence of the micro-operations required to execute Instruction 51 is illustrated in the timing diagram of FIG. 140. During time period TL0 of the W04R block, signal DABX issues to transfer the address of accumulator word A from the A-Register and the Accumulator Counter Register through the B-Gates into the B-Register of Memory. Signal DABX is present during the entire W04 block of micro-operations. Working clock signal QTCK causes flip-flop MRD to be set to the 1-state at this time to initiate a read operation, which causes accumulator word A to be transferred from Memory into the M-Register. During time period TL1, flip-flop MSX is set to the 1-state to cause accumulator word A to be applied to the Adder when it is received in the M-Register from Memory during the read operation. Block W04R terminates, the TL-Counter and the W-Selector Register being preset to define block W05.

During time period TL0 of the W05 block, signal DDBX issues to transfer the address of operand 1 from the D-Register through the B-Gates to the B-Register of Memory. In the description, operand 1 identifies the least-significant word of the memory field which is to be added to accumulator word A, the result of this addition being designated result 1; operand 2 identifies the word of the memory field which is to be added to accumulator word B, the result of this addition being designated result 2. Flip-flop MRD is set to the 1-state to initiate a read operation which effects the transfer of operand 1 from Memory into the M-Register. At this time, signal QSEX issues to store accumulator word A in the E-Register.

During time period TL1 of the W05 block, flip-flop DCE is set to the 1-state to enable the decimal correction circuits of the Adder, permitting the Adder to operate in the decimal mode. The M-Register is cleared at this time by signals DCM0–DCM3, preparatory to receiving operand 1 from Memory, unless the address of operand 1 and of accumulator word A are identical. In this event, accumulator word A, previously cleared from Memory, is retained in the M-Register to form operand 1. At this time, flip-flop ESX is set to the 1-state to apply accumulator word A in the E-Register to the adder inputs. The first half-sequence of the W05 block is terminated and the second half-sequence is initiated in the conventional manner.

During time period TL2 of the second half-sequence, a memory write operation is initiated to restore operand 1 to Memory, unless the addresses of operand 1 and accumulator word A are identical. In this event, the addressed location remains cleared to receive result 1 of the addition. At this time, flip-flop SIN is set to the 1-state if the sign of operand 1 in the M-Register is minus. Working clock signal QCER issues to form the complement of accumulator word A in the E-Register if the addition is an unlike-signs addition, i.e. if the contents of the M- and E-Registers have opposite signs. If the addition is an unlike-signs addition, flip-flop LSN is reset to the 0-state and flip-flop CAY is set to the 1-state to add one to the complement of accumulator word A in the Adder, forming the 10's complement of accumulator word A. Carry Remember flip-flop CRE is reset to the 0-state during time period TL3 and the addition occurs. Block W05 is terminated and the Program Processor goes to block W04S to store the result 1.

During time period TL2 of the W04S block, the address of accumulator word A is transferred to the B-Register of Memory. Signal $\overline{\text{DS55}}$ issues at this time to insert a binary 1 in bit position 5 of result character 3, if the sign of operand 1 was minus and no correction of the result is required, or if the sign of operand 1 was plus and correction of the result is required. The binary digit in bit position 4 of character 3 is always a binary 0. Thus, when signal $\overline{\text{DS55}}$ issues, the sign of the result is minus.

Flip-flop MWR is set to the 1-state during time period TL1 of block W04S to initiate a write operation which stores result 1 in the accumulator word A location in Memory. Signals QSMA, QSMB and QSMC issue at this time to transfer the result of the addition from the Adder to the M-Register, preparatory to transferring the result to Memory during the write operation. Flip-flop CRE is set to the 1-state at this time if a carry was propagated from character 0 of the result. The W-Selector Register is preset to define the W04R block. The count in the Accumulator Counter Register is advanced by signal DACU to address accumulator word B during the subsequent W04R block.

During the second performance of the W04R block of an addition sequence, the address of accumulator word B is transferred to the B-Register of Memory during time period TL0. Flip-flop MRD is set to the 1-state to initiate a read operation in Memory. During time period TL1, signal DDCD issues to reduce the count in the D-Register by one so that the contents of the D-Register represent the address of operand 2. When working clock signal QTCK issues during time period TL1, flip-flop MSX is set to the 1-state to transfer accumulator word B to the Adder when accumulator word B has been read from Memory to the M-Register, unless the working length of the Accumulator, as defined in the Accumulator Length Indicator Register, is single. In that event, flip-flop MSX is not set to the 1-state and accumulator word B is not applied to the Adder and is not added to operand 2; result 2 stored in the accumulator word B location in Memory is operand 2 plus a carry, if any, from result 1.

During block W05, operand 2 is read from the specified memory location into the M-Register, as described for

operand 1, and accumulator word B is transferred to the E-Register, assuming the accumulator working length to be greater than single. When working clock signal QTCK issues during time period TL1, Carry flip-flop CAY is set to the 1-state if flip-flop CRE is set to the 1-state, indicating a carry from the addition of accumulator word A and operand 1, so that the carry is added to the sum of accumulator word B and operand 2. The first half-sequence of block W05 terminates and the second half-sequence is initiated as previously described with regard to accumulator word A and operand 1.

During time period TL3 of the second half-sequence of block W05, the outputs of flip-flops IR1 and IR0 of the I-Register are transferred to the Accumulator Length Indicator Register flip-flops PL2 and PL1 to change the working length of the Accumulator to double if the working length was single. During time period TL5, Decimal Correction flip-flop COR is set to the 1-state during an unlike-signs addition if no carry was propagated from the character 0 sum or if the sum is a minus zero. The W05 block is terminated and the TL-Counter and the W-Selector Register are preset to define block W04S.

During time period TL2 of block W04S, signal D8S8 is a binary 0, so that the zone bits of result 2 are both binary 0's. A write operation is again initiated to store result 2 in the accumulator word B location in Memory. During time period TL4, Overflow flip-flop FVO is set to the 1-state if a carry has been propagated from character 0 of result 2 during a like-signs addition. During time period TL5, if flip-flop COR is reset to the 0-state indicating that no correction of the result is necessary, the address of the next instruction is transferred from the P-Register to the Adder. During time period TL0, flip-flops AC1 and AC2 of the Accumulator Counter Register are reset to the 0-state by signal DACU so that the accumulator word A location in Memory is addressed during the subsequent correction sequence if correction of the result is required. The Program Processor returns to block W04R if correction of the result is required or goes to the W00 block if no correction of the result is necessary.

The above description assumes that the working length of the Accumulator is single or double. If the working length of the Accumulator is triple, the sequence of blocks W04R, W05 and W04S is repeated so that a carry propagated from character 0 of result 2 is added to accumulator word C. Flip-flop MSX is reset during time period TL1 of the W05 block to inhibit transfer of an operand word to the Adder so that the word restored to the accumulator word C location in Memory during block W04S is the sum of accumulator word C and the carry, if any, propagated from result 2. The micro-operations occurring during the last accumulator operation, identified by DLAO, are not performed until the third addition sequence of blocks W04R, W05 and W04S. Similarly, if the working length of the Accumulator is quadruple, the sequence of blocks W04R, W05 and W04S is repeated a fourth time.

Assuming that an unlike-signs addition was performed during the preceding sequences and that no carry was propagated from character 0 of result 2, or that the sum was a minus zero, flip-flop COR is set to the 1-state and correction sequences, each comprising blocks W04R, W06S and W04S, are next performed. During block W04R of the first correction sequence, result 1 is read from the accumulator word A location in Memory into the M-Register. During the subsequent W06S block, result 1 is transferred to the E-Register during time period TL2 by working clock signal QSEX. During time period TL3, flip-flop DCE is set to the 1-state to enable the Adder to operate in the decimal mode. Working clock signal QCER issues at this time to form the complement of result 1 in the E-Register. Flip-flop ESX is set to the 1-state to apply the complement of result 1 to the Adder. Flip-flop CAY is set to the 1-state to add one to the complement of result 1 in the Adder to form the 10's complement of result 1. Block W06S is terminated and the Program Processor goes to block W04S. During block W04S, previously described, the 10's complement of result 1, which is the corrected result, is stored in the accumulator word A location in Memory.

The Program Processor then performs block W04R to read result 2 from the accumulator word B location in Memory into the M-Register. During the subsequent W06S block, the 10's complement of result 2 is formed to provide the corrected result 2 which is restored to the accumulator word B location in Memory during the W04S block which follows. At this time, flip-flop PSX is set to the 1-state to transfer the address of the next instruction to the Adder. Block W04S terminates and the W-Selector Register is preset to define the W00 block, preparatory to execution of the next instruction.

The above description of the correction sequences assumes that the accumulator working length is double or single. If the working length of the Accumulator is triple, the correction sequence of blocks W04R, W06S and W04S is repeated a third time, just as was the addition sequence of blocks W04R, W05 and W04S, to effect correction of accumulator word C. Similarly, if the working length of the Accumulator is quadruple, the correction sequence is repeated a fourth time before the Program Processor returns to the W00 block.

Instruction 52: Add decimal triple (ADT)

This instruction causes the contents of a three-word memory field in specified adjacent memory locations to be added to the contents of the working Accumulator. The result is placed in the Accumulator and the memory field is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the working Accumulator are set to zero. If the working length of the Accumulator is triple or quadruple, the length is not changed. If the working length of the Accumulator is double or single, the working length is set to triple and the one or two added working accumulator words are cleared to zeros before addition occurs. A carry out of the working Accumulator sets the Overflow flip-flop to the 1-state. The carry is lost and the initial sign of the Accumulator is not changed.

The execution of Instruction 52 is the same as that for Instruction 51 with the following exceptions. Assuming a working accumulator length less than quadruple, the addition sequence of blocks W04R, W05 and W04S is repeated three times to effect addition of accumulator words A, B and C to operands 1, 2 and 3 of the three-word memory field. The micro-operations of blocks W04R, W05 and W04S which are peculiar to the last accumulator operation, as described above with reference to the addition of accumulator word B and operand 2 during execution of Instruction 51, occur during the addition of accumulator word C and operand 3. If the accumulator working length is quadruple, the addition sequence is repeated a fourth time to add a carry, if any, to accumulator word D.

Assuming that correction of the result is necessary, the correction sequence of blocks W04R, W06S and W04S is also repeated three times if the working length of the Accumulator is triple, and four times if the working length of the Accumulator is quadruple. The micro-operations peculiar to the last accumulator operation are performed during the correction of result 3 if the accumulator working length is triple and during the correction of result 4 if the accumulator working length is quadruple.

Instruction 53: Add decimal quadruple (ADQ)

This instruction causes the contents of a four-word memory field in specified adjacent memory locations to be added to the contents of the working Accumulator. The result is placed in the Accumulator and the memory field is not changed. The sign of the result is placed in the

zone bits of the least-significant result character; the remaining zone bits of the Accumulator are set to zero. If the working length of the Accumulator is quadruple, the working length is not changed. If the working length of the Accumulator is triple, double or single, the working length is set to quadruple, and the one, two or three added working accumulator words are cleared to zeros before addition occurs. A carry out of the Accumulator sets the Overflow flip-flop to the 1-state. The carry is lost and the initial sign of the Accumulator is not changed.

The execution of Instruction 53 is the same as that for Instruction 52 with the following exceptions. The addition sequence of blocks W04R, W05 and W04S is repeated four times to effect the addition of accumulator words A, B, C and D to operands 1, 2, 3 and 4 of the four-word memory field. The micro-operations peculiar to the last accumulator operation are performed during the fourth addition sequence. Assuming that correction of the result is required, the correction sequence of blocks W04R, W06S and W04S is also repeated four times. The micro-operations peculiar to the last accumulator operation are performed during the fourth correction sequence.

Instruction 61: Subtract decimal double (SDD)

This instruction causes the contents of a two-word memory field in specified adjacent memory locations to be subtracted from the contents of the working Accumulator. The result is placed in the Accumulator and the contents of the specified memory locations are unchanged. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the working Accumulator are set to zero. If the working Accumulator is double, triple or quadruple, the working length is not changed. If the working length of the Accumulator is single, the working length is set to double and the added working accumulator word is cleared to zeros before subtraction occurs. If the Accumulator and the contents of the specified memory locations have unlike-signs, the result can exceed the capacity of the working Accumulator, setting the Overflow flip-flop to the 1-state. The carry is lost and the initial sign of the Accumulator is not changed.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine illustrated in FIG. 121 is unaltered if development of the address field of the instruction word is required.

After completion of the initial routine, the Program Processor performs the sequence of blocks W04R, W05 and W04S, in that order, twice to perform the subtraction. If correction of the result is necessary, the Program Processor performs the sequence of blocks W04R, W06S and W04S, in that order, twice to effect correction of the result.

During the W04 block of each subtraction sequence, the appropriate accumulator word is read from Memory into the M-Register. During the subsequent W05 block of a subtraction sequence, the accumulator word is stored in the E-Register and the corresponding operand word is read from Memory into the M-Register. The accumulator word and the operand are applied to the adder inputs to effect the subtraction. If the subtraction operation gives rise to an unlike-signs addition, the 10's complement of the accumulator word is applied to the Adder. The W04S block is then performed to store the result in the appropriate accumulator word location in Memory. If the subtraction operation gives rise to an unlike-signs addition and if no carry is propagated from the character 0 result during the last accumulator operation or if the result is a minus zero, the correction sequences are performed. During the W04R block of each correction sequence, the result stored in the appropriate accumulator word location is read from Memory. During the subsequent W06S block of the correction sequence, that result is corrected and then restored to the accumulator word location during the W04S block.

The sequence of occurrence of the micro-operations required to execute Instruction 61 is illustrated in the timing diagram of FIG. 141. During time period TL0 of the W04R block, signal DABX issues to transfer the address of accumulator word A from the A-Register and the Accumulator Counter Register through the B-Gates into the B-Register of the Memory. Signal DABX is present during the entire W04 block of micro-operations. Working clock signal QTCK causes flip-flop MRD to be set to the 1-state at this time to initiate a read operation which transfers accumulator word A from Memory into the M-Register. During time period TL1, flip-flop MSX is set to the 1-state to apply accumulator word A to the Adder when it is received into the M-Register from Memory during the read operation. Block W04R terminates with the TL-Counter and the W-Selector Register being preset to define block W05.

During time period TL0 of the W05 block, signal DDBX issues to transfer the address of operand 1 from the D-Register through the B-Gates to the B-Register of Memory. In the description, operand 1 identifies the least-significant word of the memory field which is to be subtracted from accumulator word A, the result being designated result 1; operand 2 identifies the word of the memory field which is to be subtracted from accumulator word B, the result being designated result 2. Flip-flop MRD is set to the 1-state to initiate a read operation which transfers operand 1 from Memory into the M-Register. At this time, signal QSEX issues to store accumulator word C in the E-Register.

During time period TL1 of the W05 block, flip-flop DCE is set to the 1-state to enable the decimal correction circuits of the Adder permitting the Adder to operate in the decimal mode. The M-Register is cleared at this time by signals DCM0–DCM3, preparatory to receiving operand 1 from Memory during the read operation, unless the address of operand 1 and of accumulator word A are identical. In this event, accumulator word A, previously cleared from Memory, is retained in the M-Register to form operand 1. At this time, flip-flop ESX is set to the 1-state to apply accumulator word A in the E-Register to the adder inputs. The first half-sequence of the W05 block is terminated and the second half-sequence is initiated.

During time period TL2 of the second half-sequence, a memory write operation is initiated to restore operand 1 to Memory, unless the addresses of operand 1 and accumulator word A are identical. In this event, the addressed location remains cleared to receive result 1 of the subtraction. At this time, flip-flop SIN is set to the 1-state if the sign of operand 1 in the M-Register is plus. Working clock signal QCER issues to form the complement of accumulator word A in the E-Register if the subtraction operation gives rise to an unlike-signs addition, i.e. if the contents of the M- and E-Registers have the same signs. If the signs of the contents of the M- and E-Registers are opposite, the subtraction operation gives rise to a like-signs addition. If the addition is an unlike-signs addition, flip-flop LSN is reset to the 0-state and flip-flop CAY is set to the 1-state to add one to the complement of accumulator word A in the Adder, forming the 10's complement of accumulator word A. Carry Remember flip-flop CRE is reset to the 0-state during time period TL3 and the subtraction occurs. Block W05 is terminated and the Program Processor goes to block W04S to store the result.

During time period TL2 of the W04S block, the address of accumulator word A is transferred to the B-Register of Memory. Signal $\overline{\text{DBSS}}$ issues at this time to insert a binary 1 in bit position 5 of result character 3 if the sign of operand 1 was plus and no correction of the result is required or if the sign of operand 1 was minus and correction is required. The binary digit in bit position 4 of character 3 is always a binary 0. Thus, when signal $\overline{\text{DBSS}}$ issues, the sign of the result is minus. Flip-flop

MWR is set to the 1-state during time period TL2 of block W04S to initiate a write operation which stores result 1 in the accumulator word A location in Memory. Signals QSMA, QSMB and QSMC issue at this time to transfer result 1 of the subtraction operation from the Adder to the M-Register, preparatory to transferring result 1 to Memory during the write operation. Flip-flop CRE is set to the 1-state at this time if a carry was propagated from character 0 of result 1. The W-Selector Register is preset to define the W04R block. The count in the Accumulator Counter Register is advanced by signal DACU to address accumulator word B during the subsequent W04R block.

During the second performance of the W04R block of a subtraction sequence, the address of accumulator word B is transferred to the B-Register of Memory during time period TL0. Flip-flop MRD is set to the 1-state to initiate a read operation in Memory. During time period TL1, signal DDCD issues to reduce the count in the D-Register by one so that the contents of the D-Register represent the address of operand 2. When working clock signal QTCK issues during time period TL1, flip-flop MSX is set to the 1-state to transfer accumulator word B to the Adder when accumulator word B has been read from Memory into the M-Register, unless the working length of the Accumulator, as defined in the Accumulator Length Indicator Register, is single. In this event, flip-flop MSX is not set to the 1-state, and accumulator word B is not applied to the Adder.

During block W05, operand 2 is read from the specified memory location into the M-Register, as described for operand 1, and accumulator word B is transferred to the E-Register, assuming the working accumulator length to be greater than single. When working clock signal QTCK issues during time period TL1, Carry flip-flop CAY is set to the 1-state if flip-flop CRE is set to the 1-state, indicating no borrow from the subtraction of operand 1 from accumulator word A. The first half-sequence of block W05 terminates and the second half-sequence is initiated as previously described with regard to accumulator word A and operand 1.

During time period TL3 of the second half-sequence of block W05, the outputs of flip-flops IR1 and IR0 of the I-Register are transferred to the Accumulator Length Indicator Register flip-flops PL2 and PL1 to change the working length of the Accumulator to double if the working length is single. During time period TL5, flip-flop COR is set to the 1-state during an unlike-signs addition if no carry was propagated from the character 0 sum or if the result is a minus zero. The W05 block is terminated and the TL-Counter and W-Selector Register are preset to define block W04S.

During time period TL2 of block W04S, signal $\overline{\text{DBSS}}$ is a binary 0, so that the zone bits of result 2 are both binary 0's. A write operation is again initiated to store result 2 in the accumulator word B location in Memory. During time period TL4, Overflow flip-flop FVO is set to the 1-state if a carry has been propagated from character 0 of result 2 during a like-signs addition. During time period TL5, if flip-flop COR is set to the 0-state indicating that no correction of the result is necessary, the address of the next instruction is transferred from the P-Register to the Adder. During time period TL0, flip-flops AC1 and AC2 of the Accumulator Counter Register are reset to the 0-state so that the accumulator word A location in Memory is addressed during the first correction sequence, if correction of the result is required. The Program Processor returns to block W04R if correction of the result is required, or goes to the W00 block if no correction of the result is necessary.

The above description assumes that the working length of the Accumulator is single or double. If the working length of the Accumulator is triple, the sequence of blocks W04, W05 and W04S is repeated so that a borrow in result 1 is propagated to accumulator word C. Flip-flop MSX is reset during time period TL1 of the W05 block to inhibit transfer of an operand word to the Adder so that the word restored to the accumulator word C location in Memory during block W04S is accumulator word C less the borrow, if any, propagated from result 2. The micro-operations occurring during the last accumulator operation, identified by signal DLA0 in the timing diagram, are not performed until the third subtraction sequence of blocks W04R, W05 and W04S. Similarly, if the working length of the Accumulator is quadruple, the sequence of blocks W04R, W05 and W04S is repeated a fourth time.

Assuming that an unlike-signs addition was performed during the preceding subtraction sequences and that no carry was propagated from character 0 of result 2, or that the result was a minus zero, flip-flop COR is set to the 1-state and correction sequences, each comprising blocks W04R, W06S and W04S, are next performed. During block W04R of the first correction sequence, result 1 is read from the accumulator word A location in Memory into the M-Register. During the subsequent W06S block, result 1 is transferred to the E-Register during time period TL2 by working clock signal QSEX. During time period TL3, flip-flop DCE is set to the 1-state to enable the Adder to operate in the decimal mode. Working clock signal QCER issues at this time to form the complement of result 1 in the E-Register. Flip-flop ESX is set to the 1-state to apply the complement of result 1 to the Adder. Flip-flop CAY is set to the 1-state to add one to the complement of the result in the Adder to form the 10's complement of result 1. Block W06S is terminated and the Program Processor goes to block W04S. During block W04S, previously described, the 10's complement of result 1, which is the corrected result, is stored in the accumulator word A location in Memory.

The Program Processor then performs block W04R to read result 2 from the accumulator word B location in Memory into the M-Register. During the subsequent W06S block, the 10's complement of result 2 is formed to provide a corrected result 2 which is restored to the accumulator word B location in Memory during the W04S block which follows. At this time, flip-flop PSX is set to the 1-state to transfer the address of the next instruction to the Adder. Block W04S terminates and the W-Selector Register is preset to define the W00 block, preparatory to the execution of the next instruction.

The above description of the correction sequences assumes that the accumulator working length is double or single. If the working length of the Accumulator is triple, the correction sequence of blocks W04R, W06S and W04S is repeated a third time, just as was the subtraction sequence of blocks W04R, W05 and W04S, to effect correction of accumulator word C. Similarly, if the working length of the Accumulator is quadruple, the correction sequence is repeated a fourth time before the Program Processor returns to the W00 block.

Instruction 62: Subtract decimal triple (SDT)

This instruction causes the contents of a three-word memory field in the specified adjacent memory locations to be subtracted from the contents of the working Accumulator. The result is placed in the Accumulator and the memory field is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the working Accumulator are set to zero. If the working length of the Accumulator is triple or quadruple, the length is not changed. If the working length of the Accumulator is double or single, the working length is set to triple, and the one or two added working accumulator words are cleared to zeros before subtraction occurs. If the Accumulator and the contents of the specified memory locations have unlike-signs, the result can exceed the capacity of the working Accumulator, setting the Overflow flip-

flop to the 1-state. The carry is lost and the initial sign of the Accumulator is not changed.

The execution of Instruction 62 is the same as that for Instruction 61 with the following exceptions. Assuming a working accumulator length less than quadruple, the sequence of blocks W04R, W05 and W04S is repeated three times to effect subtraction of the three-word memory field from accumulator words A, B and C. The micro-operations of blocks W04R, W05 and W04S which are peculiar to the last accumulator operation, as described above with reference to the subtraction of operand 2 from accumulator word B during execution of Instruction 61, occur during the subtraction of operand 3 from accumulator word C. If the accumulator working length is quadruple, the subtraction sequence is repeated a fourth time to propagate a borrow, if any, to accumulator word D.

Assuming that correction of the result is necessary, the correction sequence of blocks W04R, W06S and W04S is also repeated three times if the working length of the Accumulator is triple, and four times if the working length of the Accumulator is quadruple. The micro-operations peculiar to the last accumulator operation are performed during the correction of result 3, if the accumulator working length is triple, and during the correction of result 4, if the accumulator working length is quadruple.

Instruction 63: Subtract decimal quadruple (SDQ)

This instruction causes the contents of a four-word memory field in specified adjacent memory locations to be subtracted from the contents of the working Accumulator. The result is placed in the Accumulator and the memory field is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the Accumulator are set to zero. If the working length of the Accumulator is quadruple, the working length is not changed. If the working length of the Accumulator is triple, double or single, the working length is set to quadruple, and the one, two or three added working accumulator words are cleared to zeros before subtraction occurs. If the Accumulator and the contents of the specified memory locations have unlike-signs, the result can exceed the capacity of the Accumulator, setting the Overflow flip-flop to the 1-state. The carry is lost and the initial sign of the Accumulator is not changed.

The execution of Instruction 64 is the same as that for Instruction 63 with the following exceptions. The subtraction sequence of blocks W04R, W05 and W04S is repeated four times to effect the subtraction of operands one, two, three and four of the four-word memory field from accumulator words A, B, C and D. The micro-operations peculiar to the last accumulator operation are performed during the fourth performance of the subtraction sequence. Assuming that correction of the result is required, the correction sequence of blocks W04R, W06S and W04S is also repeated four times. The micro-operations peculiar to the last accumulator operation are performed during the fourth correction sequence.

Instruction 55: Add to memory double (AMD)

If the working length of the Accumulator is single or double, this instruction causes the contents of the working Accumulator to be added to the contents of a two-word memory field in specified adjacent memory locations. The result is stored in the specified memory locations and the accumulator contents and length are not changed. The sign of the result is placed in the zone bits of the least-significant result character in the specified memory locations; the zone bits of all other characters in the specified memory locations are set to zero. A carry out of the most-significant result is lost and the Overflow flip-flop is set to the 1-state. If the working length is triple or quadruple, the contents of the specified memory locations are not changed and the Overflow flip-flop is set to the 1-state.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine, illustrated in FIG. 121, is unaltered if development of the address field of the instruction word is required.

After completion of the initial routine, the Program Processor performs the sequence of blocks W04R, W05 and W04S, in that order, twice to perform the addition. If correction of the result is necessary, the Program Processor performs the sequence of blocks W04R, W06S and W04S, in that order, twice to effect correction of the result.

During the W04 block of each addition sequence, the appropriate operand word is read from Memory into the M-Register. During the subsequent W05 block, the operand word is stored in the E-Register and the corresponding accumulator word is read from Memory into the M-Register. Both the accumulator word and the operand word are applied to the adder inputs to effect the addition. The W04S block is then performed to store the result in the appropriate operand word location in Memory. If the addition is an unlike-signs addition, and if no carry is propagated from the character 0 result during the last accumulator operation, or if the result is a minus zero, the correction sequences are performed. During the W04R block of each correction sequence, the result stored in the appropriate operand word location is read from Memory. During the subsequent W06S block, that result is corrected and the corrected result is then restored to the operand word location during the W04S block which follows.

The sequence of occurrence of the micro-operations required to execute Instruction 55 is illustrated in the timing diagram of FIG. 142. During time period TL0 of the W04R block, signal DDBX issues to transfer the address of operand 1 from the D-Register through the B-Gates into the B-Register of the Memory. In the description, operand 1 identifies the least-significant word of the memory field to which accumulator word A is to be added, the result being designated result 1; operand 2 designates the word of the memory field to which accumulator word B is to be added, the result being designated result 2. Signal DDBX is present during the entire W04 block of micro-operations. Working clock signal QTCK causes flip-flop MRD to be set to the 1-state at this time to initiate a read operation, which transfers operand 1 from Memory into the M-Register. During time period TL1, flip-flop MSX is set to the 1-state to apply operand 1 to the Adder when operand 1 is received in the M-Register from Memory during the read operation. Block W04R terminates with the TL-Counter and W-Selector Register being preset to define block W05.

During time period TL0 of the W05 block, signal DABX issues to transfer the address of accumulator word A from the A-Register and the Accumulator Counter Register through the B-Gates to the B-Register of Memory. Flip-flop MRD is set to the 1-state to initiate a read operation which transfers accumulator word A from Memory into the M-Register. At this time, signal QSEX issues to store operand 1 in the E-Register.

During time period TL1 of the W05 block, flip-flop DCE is set to the 1-state to enable the decimal correction circuits of the Adder, permitting the Adder to operate in the decimal mode. The M-Register is cleared at this time, preparatory to receiving accumulator word A from Memory during the read operation, unless the address of operand 1 and accumulator word A are identical. In this event, operand 1, previously cleared from Memory, is retained in the M-Register. At this time, flip-flop ESX is set to the 1-state to apply operand 1 in the E-Register to the adder inputs. The first half-sequence of the W05 block is terminated and the second half-sequence is initiated.

During time period TL2 of the second half-sequence, a memory write operation is initiated to restore accumulator word A to Memory, unless the addresses of operand 1 and accumulator word A are identical. In this event, the

addressed location remains cleared to receive the result of the addition. At this time, flip-flop SIN is set to the 1-state if the sign of accumulator word A in the M-Register is minus. Working clock signal QCER issues to form the complement of operand 1 in the E-Register if the addition is an unlike-signs addition, i.e. if the contents of the M- and E-Registers have opposite signs. If the addition is an unlike-signs addition, flip-flop LSN is reset to the 0-state and flip-flop CAY is set to the 1-state to add one to the complement of operand 1 in the Adder, forming the 10's complement of operand 1. Carry Remember flip-flop CRE is reset to the 0-state during time period TL3 and the addition occurs. Block W05 is terminated and the Program Processor goes to block W04S to store result 1.

During time period TL2 of the W04S block, the operand 1 address is transferred to the B-Register of Memory. Signal $\overline{D855}$ issues at this time to insert a binary 1 in bit position 5 of result character 3 if the sign of the accumulator word A was minus and no correction of the result is required, or if the sign of the accumulator word A was plus and correction of the result is required. The binary digit in bit position 4 of character 3 is always a binary 0. Thus, when signal $\overline{D855}$ issues, the sign of the result is minus.

Flip-flop MWR is set to the 1-state during time period TL2 of block W04S to initiate a write operation which stores result 1 in the operand 1 location in Memory. Signals QSMA, QSMB and QSMC issue at this time to transfer result 1 from the Adder to the M-Register, preparatory to transferring result 1 to Memory during the write operation. Flip-flop CRE is set to the 1-state at this time if a carry was propagated from character 0 of result 1. During time period TL5, signal DDCD issues to decrease the contents of the D-Register by one to form the address of accumulator word B. The W-Selector Register is preset to define the W04R block. The count in the Accumulator Counter Register is advanced by signal DACU to address accumulator word B during the next addition sequence.

During the second performance of the W04R block of an addition sequence, the address of operand 2 is transferred from the D-Register to the B-Register of Memory during time period TL0. Flip-flop MRD is set to the 1-state to initiate a read operation in the Memory. During time period TL1, the M-Register is cleared, preparatory to receiving operand 2 from Memory during the read operation. Flip-flop MSX is set to the 1-state to transfer operand 2 to the Adder, when operand 2 has been received in the M-Register from Memory during the read operation.

During block W05, accumulator word B is read from Memory into the M-Register, as described for accumulator word A, and operand 2 is transferred to the E-Register. When working clock signal QTCK issues during time period TL1, Carry flip-flop CAY is set to the 1-state if flip-flop CRE is set to the 1-state, indicating a carry from the addition of accumulator word A and operand 1, so that the carry is added to the sum of accumulator word B and operand 2. Flip-flop MSX is reset to the 0-state at this time to inhibit the transfer of accumulator word B from the M-Register to the Adder, if the addressed accumulator word is outside the working Accumulator, i.e. if the accumulator working length is single. The first half-sequence of block W05 terminates and the second half-sequence is initiated.

During time period TL5 of the second half-sequence of block W05, Decimal Correction flip-flop COR is set to the 1-state during an unlike-signs addition if no carry was propagated from the character 0 sum or if the sum is a minus zero. The W05 block is terminated and the TL-Counter and the W-Selector Register are preset to define block W04S.

During time period TL2 of block W04S, signal $\overline{D855}$ is a binary 0, so that the zone bits of result 2 are both binary 0's. A write operation is again initiated to store result 2 in the operand 2 location in Memory. During time period TL4, Overflow flip-flop FVO is set to the 1-state if a carry has been propagated from character 0 of result 2 during a like-signs addition. If correction of the result is necessary, flip-flop DSX is set to the 1-state at this time to transfer the address in the D-Register to the Adder. During time period TL5, signal FDSX issues and the contents of flip-flops IR1 and IR0 of the I-Register are used to modify the D-Register address in the Adder so that the contents of the D-Register again represent the address of the operand 1 location in Memory. If flip-flop COR is reset to the 0-state at this time, indicating that no correction of the result is necessary, flip-flop PSX is set to the 1-state to transfer the address of the next instruction from the P-Register to the Adder. The Program Processor returns to block W04R if correction of the result is required or goes to the W00 block if no correction of the result is necessary.

The above description assumes that the working length of the Accumulator is single or double. If the working length of the Accumulator is triple or quadruple, Overflow flip-flop FVO is set to the 1-state and the instruction is not executed.

Assuming that an unlike-signs addition was performed during the preceding addition sequences and that no carry was propagated from character 0 of result 2, or that the sum was a minus zero, flip-flop COR is set to the 1-state and the correction sequences, each comprising blocks W04R, W06S and W04S, are next performed. During block W04R of the first correction sequence, result 1 is read from the operand 1 location in Memory into the M-Register. During the subsequent W06S block, result 1 is transferred to the E-Register during time period TL2 by working clock signal QSEX. During time period TL3 of block W06S, flip-flop DCE is set to the 1-state to enable the Adder to operate in the decimal mode. Working clock signal QCER issues at this time to form the complement of result 1 in the E-Register. Flip-flop ESX is set to the 1-state to apply the complement of result 1 to the adder. Flip-flop CAY is set to the 1-state to add one to the complement in the Adder to form the 10's complement of result 1. Block W06S is terminated and the Program Processor goes to block W04S to store the 10's complement of result 1, which is the correct result, in the operand 1 location in Memory.

The Program Processor then again performs block W04R to read result 2 from the operand 2 location in Memory into the M-Register. During the subsequent W06S block, the 10's complement of result 2 is formed to provide the corrected result 2, which is restored to the operand 2 location in the Memory during the W04S block which follows. At this time, flip-flop COR is reset to the 0-state. Flip-flop PSX is set to the 1-state to transfer the address of the next instruction to the Adder. Block W04S terminates and the W-Selector Register is preset to define the W00 block, preparatory to execution of the next instruction.

Instruction 56: Add to memory triple (AMT)

If the working length of the Accumulator is single, double or triple, this instruction causes the contents of the working Accumulator to be added to the contents of a three-word memory field in specified adjacent memory locations. The result is stored in the specified memory locations. The accumulator contents and length are not changed. The sign of the result is placed in the zone bits of the least-significant result character; the zone bits of all other characters of the result are set to zero. A carry out of the most-significant result is lost and the Overflow flip-flop is set to the 1-state. If the accumulator length is quadruple, the contents of the specified memory locations are not changed and the Overflow flip-flop is set to the 1-state.

The execution of Instruction 56 is the same as that

for Instruction 55 with the following exceptions. Assuming a working accumulator length less than quadruple, the addition sequence of blocks W04R, W05 and W04S is repeated three times to effect addition of accumulator words A, B and C to operands 1, 2 and 3 of the three-word memory field. The micro-operations of blocks W04R, W05 and W04S which are peculiar to the last accumulator operation, as indicated by signal DLAO in the timing diagram and as described above with reference to the addition of accumulator word B and operand 2 during execution of Instruction 55, occur during the third addition sequence when accumulator word C is added to operand 3. Signal DDCD issues during the second addition sequence to form the address of operand 3 in the D-Register. Assuming that correction of the result is necessary, the correction sequence of blocks W04R, W06 and W04S is also repeated three times. The micro-operations peculiar to the last accumulator operation are performed during the correction of result 3.

Instruction 57: Add to memory quadruple (AMQ)

This instruction causes the contents of the working Accumulator to be added to the contents of a four-word memory field in specified adjacent memory locations. The result is stored in the specified memory locations and the accumulator contents and length are not changed. The sign of the result is placed in the zone bits of the least-significant result character; the zone bits of all other characters in the specified memory locations are set to zero. A carry out of the most-significant result is lost and the Overflow flip-flop is set to the 1-state.

The execution of Instruction 57 is the same as that for Instruction 56 with the following exceptions. The addition sequence of blocks W04R, W05 and W04S is repeated four times to effect the addition of accumulator words A, B, C and D to operands 1, 2, 3 and 4 of the four-word memory field. The micro-operations peculiar to the last accumulator operation are performed during the fourth addition sequence. Signal DDCD issues during the third addition sequence to form the address of operand 4 in the D-Register. Assuming that correction of the result is required, the correction sequence of blocks W04R, W06S and W04S is also repeated four times. The micro-operations peculiar to the last accumulator operation are performed during the fourth correction sequence.

Instruction 34: Add binary to memory (ABM)

This instruction causes the contents of a first specified memory location to be added absolutely, in binary, to the contents of a second specified memory location. The result is placed in the second memory location. The contents of the first memory location are not changed. The address of the first memory location may be developed by employing auxiliary words while the address of the second memory location is specified to be the address of a Fixed Location Index Word or is developed by employing SAS auxiliary words. A carry out of the most-significant bit position of the sum is lost and the Overflow flip-flop is set to the 1-state.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine, illustrated in FIG. 121, is unaltered during development of the address of the first specified memory location, if required.

After development of the first address of the instruction, if required, the W-Selector Register is set to W02 and the W02 block of micro-operations is performed by the Program Processor to initiate execution of the instruction. The Program Processor then returns to block W01 to develop the second address of the instruction. The W-Selector Register is next set to W15 and the W15S block of micro-operations is performed to complete execution of the instruction.

During the W02 block of micro-operations, after the initial routine is completed, operand 1 is read from the first specified memory location and stored in the E-Register. In the description, operand 1 identifies the word in the first specified memory location while operand 2 identifies the word in the second specified memory location. During the W01 block, after development of the second address, operand 2 from the second specified memory location is added to operand 1. During block W15S, the result of the addition is stored in the second specified memory location.

The sequence of occurrence of the micro-operations required to execute Instruction 34 is illustrated in the timing diagram of FIG. 143. During time period TL0 of W02 block of mirco-operations, signal DDBX issues to transfer the address of the first memory location from the D-Register through the B-Gates into the B-Register of Memory. When working clock signal QTCK issues during time period TL0, flip-flop MRD is set to the 1-state, to initiate a read operation which transfers operand 1 from Memory into the M-Register. During time period TL1, the M-Register is cleared by signals DCM0–DCM3, preparatory to receiving operand 1 from Memory during the read operation. Working clock signal QTCK during time period TL1 sets flip-flop MSX to the 1-state to cause operand 1 to be applied to the Adder as soon as it is received in the M-Register from Memory during the read operation. Flip-flop RCK is reset to the 0-state at this time and the first half-sequence of block W02 is terminated.

Upon issuance of synchronizing signal DMDA from Memory, the second half-sequence of block W02 is initiated. Flip-flop MWR is set to the 1-state to cause a write operation to be performed in Memory which restores operand 1 to the first memory location. During time period TL4, working clock signal QSEX issues to transfer operand 1 through the Adder into the E-Register. Flip-flop MSX is reset to the 0-state during time period TL5 and flip-flop RCK is also reset to the 0-state to terminate the second half-sequence of block W02. Flip-flops WR0 and WR1 of the W-Selector Register are set and reset respectively to define block W01. The Program Processor then returns to block W01 to develop the second address of the instruction, i.e. the address of the second specified memory location.

The performance of block W01 in developing the second address is the same as that illustrated in FIG. 121 with the following exceptions. During time period TL2, signal DVCA issues when the second address has been developed to inhibit restoration of operand 2 into the second memory location after it has been read into the M-Register. Flip-flop MSX is set to the 1-state at this time to apply operand 2 to the Adder. Flip-flop ESX is also set to the 1-state at this time, upon completion of development of the second address, to apply operand 1 in the E-Register to the Adder. The addition of operand 1 and operand 2 is performed at this time. During time period TL0 of the W01 block, the flip-flops of the TL-Counter and the W-Selector Register are preset to define block W15S.

During time period TL2 of block W15S, either signal DDBX or signal DQBX issues, depending upon the value of the address control field of the instruction word, to transfer the address of the second memory location from either the D-Register or the Q-Register to the B-Register of Memory. Flip-flop CRE is set to the 1-state at this time, if a carry was propagated from the most-significant character of the sum. Flip-flop MWR is set to the 1-state to initiate a write operation in Memory which stores the result in the second specified memory location. Working clock signals QSMA, QSMB and QSMC issue at this time to transfer the result from the adder output terminals into the M-Register, preparatory to storing the result in Memory during the write operation. This transfer clears operand 2 from the M-Register. During time period TL4,

Overflow flip-flop FVO is set to the 1-state, if flip-flop CRE was set to the 1-state. During time period TL5, flip-flop PSX is set to the 1-state to transfer the address of the next instruction to the Adder. Flip-flop RCK is reset to the 0-state to stop the Program Processor Clock Generator until synchronizing signal DMBU is transmitted by Memory. During time period TL0, flip-flops WR3, WR2, WR1 and WR0 are reset to the 0-state to define the W00 block, preparatory to execution of the next instruction.

Instruction 35: Subtract binary from memory (SBM)

This instruction causes the contents of a first specified memory location to be subtracted absolutely, in binary, from the contents of a second specified memory location. The result is placed in the second specified memory location. The contents of the first specified memory location are not changed. The address field of the instruction word may provide the address of the first specified memory location or this address may be developed by employing auxiliary words. The address of the second specified memory location is specified to be the address of a Fixed Location Index Word or is developed by employing SAS auxiliary words. In the absolute subtraction, if the value of the contents of the first specified memory location exceeds the value of the contents of the second specified memory location, the result (in 2's complement form) is placed in the second specified memory location and the Overflow flip-flop is set to the 1-state.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine, illustrated in FIG. 121, is unaltered if development of the address field of the instruction word is required.

After development of the first address of the instruction in the W01 block, if required, the W-Selector Register is set to W02 and the micro-operations of the W02 block are performed by the Program Processor to initiate execution of the instruction. The Program Processor then returns to block W01 to develop the second address of the instruction. The micro-operations of block W15S are next performed to complete execution of the instruction.

During the W02 block of micro-operations, after the initial routine is completed, operand 1 in the first specified memory location is read from Memory and stored in the E-Register. In the description, operand 1 identifies the word in the first specified memory location while operand 2 identifies the word in the second specified memory location. The micro-operations of the W01 block effect development of the address of the second specified memory location and apply operand 1 and operand 2 to the Adder to perform the subtraction. During the subsequent W15S block, the result of the subtraction is stored in the second specified memory location.

The sequence of occurrence of the micro-operations required to execute Instruction 35 is illustrated in the timing diagram of FIG. 143. During time period TL0 of the W02 block of micro-operations, signal DDBX issues to transfer the address of the first memory location from the D-Register through the B-Gates into the B-Register of Memory. Working clock signal QTCK sets flip-flop MRD to the 1-state to initiate a read operation in Memory which transfers operand 1 into the M-Register. During time period TL1, the flip-flops of M-Register are cleared, preparatory to receiving operand 1 from Memory during the read operation. Flip-flop DSX is reset to the 0-state at this time. Working clock signal QTCK causes flip-flop MSX to be set to the 1-state to apply operand 1 in the M-Register to the Adder as soon as operand 1 is received in the M-Register from Memory during the read operation. Flip-flop ESX is reset to the 0-state at this time and flip-flop RCK is also reset to the 0-state to terminate the first half-sequence of block W02.

The second half-sequence of the W02 block of micro-operations is initiated when synchronizing signal DMDA is provided by Memory to start the Program Processor Clock Generator. At this time, operand 1 is transferred from the Memory Data Drivers into the M-Register. Working clock signal QTCK during time period TL2 sets flip-flop MWR to the 1-state to initiate a write operation in Memory, which restores operand 1 to the first memory location. During time period TL4, working clock signal QSEX issues to transfer operand 1 from the M-Register through the Adder into the E-Register. During time period TL5, working clock signal QCER issues to complement the contents of the E-Register, forming the 1's complement of operand 1. The second half-sequence of the W02 block is terminated and the flip-flops of the W-Selector Register are preset to define block W01.

During the W01 block, the second address of Instruction 35 is specified to be the address of a Fixed Location Index Word or is developed by employing a P-Sequence SAS Word and Remote SAS Words, if necessary. The timing and micro-operations of the W01 block are essentially the same as that illustrated in FIG. 121, with the following exceptions. After development of the second address, signal DVCA prevents flip-flop MWR from being set to the 1-state during time period TL2, thereby inhibiting restoration of operand 2 in the M-Register into the second specified memory location. Flip-flop MSX is set to the 1-state at this time to transfer operand 2 from the M-Register to the adder inputs. Flip-flop ESX is also set at this time to apply the 1's complement of operand 1 in the E-Register to the adder inputs. During time period TL3, Carry flip-flop CAY is set to the 1-state to add one to the 1's complement of operand 1 in the Adder, forming the 2's complement of operand 1. The subtraction is performed at this time with the result appearing at the adder output terminals. During time period TL0, the flip-flops of the TL-Counter and the W-Selector Register are preset to define block W15S.

During time period TL2 of block W15S, either signal DDBX or signal DBQX issues, depending upon the value of the address control field of the instruction word, to transfer the address of the second specified memory location from either the D-Register or the Q-Register to the B-Register of Memory. Carry Remember flip-flop CRE is set to the 1-state at this time if no carry was propagated from the most-significant character of the result of the subtraction. Flip-flop MWR is set to the 1-state at this time to initiate a write operation in Memory which stores the result in the second specified memory location. Working clock signals QSMA, QSMB and QSMC issue to transfer the result from the Adder to the M-Register, preparatory to storing the result in Memory during the write operation. This transfer clears operand 2 from the M-Register. During time period TL4, Overflow flip-flop FVO is set to the 1-state if flip-flop CRE was set to the 1-state during time period TL2. During time period TL5, flip-flop PSX is set to the 1-state to transfer the address of the next instruction to the Adder. Block W15S is terminated and the flip-flops of the W-Selector Register are preset to define the W00 block, preparatory to execution of the next instruction.

Instruction 33: Add Immediate to memory (AMI)

This instruction causes the address field of the instruction word to be added in binary to the address field of an operand word in a second specified memory location. The operation code and the address control field of the operand word in the second specified memory location are not changed. The address field of the instruction word may be developed by employing auxiliary words. The address of the second specified memory location is specified to be the address of a Fixed Location Index Word or is defined by employing SAS auxiliary words. A carry out of the sum of the address fields is lost and does not cause the Overflow flip-flop to be set to the 1-state.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine, illustrated in FIG. 121, is unaltered if develop-

ment of the address field of the instruction word is required. During development of the address of the second memory location, the W01 block of FIG. 121 is altered as follows. During time period TL2 after development of the address of the second memory location has been completed, signal DVCA prevents flip-flop MWR from being set to the 1-state, inhibiting the restoration of the operand word read from the second memory location so that the second memory location remains cleared, as illustrated in FIG. 144. At this time, flip-flop MSX is set to the 1-state to apply the operand in the M-Register to the adder inputs. Flip-flop ESX is also set to the 1-state at this time to apply the instruction word in the E-Register to the adder inputs to effect addition of the address field of the instruction word to the address field of the operand.

After completion of the initial routine, the flip-flops of the TL-Counter and the W-Selector Register are preset to define block W15S and the micro-operations of block W15S are performed to complete execution of the instruction. During the W15S block, the address field of the operand in the M-Register is replaced by the sum of the address fields from the Adder and the operand is stored in the second memory location.

The sequence of occurrence of the micro-operations required to execute Instruction 33 is illustrated in the timing diagram of FIG. 144. During time period TL2 of block W15S, the address of the second specified memory location is transferred from the D-Register or the Q-Register, depending upon the value of the address control field of the instruction word, to the B-Register of Memory. Flip-flop MWR is set to the 1-state at this time to initiate a write operation which stores the operand word with its new address field in the second specified memory location. Working clock signals QSMA and QSMB issue at this time to transfer the sum of the address fields of the instruction word and of the operand from the adder output terminals into bit positions 0–14 of the M-Register. This transfer clears the old address field of the operand from the M-Register. The operand with the new address field is then stored in the second specified memory location during the write operation. During time period TL5, flip-flop PSX is set to the 1-state to transfer the address of the next instruction in the P-sequence to the Adder. Flip-flop RCK is reset to the 0-state to terminate block W15S. The flip-flops of the W-Selector Register are reset to the 0-state to define the W00 block, preparatory to execution of the next instruction.

Instruction 03: Compare alphanumeric accumulator to memory (CAA)

This instruction causes the contents of the working Accumulator to be compared with the contents of a memory field, the least-significant word of which is in a specified memory location. The length of the memory field corresponds to the length of the working Accumulator. The comparison is based upon the absolute binary contents of the working Accumulator and the memory field. The contents of the Accumulator and the memory field can be either binary or alphanumeric. The result of the comparison (working Accumulator less than, equal to or greater than the contents of the memory field) is placed in the comparison indicator flip-flops GRE and LES. The accumulator contents and length and the memory field contents are not changed.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. If development of the address of the specified memory location is required, the W01 block illustrated in FIG. 121 is unaltered.

After completion of the initial routine, the W-Selector Register is preset to define blocks W08 and W09, in that order, and the W08 and W09 blocks of micro-operations are performed to execute the instruction. During the W08 block of micro-operations, an accumulator word is read from Memory and stored in the E-Register where the 1's complement of the accumulator word is formed, the 1's complement being applied to the adder inputs. During the W09 block of micro-operations, the corresponding operand of the memory field is read from Memory and added to the 2's complement of the accumulator word in the Adder. The presence of a carry from the result when the result is not equal to zero or the absence of a carry from the result determines the state of the comparison indicator flip-flops. If the working length of the Accumulator is double, the Program Precessor performs the comparison sequence of blocks W08 and W09 twice to execute the instruction, the next higher-order words of the Accumulator and the memory field being compared during each successive performance. The last state of the comparison indicator flip-flops determines the result of the comparison. If the working length of the Accumulator is triple or quadruple, the comparison sequence of blocks W08 and W09 is repeated a total of three or four times respectively to execute the instruction.

The sequence of occurrence of the micro-operations required to execute Instruction 03 is illustrated in the timing diagram of FIG. 145. During time period TL0 of the W08 block of micro-operations, signal DABX issues to transfer the address of accumulator word A to Memory. Signal DABX issues to transfer the address of accumulator word A from the A-Register and the Accumulator Counter Register through the B-Gates into the B-Register of Memory. Working clock signal QTCK sets flip-flop MRD to the 1-state at this time to initiate a read operation in Memory which transfers accumulator word A into the M-Register. During time period TL1, signals DCM0–DCM3 issue to clear the M-Register, preparatory to receiving accumulator word A from Memory during the read operation. Flip-flop MSX is set to the 1-state in response to working clock signal QTCK to transfer accumulator word A from the M-Register to the Adder when it is received in the M-Register during the read operation. Flip-flops ESX and RCK are reset to the 0-state at this time, signal FTCK causing the first half-sequence of the W08 block to terminate.

The second half-sequence of the W08 block is initiated by synchronizing signal DMDA from Memory, which starts the Program Processor Clock Generator. Working clock signal QTCK sets flip-flop MWR to the 1-state at this time to initiate a write operation in Memory which restores accumulator word A. During time period TL3, signal QSEX issues to transfer accumulator word A through the Adder into the E-Register. During time period TL4, signal QCER issues in response to working clock signal QTCK to complement the contents of the E-Register, forming the 1's complement of accumulator word A. Flip-flop MSX is reset to the 0-state to inhibit the transfer of accumulator word A in the M-register to the adder inputs. During time period TL5, flip-flop RCK is reset to the 0-state to terminate the second half-sequence of the W08 block. During time period TL0, flip-flop WR0 of the W-Selector Register is set to the 1-state, causing the Program Processor to go to block W09.

During time period TL0 of the W09 block, signal DDBX issues to transfer the address of the specified memory location which contains the least-significant word of the memory field into the B-Register of Memory. In the description, operand 1 identifies the least-significant word in the memory field while operands 2, 3, and 4 designate progressively more-significant words of the memory field. Flip-flop MRD is set to the 1-state in response to working clock signal QTCK to initiate a read operation in Memory which transfers operand 1 into M-Register. During time period TL1, comparison indicator flip-flops GRE and LES are reset to the 0-state and signals DCM0–DCM3 issue to clear the M-Register, preparatory to receiving operand 1 from Memory during the read operation. Flip-flop ESX is set to the 1-state in response to signal QTCK to apply the 1's complement of accumu-

lator word A in the E-Register to the adder inputs. Carry flip-flop CAY is set to the 1-state at this time to add one to the 1's complement of accumulator word A in the Adder to form the 2's complement of accumulator word A. Flip-flop MSX is also set to the 1-state at this time to apply operand 1 and the M-Register to the adder inputs, effecting the addition of operand 1 and the 2's complement of accumulator word A. The first half-sequence of block W09 is terminated and the second half-sequence is initiated by signal DMDA from Memory.

Flip-flop MWR is set to the 1-state during time period TL2 of the second half-sequence to initiate a write operation in Memory to restore operand 1. During time period TL4, a borrow during the subtraction of accumulator word A from operand 1, indicating that accumulator word A is greater than operand 1, causes flip-flop GRE to be set to the 1-state and flip-flop LES to be reset to the 0-state. Conversely, a carry from the most-significant character of the result, indicating that the absolute binary value of accumulator word A is less than that of operand 1, causes flip-flop LES to be set to the 1-state and flip-flop GRE to be reset to the 0-state, if the result is not equal to zero. If the result is equal to zero, both of the comparison indicator flip-flops GRE and LES remain in the 0-state to indicate the equality of accumulator word A and operand 1.

If the working length of the Accumulator is single, the state of the comparison indicator flip-flops at this time indicates the result of the comparison, i.e. whether accumulator word A is greater than, equal to, or less than operand 1. Flip-flop PSX is set to the 1-state during time period TL5 to transfer the address of the next instruction to the Adder. Signal DACU issues to advance the count in the Accumulator Counter Register, addressing accumulator word B. Flip-flops CAY, MSX and ESX are reset to the 0-state. Flip-flop RCK is reset to the 0-state to terminate the second half-sequence of the W09 block and the W-Selector Register is preset to define block W00, preparatory to execution of the next instruction.

Assuming that the working length of the Accumulator is double, flip-flop PSX is not set to the 1-state and the W-Selector Register and the Program Processor return to block W08 to compare accumulator word B and operand 2 in another comparison sequence of blocks W08 and W09. During block W08 of the second comparison sequence, accumulator word B is read from Memory and transferred to the E-Register. The 1's complement of accumulator word B is formed in the E-Register and applied to the adder inputs. During time period TL1 of block W08 of the second comparison sequence, signal DDCD issues to decrease the count in the D-Register by one, forming the address of operand 2.

During block W09 of the second comparison sequence, operand 2 is read from Memory into the M-Register and applied to the adder inputs. Accumulator word B is subtracted from operand 2 and the presence of a carry when the result is not equal to zero or the absence of a carry from the result again determines the states of the comparison indicator flip-flops GRE and LES. The W09 block is terminated and the Program Processor returns to the W00 block with the comparison indicator flip-flops containing the result of the comparison.

If the working accumulator length is triple, the comparison sequence of blocks W08 and W09 is repeated a third time to subtract accumulator word C from operand 3, the presence of a carry from the result when the result is not equal to zero or the absence of a carry from the result controlling the state of the comparison indicator flip-flops. If the accumulator working length is quadruple, the comparison sequence is repeated a fourth time to subtract accumulator word D from operand 4.

Instruction 02: Compare decimal accumulator to memory (CDA)

This instruction causes the contents of the working Accumulator to be compared algebraically with the contents of a memory field, the least-significant word of which is stored in a specified memory location. The length of the memory field corresponds to the length of the working Accumulator. The only zone bits considered in the comparison are the zone bits of character 3 of accumulator word A and the zone bits of character 3 of the least-significant operand word, which establish the signs of the Accumulator and the memory field respectively. When the signs are compared, a minus sign (zone bits=10) is considered less than all three plus sign configurations (zone bits=00, 01 or 11). The result of the decimal comparison (contents of the Accumulator less than, equal to or greater than the contents of the memory field) is placed in the comparison indicator flip-flops GRE and LES. The accumulator contents and length and the memory field contents are not changed.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine, illustrated in FIG. 121, is unaltered, if address development is required.

After completion of the initial routine, the W-Selector Register is preset to define blocks W08 and W09, in that order, and the W08 and W09 blocks of micro-operations are performed by the Program Processor to execute the instruction. The comparison sequence of blocks W08 and W09 is repeated once for each word of the working Accumulator. During the W08 block of each comparison sequence, the appropriate operand word is extracted from Memory and stored in the E-Register, the complement of the Accumulator word being formed in the E-Register and applied to the adder inputs. During the subsequent W09 block of each comparison sequence, the appropriate operand word is read from Memory into the M-Register and applied to the adder inputs, the operand being added to the 2's complement of the accumulator word to effect a subtraction of the accumulator word from the operand. The state of the comparison indicator flip-flops is controlled by the signs of the operand and Accumulator or by the presence of a carry when the result is not equal to zero or the absence of a carry from the result of the subtraction.

The sequence of occurrence of the micro-operations required to execute Instruction 02 is illustrated in the timing diagram of FIG. 146. During time period TL0 of the W08 block of micro-operations, signal DABX issues to transfer the address of accumulator word A from the A-Register and the Accumulator Counter Register through the B-Gates into the B-Register of Memory. Flip-flop MRD is set in response to working clock signal QTCK to initiate a read operation in Memory which transfers accumulator word A into the M-Register. During time period TL1, the M-Register is cleared, preparatory to receiving accumulator word A during the read operation. Flip-flop MSX is also set to the 1-state at this time to transfer accumulator word A to the adder inputs, as soon as it is received in the M-Register from Memory during the read operation. Flip-flops ESX and RCK are reset to the 0-state at this time, signal FRCK terminating the first half-sequence of the W08 block.

During time period TL2 of the second half-sequence of the W08 block, flip-flop MWR is set to the 1-state to initiate a write operation in Memory which restores accumulator word A to Memory. During time period TL3, working clock signal QSEX issues to transfer accumulator word A through the Adder into the E-Register. During time period TL4, signal QCER issues to complement the contents of the E-Register, forming the 1's complement of the accumulator word A. Flip-flop MSX is reset to the 0-state and flip-flop ESX is set to the 1-state to apply the 1's complement of accumulator word A in the E-Register to the adder inputs. Block W08 is terminated and the Program Processor goes to block W09.

During time period TL0 of block W09, signal DDBX issues to transfer the address of the specified memory

location, which contains the least-significant word of the memory field, to the B-Register of Memory. Flip-flop MRD is set to the 1-state at this time to initiate a read operation which transfers operand 1 from the specified memory location into the M-Register. In the description, operand 1 designates the least-significant word of the memory field while operands 2, 3 and 4 identify words of progressively greater significance in the memory field. During time period TL1, flip-flop DCE is set to the 1-state to enable the decimal correction circuits of the Adder, permitting the Adder to operate in the decimal mode. Comparison indicator flip-flops GRE and LES are reset to the 0-state at this time. Signals DCM0–DCM3 issue to clear the M-Register, preparatory to receiving operand 1 from Memory during the read operation. Flip-flop ESX is set to the 1-state in response to working clock signal QTCK to apply the 1's complement of accumulator word A in the E-Register to the adder inputs. Carry flip-flop CAY is also set to the 1-state at this time to add one to the 1's complement of accumulator word A in the Adder, forming the 2's complement of accumulator word A. Flip-flop MSX is set to the 1-state at this time to apply operand 1 in the M-Register to the adder inputs, thereby effecting the addition of operand 1 and the 2's complement of accumulator word A, i.e. the subtraction of accumulator word A from operand 1. The first half-sequence of block W09 terminates and the second-half sequence is initiated by synchronizing signal DMDA from Memory.

During time period TL2 of the second half-sequence of the W09 block, flip-flop MWR is set to the 1-state to initiate a write operation in Memory which restores operand 1 to the specified memory location. Sign flip-flop SIN is set to the 1-state at this time if the sign of operand 1 is minus. Like-signs flip-flop LSN is reset to the 0-state at this time if operand 1 and accumulator word A have opposite signs. During time period TL4, comparison indicator flip-flop GRE is set to the 1-state and flip-flop LES is reset to the 0-state to indicate that accumulator word A is greater than operand 1, if accumulator word A and operand 1 are both positive and a borrow is propagated from the result, or if operand 1 is negative, the result is not equal to zero and a carry is propagated from the result. If the result is equal to zero, both comparison indicator flip-flops GRE and LES remain in the 0-state to indicate the equality of accumulator word A and operand 1. Flip-flop LES is set to the 1-state and flip-flop GRE is reset to the 0-state to indicate that accumulator word A is less than operand 1, if operand 1 is positive and accumulator word A is negative, if both operand 1 and accumulator word A are negative and a borrow is propagated from the result, or if operand 1 is positive, a carry is propagated from the result and the result is not equal to zero.

During time period TL5 of the second half-sequence of block W09, flip-flop DCE is reset to the 0-state. Flip-flop PSX is set to the 1-state at this time to transfer the address of the next instruction to the Adder if the signs of operand 1 and accumulator word A are opposite or if the working length of the Accumulator is single. Signal DACU issues at this time to advance the count in the Accumulator Counter Register, addressing accumulator word B. Flip-flops CAY, MSX and ESX are reset to the 0-state. Flip-flop RCK is reset to the 0-state to terminate the second half-sequence. If operand 1 and accumulator word A have unlike-signs or if the accumulator working length is single, the flip-flops of the W-Selector Register are reset to the 0-state and the Program Processor goes to block W00, preparatory to execution of the next instruction. If operand 1 and accumulator word A have like-signs and if the accumulator working length is greater than single, the W-Selector Register is preset to define block W08 and the comparison sequence of blocks W08 and W09 is repeated.

Assuming that the accumulator working length is double, the Program Processor returns to block W08 to read accumulator word B from Memory and stores it in the E-Register. The 1's complement of accumulator word B is formed in the E-Register and applied to the adder inputs. Signal DDCD issues during time period TL1 of the W08 block to reduce the count in the D-Register by one, forming the address of operand 2.

During block W09, operand 2 is read from Memory into the M-Register and accumulator word B is subtracted from operand 2 in the Adder. During time period TL4 of the W09 block, the states of comparison indicator flip-flops GRE and LES are determined by the presence of a carry when the result is not equal to zero or the absence of a carry from the result and by the sign of the memory field and the Accumulator. Block W09 is terminated with the comparison indicator flip-flops containing the result of the comparison.

Assuming that the accumulator working length is triple, the comparison sequence of blocks W08 and W09 is repeated a third time. Similarly, if the accumulator working length is quadruple, the comparison sequence of blocks W08 and W09 is repeated a fourth time to determine the result of the comparison of the working Accumulator and the memory field.

Instruction 04: Compare second to first memory (CMM)

This instruction causes the contents of a first specified memory location and the contents of a second specified memory location to be compared. The comparison is based upon the absolute binary value of the contents of the two memory locations, although the contents can either be binary or alphanumeric. The result of the comparison (contents of the second location greater than, equal to or less than the contents of the first location) is placed in the comparison indicator flip-flops GRE and LES. The contents of the first and second specified memory locations are unchanged. The address field of the instruction word may provide the address of the first specified memory location or this address may be developed by employing auxiliary words. The address of the second specified memory location is specified to be the address of a Fixed Location Index Word or is developed by employing SAS auxiliary words.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. If development of the address of the first specified memory location is required, the W01 block illustrated in FIG. 121 is unaltered.

After completion of the W00 block and development of the address of the first specified memory location in the W01 block, if required, the W-Selector Register is preset to define blocks W02, W01 and W09, in that order. The micro-operations corresponding to these blocks are performed to develop the address of the second specified memory location and to execute the instruction. During the W02 block of micro-operations, operand 1 is read from Memory and stored in the E-Register. In the description, operand 1 identifies the contents of the first specified memory location, while operand 2 designates the contents of the second specified memory location. During the W01 block, the address of the second specified memory location is developed and operand 2 is read from Memory into the M-Register. During the subsequent W09 block, operand 1 is subtracted from operand 2 in the Adder and the states of the comparison indicator flip-flops are controlled by the result of the subtraction to indicate the result of the comparison.

The sequence of occurrence of the micro-operations required to execute Instruction 04 is illustrated in the timing diagram of FIG. 147. During time period TL0 of the W02 block of micro-operations, signal DDBX issues to transfer the address of the first memory location from the D-Register through the B-Gates into the B-Register of Memory. Flip-flop MRD is set to the 1-state in response to working clock signal QTCK to initiate a read operation in Memory, which transfers operand 1 from the first

memory location into the M-Register. During time period TL1, signals DCM0–DCM3 issue to clear the M-Register, preparatory to receiving operand 1 from Memory during the read operation. Flip-flop MSX is set to the 1-state at this time to transfer operand 1 from the M-Register to the Adder when operand 1 is received in the M-Register from Memory during the read operation. The first half sequence of the W02 block is terminated and the second half-sequence is initiated by synchronizing signal DMDA from Memory.

During time period TL2 of the second half-sequence of the W02 block, flip-flop MWR is set to the 1-state to initiate a write operation in Memory, which restores operand 1 to the first memory location. During time period TL4, working clock signal QSEX issues to transfer operand 1 through the Adder into the E-Register. During time period TL5, flip-flop PSX is set to the 1-state to apply the address of the next P-sequence word to the Adder. Flip-flop MSX is reset to the 0-state and the second half-sequence of block W02 is terminated. Flip-flops WR0 and WR1 of the W-Selector Register are set and reset respectively to define the W01 block.

During the W01 block, the address of the second specified memory location is developed, as illustrated in the timing diagram of FIG. 121, and operand 2 in the second specified memory location is read from Memory into the M-Register. The Program Processor then goes to block W09 to complete execution of the instruction.

During time period TL1 of block W09, comparison indicator flip-flops GRE and LES are reset to the 0-state. Working clock signal QCER issues at this time to complement the contents of the E-Register, forming the 1's complement of operand 1 in the E-Register. Flip-flop ESX is set to the 1-state to apply the 1's complement of operand 1 in the E-Register to the adder inputs. Carry flip-flop CAY is also set to the 1-state to add one to the 1's complement of operand 1 in the Adder, forming the 2's complement of operand 1. Flip-flop MSX is set to apply operand 2 in the M-Register to the adder inputs, to effect the addition of operand 2 and the 2's complement of operand 1, i.e. to subtract operand 1 from operand 2. The first half-sequence of block W09 is terminated and the second half-sequence in initiated by synchronizing signal DMDA from Memory.

During time period TL4 of the second half-sequence, comparison indicator flip-flop GRE is set to the 1-state to indicate that operand 2 is greater than operand 1, if the result of the substraction of operand 1 from operand 2 is not equal to zero and if a carry is propagated from the result. If no carry, i.e. a borrow, is propagated from the result, flip-flop LES is set to the 1-state, indicating that operand 2 is less than operand 1. During time period TL5, flip-flop PSX is set to the 1-state to transfer the address of the next instruction word to the Adder. Block W09 is then terminated and the W-Selector Register is preset to define blocks W00, preparatory to execution of the next instruction. The result of the comparison remains in the comparison indicator flip-flops GRE and LES.

Instruction 01: Compare memory to Immediate (CMI)

This instruction causes the address field of the instruction word to be compared with the address field of an operand word in a second specified memory location. The comparison is based upon the absolute binary contents of the two address fields. The result of the comparison (address field of the operand word in the second memory location greater than, equal to or less than the address field of the instruction word) is placed in the comparison indicator flip-flops GRE and LES. The instruction word and the contents of the second memory location are unchanged. The address field of the instruction word may be developed by employing auxiliary words. The address of the second memory location is specified to be the address of a Fixed Location Index Word or is developed by employing SAS auxiliary words.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. If development of the address field of the instruction word is required, the W01 block, illustrated in FIG. 121, is unaltered. Development of the address of the second specified memory location is effected in accordance with the micro-operations of the W01 block, as illustrated in FIG. 121.

After completion of the initial routine, the Program Processor performs the W09 block of micro-operations to complete execution of the instruction. During the W09 block, the address field of the instruction word is subtracted from the address field of the operand in the second memory location and the states of the comparison indicator flip-flops are controlled by the result of the subtraction, indicating the comparison result.

The sequence of occurrence of the micro-operations required to execute Instruction 01 is illustrated in the timing diagram of FIG. 148. During time period TL1 of the W09 block, comparison indicator flip-flops GRE and LES are reset to the 0-state, preparatory to performance of the comparison. Working clock signal QCER issues at this time in response to signal QTCK to complement the instruction word in the E-Register, forming the 1's complement of the instruction address field. Flip-flop ESX is set to the 1-state at this time to apply the 1's complement of the instruction address field in the E-Register to the adder inputs. Carry flip-flop CAY is set to the 1-state to add one to the 1's complement of the instruction address field in the Adder to form the 2's complement of the instruction address field. Flip-flop MSX is set to the 1-state to apply the address field of the operand from the second memory location to the adder inputs. The 2's complement of the instruction address field is thus added to the address field of the operand to effect a subtraction of the instruction address field from the operand address field. The first half-sequence of the W09 block is terminated and the second half-sequence is initiated by signal DMDA from Memory.

During time period TL4 of the second half-sequence of block W09, comparison indicator flip-flop GRE is set to the 1-state in response to signal $\overline{MC14}$, indicating a carry from bit position 14 of the result, and signal $\overline{DSE5}$, indicating that the sum of the 2's complement to the instruction address field and the operand address field is not equal to zero. Flip-flop LES remains set to the 0-state under the same conditions in response to the same signals, indicating that the address field of the operand is greater than the instruction address field. If signal DSE5 issues, indicating that the result of the arithmetic operation is zero, both comparison indicator flip-flops remain in the 0-state to indicate the equality of the instruction address field and the operand address field. If signal MC14 issues, indicating that no carry was propagated from bit position 14 of the result, flip-flop LES is set to the 1-state and flip-flop GRE remains reset to the 0-state to indicate that the address field of the operand is less than the instruction address field.

During time period TL5 of the W09 block, flip-flop PSX is set to the 1-state to transfer the address of the next instruction word to the Adder. The second half-sequence of the W09 block is terminated and the flip-flops of the W-Selector Register are preset to define the W00 block, preparatory to execution of the next instruction.

Instruction 10: Branch unconditionally (BRU)

This instruction causes the contents of the P-Register to be replaced by the instruction address field, causing a program branch to the location specified by the instruction address field. The instruction address field may be developed by employing auxiliary words.

The Program Processor executes the instruction during the performance of block W00 of the initial routine, if address development is not required, or during block W01 of the initial routine, if address development is required. The micro-operations required to execute Instruction 10 are illustrated in the timing diagram of FIG. 149.

Referring to FIG. 149, the W00 block of the initial routine, illustrated in FIG. 119, is altered to execute the instruction if, address development is not required, as follows. During time period TL4 of the W00 block, working clock signal QSPX issues to transfer the address field of the instruction word through the Adder to the P-Register, replacing the contents of the P-Register. The Program Processor then returns to block W00, preparatory to execution of the next instruction.

If development of the address field of the instruction word is required, the W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine, illustrated in FIG. 121, is altered as indicated in FIG. 149 to execute the instruction. During time period TL5, after address development is completed, working clock signal QSPX issues to transfer the developed address field of the instruction word through the Adder into the P-Register, replacing the contents of the P-Register. The Program Processor then returns to block W00, preparatory to execution of the next instruction.

Instruction 17: Store program counter and branch (SPB)

This instruction causes the contents of the P-Register to be stored in a specified memory location and the address field of the instruction to be placed in the P-Register, to effect a branch to the memory location specified by the instruction address field. The address field of the instruction may be developed by employing auxiliary words. The address of the specified memory location is specified to be the address of a Fixed Location Index Word or is developed by employing SAS auxiliary words. Only bits 0–14 of the word in the specified memory location are changed.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. If development of the address field of the instruction word is required, the W01 block illustrated in FIG. 121 is unaltered. During development of the address of the specified memory location, the W01 block of FIG. 121 is altered as follows. During time period TL2 signal DVCA, which indicates that development of the address of the specified memory location is completed, inhibits the restoration of the word in the second location by preventing flip-flop MWR from being set to the 1-state, as illustrated in FIG. 150. Signal FNXF prevents flip-flop ESX from being set to the 1-state at this time, inhibiting the transfer of the instruction word in the E-Register to the Adder. During time period TL5, flip-flop PSX is set to the 1-state, upon completion of development of the address of the specified memory location, to transfer the contents of the P-Register to the Adder. Flip-flop MSX is reset to the 0-state at this time to inhibit the transfer of the instruction word in the M-Register to the Adder. During time period TL0, the flip-flops of the TL-Counter and the W-Selector Register are preset to define block W15S.

Upon completion of the initial routine, described above, the Program Processor performs the W15S block of micro-operations to execute the instruction. During the W15S block, the contents of the P-Register are stored in the specified memory location. During the W00 block of the initial routine of the next instruction, the address field of the instruction word is transferred to the P-Register.

The sequence of occurrence of the micro-operations required to execute Instruction 17 is illustrated in the timing diagram of FIG. 150. During time period TL2 of the W15S block, the address of the specified memory location is transferred from the D-Register or the Q-Register, depending upon the value of the address control field of the instruction word, through the B-Gates into the B-Register of Memory. Flip-flop MWR is set to the 1-state at this time to initiate a write operation, which stores the contents of the P-Register in the specified memory location. Working clock signals QSMA and QSMB issue at this time to transfer the contents of the P-Register stored in the E-Register through the Adder to bit positions 0–14 of the M-Register. This transfer clears the old address field from the M-Register. The word in the M-Register with the P-Register contents in bit positions 0–14 is then stored in the specified memory location during the write operation. During time period TL5, flip-flop ESX is set to the 1-state to apply the instruction address field in the E-Register to the Adder. Block W15S is terminated and the Program Processor goes to block W00, preparatory to execution of the next instruction.

During the W00 block of the initial routine of the next instruction, working clock signal QSPX issues during time period TL0 to transfer the address field of the instruction word stored in the E-Register through the Adder into the P-Register. Signal QSDX, which normally issues at this time during block W00, transfers the address field of the instruction word to the D-Register to effect the branch to the location specified by the instruction address field.

Instruction 12: Branch if greater (BRG)

This instruction causes the contents of the P-Register to be replaced by the address field of the instruction if the specified condition exists in the comparison indicator flip-flops, i.e. if flip-flop GRE is set to the 1-state, causing a program branch to the location specified by the instruction address field. If the specified condition does not exist in the comparison indicator flip-flops, the branch is not made and the next instruction is taken in normal sequence. Execution of the instruction does not change the states of the comparison indicator flip-flops.

The W00 block of the initial routine, illustrated in FIG. 119, is altered as indicated in FIG. 151 to execute the instruction if address development is not required. With reference to FIG. 151, flip-flop MSX is set to the 1-state during time period TL2 of the W00 block to apply the instruction word in the M-Register to the Adder. During time period TL4, working clock signal QSPX issues if comparison indicator flip-flop GRE is set to the 1-state, transferring the address field of the instruction word through the Adder into the P-Register. This transfer clears the former contents from the P-Register. Block W00 is terminated and the Program Processor branches to the location specified by the address field of the instruction word.

If development of the address field of the instruction word is required, the W00 block of the initial routine, as illustrated in FIG. 119, is unaltered. The W01 block of the initial rountine, illustrated in FIG. 121, is altered as indicated in FIG. 151. With reference to FIG. 151, working clock signal QSPX issues, after development of the address field of the instruction is completed, if flip-flop GRE is set to the 1-state, transferring the developed address field of the instruction word through the Adder into the P-Register. Block W01 is terminated and the Program Processor then branches to the location specified by the developed address field of the instruction word. If the GRE flip-flop is not set to the 1-state, the branch is not made and the address of the next instruction is taken in the normal P-sequence.

Instruction 13: Branch if equal (BRE)

This instruction causes the contents of the P-Register to be replaced by the address field of the instruction word if the specified condition exists in the comparison indicator flip-flips, i.e. if flip-flops GRE and LES are both reset to the 0-state, causing a program branch to the location specified by the instruction address field. If the specified condition does not exist, the branch is not made and the next instruction is taken in normal sequence. Execu-

tion of the instruction does not change the states of the comparison indicator flip-flops.

The execution of Instruction 13 is identical to the execution of Instruction 12 with the following exception. Working clock signal QSPX issues during either the W00 or W01 block only if flip-flops GRE and LES are both reset to the 0-state, satisfying the condition specified by the instruction word.

Instruction 14: Branch if less (BRL)

This instruction causes the contents of the P-Register to be replaced by the address field of the instruction word if the specified condition exists in the comparison indicator flip-flops, i.e., if flip-flop LES is set to the 1-state, causing a program branch to the location specified by the instruction address field. If the specified condition does not exist, the branch is not made and the next instruction is taken in normal sequence. Execution of the instruction does not change the states of the comparison indicator flip-flops.

The execution of Instruction 14 is identical to the execution of Instruction 12 with the following exception. Working clock signal QSPX issues during either the W00 or W01 block only if flip-flop LES is set to the 1-state, as indicated in the timing diagram of FIG. 151.

Instruction 15: Branch if zero (BRZ)

This instruction causes the contents of the P-Register to be replaced by the address field of the instruction word and a branch taken to the memory location specified by the instruction address field, if all working accumulator bits are zero. If any bits of the working Accumulator are not zero, the next instruction in the P-sequence is executed.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine, illustrated in FIG. 121, is unaltered if address development is required.

After completion of the initial routine, the W-Selector Register is preset to W02 and the Program Processor performs the micro-operations of block W02 to execute the instruction. During the W02 block of micro-operations, an accumulator word is extracted from Memory and the bits of the accumulator word are checked for binary 1's. Block W02 is repeated until each word of the working Accumulator has been checked or until a binary 1 is detected in any bit position of the working Accumulator, at which time the Program Processor returns to block W00. If no binary 1's are detected in the working Accumulator, the address field of the instruction word in the D-Register is transferred to the P-Register and the Program Processor branches to the memory location specified by the instruction address field.

The sequence of occurrence of the micro-operations required to execute Instruction 15 is illustrated in the timing diagram of 152. During time period TL0 of the W02 block of micro-operations, the address of accumulator word A is transferred from the A-Register and the Accumulator Counter Register through the B-Gates into the B-Register of Memory. Flip-flop MRD is set to the 1-state at this time to initiate a read operation in Memory which transfers accumulator word A into the M-Register. During time period TL1, signals DCM0–DCM3 issue to clear the M-Register, preparatory to receiving accumulator word A from Memory during the read operation. Flip-flop DSX is reset to the 0-state at this time. Flip-flop MSX is set to the 1-state in response to working clock signal QTCK to apply accumulator word A in the M-Register to the Adder, as soon as it is received in the M-Register during the read operation. The first half-sequence of the W02 block is terminated and the second half-sequence is initiated.

During time period TL2 of the second half-sequence, flip-flop MWR is set to the 1-state to initiate a write operation in Memory which restores accumulator word A. During time period TL3, flip-flop RNZ is set to the 1-state if any of the bits of accumulator word A, as sensed at the adder output terminals, are a binary 1. Assuming that the working length of the Accumulator is single, flip-flop DSX is set to the 1-state during time period TL5 if flip-flop RNZ is reset to the 0-state, i.e. if the bits of accumulator word A are all binary 0's. Signal FDSX causes the address field of the instruction word, stored in the D-Register, to be applied to the Adder. During the W00 block of the initial routine of the next instruction, signal QSPX issues during time period TL0 to transfer the instruction address field through the Adder to the P-Register, to effect a branch to the memory location corresponding to the address. Flip-flop PSX is set to the 1-state if accumulator word A does not contain all binary 0's to apply the address of the next instruction in the P-sequence to the Adder. The second half-sequence of the W02 block is terminated and the Program Processor goes to block W00, preparatory to execution of the next instruction.

Assuming that the working length of the Accumulator is double, the Program Processor repeats block W02 if accumulator word A contained no binary 1's. During the second performance of block W02, accumulator word B is read from Memory and flip-flop RNZ is set to the 1-state if any bits of accumulator word B are binary 1's. If the bits of accumulator word B are also all binary 0's, the Program Processor takes a branch to the memory location specified by the instruction address field in the D-Register. If accumulator word B contains a binary 1 in any bit position, the Program Processor goes to the next instruction in the P-sequence.

Assuming that the accumulator working length is triple, and accumulator words A and B contain a binary 0 in each bit position, the Program Processor repeats block W02 a third time to check accumulator word C. If the accumulator working length is quadruple and if accumulator words A, B and C contain a binary 0 in each bit position, block W02 is repeated a fourth time to check accumulator word D.

Instruction 11: Branch if minus (BRM)

This instruction causes the contents of the P-Register to be replaced by the address field of the instruction word and a branch taken to the memory location specified by the instruction address field, if the sign of the Accumulator, contained in the zone bits of character 3 of accumulator word A, is minus. If the sign of the Accumulator is plus, the next instruction in the P-sequence is executed.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine illustrated in FIG. 121 is unaltered, if address development is required.

After completion of the initial routine, the W-Selector Register is preset to W02 and the Program Processor performs the micro-operations of the W02 block to execute the instruction. During the W02 block, accumulator word A is read from Memory and the zone bits of character 3 are checked. If the zone bits indicate that the sign of the Accumulator is minus, the address field of the instruction word is transferred to the P-Register to effect a branch. If the zone bits indicate the sign of the Accumulator to be plus, the address of the next instruction in the P-sequence is applied to the Adder.

The sequence of occurrence of the micro-operations required to execute Instruction 11 is illustrated in the timing diagram of FIG. 153. During time period TL0 of the W02 block, signal DABX issues to transfer the address of accumulator word A from the A-Register and the Accumulator Counter Register through the B-Gates into the B-Register of Memory. Flip-flop MRD is set to the 1-state at this time to initiate a read operation in Memory which transfers accumulator word A into the M-Register. During time period TL1, signals DCM0–DCM3 issue to clear the M-Register, preparatory to receiving accumulator word A from Memory during the read oper-

ation. Flip-flop DSX is reset to the 0-state at this time. Flip-flop ESX is reset to the 0-state. The first half-sequence of block W02 is terminated and the second half-sequence is initiated.

During the second half-sequence of block W02, flip-flop MWR is set to the 1-state during time period TL2 to initiate a write operation in Memory to restore accumulator word A. During time period TL5, flip-flop DSX is set to the 1-state in response to signal DMRM, which issues if the zone bits of character 3 of accumulator word A are 1, 0 indicating the sign of the Accumulator to be minus. Signal FDSX causes the instruction address field in the D-Register to be applied to the Adder. During the W00 block of the initial routine of the next instruction, signal QSPX issues during time period TL0 to transfer the instruction address field through the Adder to the P-Register to effect the branch to the location specified. If signal DMRM issues, indicating the sign of the Accumulator to be plus, flip-flop PSX is set to the 1-state to apply the address of the next instruction in the P-sequence from the P-Register to the Adder. Block W02 terminates and the Program Processor goes to block W00, preparatory to execution of the next instruction.

Instruction 16: Branch on count (BRC)

This instruction causes the Program Processor to branch to a subroutine and to execute the subroutine a predetermined number of times before continuing execution of instructions in the P-sequence. The repeated execution of the branch subroutine is directed by a control word containing a count which determines the number of times the subroutine is to be performed. The branch is taken to the memory location specified by the address field of the instruction word. The instruction address field may be developed by employing auxiliary words. The control word is obtained from a specified memory location. The address of the specified memory location is either the address of a Fixed Location Index Word or is developed by employing SAS auxiliary words.

The control word is organized, as illustrated in FIG. 2*l*, so that bit positions 15–23 contain a work count of $511-N$, where N is the number of times that a branch is taken. Bit positions 0–8 of the control word contain a restore count of $511-N$ which is employed to restore the work count field following the Nth execution of the branch subroutine. Bit positions 9–14 of the control word are not used. The work count field is incremented each time the instruction is executed. The restore count field is not changed.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine illustrated in FIG. 121 is unaltered, if development of the address field of the instruction word is required. After completion of the W00 block and development of the instruction address field in the W01 block, as required, the Program Processor again performs the appropriate micro-operations of the W01 block to develop the address of the specified memory location in which the control word is stored and to begin execution of the instruction. The Program Processor then performs the micro-operations of block W15S to complete execution of the instruction. The sequence of blocks W00, W01 to develop the instruction address field, if required, W01 to develop the address of the specified memory location, and W15S is repeated once for each time that the Program Processor branches to the subroutine.

The sequence of occurrence of the micro-operations required to execute Instruction 16 is illustrated in the timing diagram of FIG. 154. The W01 block of the initial routine illustrated in FIG. 121 is altered during development of the address of the control word, as indicated in FIG. 154. During the second half-sequence of the W01 block, after development of the address of the specified memory location has been completed, signal DVCA issues to prevent flip-flop MWR from being set to the 1-state, inhibiting the restoration of the control word in the M-Register to Memory. Flip-flop ESX is also retained in the 0-state at this time to inhibit the transfer of the instruction word in the E-Register to the adder inputs.

During time period TL3, upon completion of development of the address of the control word, flip-flop IRE is set to the 1-state to increment the work count field of the control word by one. Assuming that the work count field does not go to zero, indicating that at least one more branch is to be taken, block W01 is terminated with the instruction word in the E-Register and the control word in the M-Register. The flip-flops of the TL-Counter and the W-Selector Register are preset to define block W15S.

During time period TL2 of block W15S, flip-flop MWR is set to the 1-state to initiate a write operation in Memory which restores the control word in the M-Register to the specified memory location. Working clock signals QSMA, QSMB and QSMC issue to transfer the control word through the Adder to the M-Register, preparatory to storing it in Memory during the write operation. During time period TL3, flip-flops IRE and MSX are reset to the 0-state. During time period TL5, flip-flop ESX is set to the 1-state to apply the address field of the instruction word, which is the branch address, to the Adder. Flip-flop RCK is reset to the 0-state to terminate the W15S block. The W-Selector Register is preset to define the W00 block and the Program Processor next performs the micro-operations of the W00 block to read from Memory the instruction in the memory location specified by the instruction address field, which is the first instruction of the branch subroutine. The Program Processor executes the instructions of the subroutine, the last instruction of the subroutine being Instruction 16 (BRC).

Upon completion of the subroutine, the Branch On Count instruction word is again read from Memory and stored in the E-Register. The address field of the instruction word is again developed in block W01, if required. The micro-operations of the W01 block are performed to develop the address of the control word and to increment the work count field of the control word. If the work count field does not go to zero, the incremented control word is again restored to Memory and the address field of the instruction word is again transferred to the P-Register during the subsequent W15S and W00 blocks. The branch subroutine is again executed by the Program Processor.

The above-described sequence occurs until the incrementing of the work count field of the control word during block W01 causes the work count field to go to zero. At this time, signal MC23 issues, indicating a carry from the work count field, causing working clock signal QSEX to issue. Signal QSEX transfers the control word in the M-Register through the Adder into the E-Register, clearing the instruction word from the E-Register. Flip-flop CRE is set to the 1-state at this time to indicate that the work count field of the control word has gone to zero. During time period TL5 of the W01 block, working clock signal QEEX issues, transferring the restore count field in bit positions 0–8 of the control word into the work count field in bit positions 15–23. The work count field is thus restored and the control word may be used in subsequent executions of Instruction 16. The restore count field is not changed. Flip-flop ESX is set to the 1-state to apply the control word to the adder inputs. Flip-flop MSX is reset to the 0-state. The W01 block terminates and the Program Processor goes to block W15S.

During time period TL2 of the W15S block, flip-flop MWR is set to the 1-state to initiate a write operation in Memory which restores the control word to the specified memory location. Working clock signals QSMA, QSMB and QSMC issue to transfer the control word from the E-Register through the Adder into the M-Register, preparatory to storing the control word in Memory dur-

ing the write operation. During time period TL3, flip-flop MPL is set to the 1-state for use in a subsequent execution of Instruction 27. Flip-flops IRE, ESX and MSX are reset to the 0-state.

During time period TL5 of the W15S block, flip-flop PSX is set to the 1-state to apply the address of the next instruction in the P-sequence to the adder inputs. Flip-flop RCK is reset to the 0-state and the W15S block is terminated. The Program Processor then goes to the W00 block, preparatory to execution of the next instruction.

FIGURE 155 is a flow diagram illustrating the sequence of operations performed by the Program Processor in executing Instruction 16 (BRC).

Instruction 24: And to memory (ANM)

This instruction causes the contents of a first specified memory location to modify the contents of a second specified memory location as follows: for each bit position of the word in the first specified memory location that contains a zero, the corresponding bit position of the word in the second specified memory location is set to zero; for each bit position of the word in the first specified memory location that contains a one, the corresponding bit position of the word in the second specified memory location is left unchanged. The result is placed in the second specified memory location, leaving the contents of the first specified memory location unchanged. The address of the first specified memory location may be developed by employing auxiliary words. The address of the second specified memory location is specified to be the address of a Fixed Location Index Word or is developed by employing SAS auxiliary words.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine illustrated in FIG. 121 is unaltered, if development of the address of the first specified memory location is required.

After completion of the W00 block and development of the address of the first specified memory location in the W01 block, if required, the W-Selector Register is preset to define blocks W02, W01, W11S and W15S, in that order. The micro-operations corresponding to these blocks are performed to develop the address of the second specified memory location and to execute the instruction. During the W02 block of micro-operations, operand 1 is read from Memory and stored in the E-Register. In the description, operand 1 identifies the contents of the first specified memory location, while operand 2 designates the contents of the second specified memory location. During the W01 block, the address of the second specified memory location is developed and operand 2 is read from Memory into the M-Register. During the subsequent W11S block, operand 1 in the M-Register is modified in accordance with operand 2. Operand 1, as modified, is stored in the second specified memory location during the W15S block, to complete execution of the instruction.

The sequence of occurrence of the micro-operations required to execute Instruction 24 is illustrated in the timing diagram of FIG. 156. During time period TL0 of the W02 block of micro-operations, signal DDBX issues to transfer the address of the first memory location through the B-Gates into the B-Register of the Memory. Flip-flop MRD is set to the 1-state in response to working clock signal QTCK to initiate a read operation in Memory which transfers operand 1 from the first memory location into the M-Register. During time period TL1, signals DCM0–DCM3 issue to clear the M-Register, preparatory to receiving operand 1 from Memory during the read operation. Flip-flop MSX is set to the 1-state to transfer operand 1 from the M-Register to the Adder when operand 1 is received in the M-Register from Memory during the read operation. The first half-sequence of the W02 block is terminated and the second half-sequence is initiated by synchronizing signal DMDA from Memory.

During time period TL2 of the second half-sequence of the W02 block, flip-flop MWR is set to the 1-state to initiate a write operation in Memory, which restores operand 1 to the first memory location. During time period TL4, working clock signal QSEX issues to transfer operand 1 through the Adder into the E-Register. During time period TL5, flip-flop PSX is set to the 1-state to transfer the address of the next P-sequence word to the Adder. Flip-flop MSX is reset to the 0-state and the second half-sequence of block W02 is terminated. Flip-flops WR0 and WR1 of the W-Selector Register are set and reset respectively to define the W01 block.

During the W01 block, the address of the second specified memory location is developed, as illustrated in the timing diagram of FIG. 121, and operand 2 in the second specified memory location is read from Memory into the M-Register. The Program Processor then goes to block W11S to continue execution of the instruction.

During time period TL3 of block W11S, signal DNSL issues to set flip-flops N04 and N03 of the N-Register to the 1-state and to reset flip-flops N02, N01 and N00 to the 0-state, presetting the N-Register to a count of 24. During time period TL3, flip-flop SRI is set to the 1-state, preparatory to shifting the M- and E-Registers. During time period TL4, working clock signals QSHM and QSHE issue to shift operand 2 in the M-Register and operand 1 in the E-Register right one bit position. Simultaneous with the shift operation, the state of flip-flop M23 of the M-Register is modified in accordance with the states of flip-flops M00 and E00 as follows: if both flip-flops M00 and E00 are set to the 1-state, flip-flop M23 is set to the 1-state; if either M00 or E00 are reset to the 0-state, flip-flop M23 is reset to the 0-state. The shifting of the operands in the M- and E-Registers and the modification of the state of flip-flop M23 simultaneous with the shift operation is made possible by the time delay between the change of state of the flip-flops of the M- and E-Registers and the resulting change in the flip-flop output signals. Working clock signal QNCD issues at this time to reduce the count in the N-Register by one, signal QNCD issuing once for each shift of the M- and E-Registers.

During time period TL5, working clock signal QTLP is inhibited by signal FSRI to prevent a change in the state of the TL-Counter. In response to working clock signal QTCK, shift signals QSHM and QSHE issue to shift operands 1 and 2 in the M- and E-Registers respectively right one more bit position. The logical operation involving the contents of flip-flops M00 and E00 is again performed, with the result of the logical operation being stored in flip-flop M23. Working clock signal QTCK continues to issue with the TL-Counter remaining at TL5, until operands 1 and 2 have been shifted through 24 bit positions and the logical operation has been performed 24 times. At this time, the count in the N-Register is one and flip-flop SRI is reset to the 0-state. Working clock signal QTLP again issues to advance the TL-Counter and the W11S block is terminated. The flip-flops of the TL-Counter and the W-Selector Register are preset to define block W15S.

During time period TL2 of block W15S, signal DDBX issues to transfer the address of the second specified memory location from the D-Register through the B-Gates into the B-Register of Memory. Flip-flop MWR is set to the 1-state at this time to initiate a write operation in the Memory which transfers operand 1, as modified, from the M-Register into the second specified memory location. During time period TL5, flip-flop PSX is set to the 1-state to transfer the address of the next instruction from the P-Register to the Adder. Flip-flop RCK is reset to the 0-state to terminate the W15S block. The flip-flops of the W-Selector Register are reset to the 0-state to define block W00, preparatory to execution of the next instruction.

Instruction 23: OR Inclusive to memory (RIM)

This instruction causes the contents of a first specified memory location to modify the contents of a second specified memory location as follows: for each bit position of the word in the first specified memory location that contains a binary 1, the corresponding bit position of the word in the second specified memory location is set to binary 1; for each bit position of the word in the first specified memory location that contains a binary 0, the corresponding bit position of the word in the second specified memory location is left unchanged. The result is placed in the second specified memory location, leaving the contents of the first specified memory location unchanged. The address of the first specified memory location may be developed by employing auxiliary words. The address of the second specified memory location is specified to be the address of a Fixed Location Index Word or is developed by employing SAS auxiliary words.

The execution of Instruction 23 (RIM) is identical to the execution of Instruction 24 (ANM), as illustrated in FIG. 156, with the following exception. During time periods TL4 and TL5 of the W11S block, flip-flop M23 is set to the 1-state if either flip-flop M00 or flip-flop E00 are set to the 1-state. Flip-flop M23 is reset to the 0-state only if both flip-flops M00 and E00 are reset to the 0-state.

Instruction 25: OR Exclusive to memory (RXM)

This instruction causes the contents of a first specified memory location to modify the contents of a second specified memory location as follows: for each bit position of the word in the first specified memory location that contains a binary 1, the corresponding bit position of the word in the second specified memory location is inverted; for each bit position of the word in the first specified memory location that contains a binary 0, the corresponding bit position of the word in the second specified memory location is left unchanged. The result is stored in the second specified memory location, leaving the contents of the first specified memory location unchanged. The address of the first specified memory location may be developed by employing auxiliary words. The address of the second specified memory location is specified to be the address of a Fixed Location Index Word or is developed by employing SAS auxiliary words.

The execution of Instruction 25 (RXM) is identical to the execution of Instruction 24 (ANM), as illustrated in FIG. 156, with the following exception. During time periods TL4 and TL5 of the W11S block, flip-flop M23 of the M-Register is set to the 1-state if flip-flops M00 and E00 have opposite states, i.e. if flip-flop M00 is set to the 1-state and flip-flop E00 is reset to the 0-state or if flip-flop M00 is reset to the 0-state and flip-flop E00 is set to the 1-state. Flip-flop M23 of the M-Register is reset to the 0-state if flip-flops M00 and E00 have the same state, i.e. both flip-flops E00 and M00 are set to the 1-state or both flip-flops E00 and M00 are reset to the 0-state.

Instruction 36: Load accumulator location and length (LAL)

This instruction establishes the location and the working length of the Accumulator as follows: bits 2–14 of the instruction address field are transferred to the A-Register to define the location of the most-significant word of the Accumulator; bits 0 and 1 of the instruction address field are transferred to flip-flops PL1 and PL2 of the Accumulator Length Indicator Register to establish the accumulator working length. Bits 0–14 of the instruction address field can also be interpreted as specifying the most-significant word of the working Accumulator. In changing the location and working length of the Accumulator, the contents of both the old and new locations are undisturbed. The instruction address field may be developed by employing auxiliary words.

The Program Processor executes the instruction during the performance of block W00 of the initial routine, if address development is not required, or during block W01 of the initial routine, if address development is required. The micro-operations required to execute Instruction 36 are illustrated in the timing diagram of FIG. 157.

Referring to FIG. 157, the W00 block of the initial routine, illustrated in FIG. 119, is altered to execute the instruction as follows. During time period TL4 of the W00 block, signal DCLA issues to clear the A-Register, resetting the flip-flops of the A-Register to the 0-state. If address development is not required, flip-flop PSX is set to the 1-state during time period TL5 of the W00 block to apply the address of the next instruction in the P-Register to the Adder. Assuming that development of the address field is not required, the Program Processor then goes to block W00 to execute the next instruction. During the W00 block of the initial routine of the next instruction, working clock signal QEAX issues during time period TL0 to transfer bits 2–14 of the instruction word in the E-Register to the A-Register to define the new accumulator location. Signal QEAX also transfers bits 0 and 1 of the instruction word in the E-Register to flip-flops PL1 and PL2 respectively of the Accumulator Length Indicator Register to establish the working length of the Accumulator.

If development of the address field of the instruction word is required, the Program Processor goes to block W01. The W01 block illustrated in FIG. 121, is altered as indicated in FIG. 157 to execute the instruction. During time period TL4 of the W01 block, working clock signal QSEX issues, after instruction address field development is complete, to transfer the developed address field to the E-Register. During time period TL5, flip-flop PSX is set to the 1-state to apply the address of the next instruction in the P-Register to the Adder. Block W01 is terminated and the Program Processor goes to block W00, preparatory to execution of the next instruction. During the W00 block of the initial routine of the next instruction, working clock signal QEAX issues, as described above, to transfer the appropriate bits of the instruction address field to the A-Register and to flip-flops PL1 and PL2 to establish the working accumulator location and length.

Instruction 37: Store accumulator location and length (SAL)

This instruction causes the current accumulator location and working length to be stored in bit positions 0–14 of a word in a specified memory location. The contents of bit positions 15–23 of the word in the specified memory location are not changed. The address of the most-significant word of the Accumulator is stored in bit positions 2–14 of the word while the bits representing the accumulator working length are stored in bit positions 0 and 1 of the word. The accumulator location, contents and length are not changed. The address of the specified memory location may be developed by employing auxiliary words.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine illustrated in FIG. 121 is unaltered, if development of the address field of the instruction word which identifies the specified memory location is required.

Upon completion of the initial routine, the W-Selector Register is preset to W03 and the Program Processor performs the micro-operations of the W03 block to execute the instruction. During the W03 block of micro-operations, the accumulator location and working length are transferred from the A-Register and the Accumulator Length Indicator Register to the M-Register and stored in the specified memory location.

The sequence of occurrence of the micro-operation required to execute Instruction 37 is illustrated in the timing diagram of FIG. 158. During time period TL0 of the W03 block, signal DDBX issues to transfer the address of the specified memory location from the D-Register

through the B-Gates into the B-Register of Memory. Flip-flop MRD is set to the 1-state at this time to initiate a read operation in Memory which transfers the word in the specified memory location into the M-Register from Memory during the read operation. During time period TL1, signals DCM0–DCM3 issue to clear the M-Register, preparatory to receiving the word from the specified memory location during the read operation. Signal DAEX issues at this time to transfer the binary representation of the accumulator location from the A-Register into bit positions 2–14 of the E-Register. Signal DAEX also transfers the binary representation of the accumulator working length from flip-flops PL1 and PL2 of the Accumulator Length Indicator Register into bit positions 0 and 1 respectively of the E-Register. Flip-flop ESX is also set at this time to apply the accumulator location and length in the E-Register to the adder inputs. Flip-flop RCK is set to the 0-state and the first half-sequence of the W03 block is terminated. Synchronizing signal DMDA provided by Memory initiates the second half-sequence of the W03 block.

During time period TL2 of the W03 block, flip-flop MWR issues to initiate a write operation in Memory which stores the binary representations of the accumulator location and length in the specified memory location. Working clock signals QSMA and QSMB issue at this time to transfer the binary representations of the accumulator location and length through the Adder into bit positions 2–14 of the M-Register, preparatory to storing the representations in Memory during the write operation. This transfer clears the former contents from bit positions 0–14 of the M-Register. During time period, TL4, flip-flop ESX is reset to the 0-state. During time period TL5 flip-flop PSX is set to the 1-state to apply the address of the next instruction in the P-Register to the adder inputs. The transfer of the word in the M-Register with the accumulator location and length stored in the address field is completed and the second half-sequence of the W03 block is terminated. Flip-flops WR1 and WR0 of the W-Selector Register are reset to the 0-state, define block W00, preparatory to execution of the next instruction.

Instruction 27: Variable length multiply (VLM)

This instruction causes the contents of a memory field in two specified adjacent memory locations to be multiplied by the least-significant character in the Accumulator. The nine-digit product generated is stored in the nine most-significant character positions of the Accumulator. The contents of the memory field comprise the multiplicand, while the least-significant character of the Accumulator comprises the multiplier character. The multiplier character is lost during each execution of Instruction 27. The location of the least significant word of the multiplicand is defined by the address field of the instruction word. The sign of the product appears in the zone bits of the least-significant character of the product; the zone bits of all other product characters are set to zero. The working length of the Accumulator does not affect and is not changed by Instruction 27. The multiplication apparatus disclosed herein includes the invention of Messrs. Edwin W. Herron, Robert D. Hunter, and David E. Keefer.

The full multiplier may comprise from one to eight characters and is stored in the accumulator word A and B locations in Memory. Instruction 27 is executed once for each character of the multiplier. Instruction 27 performs data shifts which correctly position the multiplier and the product, preparatory to re-execution of Instruction 27 with another multiplier character. A single Instruction 27 can be used to form an $8+N$-character product, where N is the number of characters in the multiplier, by controlling the re-execution of Instruction 27 with Instruction 16, Branch on Count, that has been set to allow N executions of Instruction 27. Upon completion of execution of Instruction 27, the product, if less than 16 characters must be shifted right for proper positioning in the Accumulator and for sign preservation, the number of character positions to be shifted eight minus the number of times Instruction 27 is executed. Execution of Instruction 27 can result in a minus zero product. The Overflow flip-flop is never set by Instruction 27.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine illustrated in FIG. 121 is unaltered, if development of the address field of the instruction word is required.

Upon completion of the initial routine, the W-Selector Register and the TL-Counter are set to define blocks W04R, W11S, W04S, W04R, W11S, W03S, W04S, W04R, W07, W11S, W03S, W04S, W04R, W06S, W07, W11S, W03S, and W04S, in that order, and the Program Processor performs the micro-operations of these blocks to execute the instruction.

The following table summarizes the macro-operation which is performed during each block of the multiplication sequence.

Block:	Macro-operations
W04R --	Accumulator word A read from Memory into the M-Register.
W11S --	Remember sign of Accumulator. Transfer multiplier character (character 3) to N-Register. Transfer accumulator word A to E-Register and shift right one character position.
W04S --	Store characters 0, 1 and 2 in character positions 1, 2 and 3 respectively of the accumulator word A location in Memory. Store partial parity for accumulator word A location.
W04R --	Read accumulator word B from Memory into the M-Register.
W11S --	Transfer accumulator word B to E-Register and shift right one character position, shifting character 3 of accumulator word B into the character 0 position of the M-Register.
W03S --	Store character 3 of accumulator word B in the character 0 position of the accumulator word A location in Memory and store parity bit in accumulator word A location.
W04S --	Store characters 0, 1 and 2 of accumulator word B in the character 1, 2 and 3 positions of the accumulator word B location in Memory. Store partial parity for the accumulator word B location.
W04R --	Read accumulator word C from Memory into the M-Register.
W07 ---	Clear accumulator word C from M-Register if first execution of instruction, i.e. if multiplicand is being multiplied by first multiplier character. If not first execution of instruction, accumulator word C is partial product and is transferred to E-Register. Least-significant multiplicand word is read from Memory into the M-Register. The multiplier character is stored in the Q-Register and the signs of the multiplier and multiplicand are compared. The multiplicand is successively added to itself for a number of times equal to the value of the multiplier to effect the multiplication. If not first execution of instruction, result is added to partial product in E-Register. Carries are stored in the CC-Register. The product is stored in the E-Register.

Block: Macro-operations

W11S -- The sign is inserted in the product in the E-Register, if the instruction is being executed for the first time. The product in the E-Register is shifted right one character position, character 3 of the result being transferred to the M-Register. The multiplier character is restored to the N-Register.

W03S -- Character 3 of the product is stored in the character 0 position of the accumulator word B location in Memory and the parity bit is generated and transferred to the accumulator word B location.

W04S -- Character 0, 1 and 2 of the product are stored in the character 1, 2 and 3 positions of the accumulator word C location. Partial parity for the accumulator word C location is stored.

W04R -- Accumulator word D is read from Memory into the M-Register.

W06S -- The carry count from the previous product is applied to the Adder.

W07 --- Accumulator word D is cleared from the M-Register, if the instruction is being executed for the first time. If not first execution of the instruction, accumulator word D is partial product and is transferred to the E-Register. The most-significant multiplicand is read from Memory into the M-Register. The most-significant multiplicand word is added to the carry count from the previous product, to the partial product in the E-Register if not first execution of instruction, and to itself a number of times equal to the value of the multiplier to effect the multiplication. Carries are stored in the CC-Register. The product is stored in the E-Register.

W11S -- The product in the E-Register is shifted right one character position, character 3 of the product being transferred to the M-Register.

W03S -- Character 3 of the product is stored in the character 0 position of the accumulator word C location in the Memory. The parity bit for the accumulator word C location is generated and stored.

W04S -- The carry count from the CC-Register and characters 0, 1 and 2 of the product are stored in the character 0, 1, 2 and 3 positions respectively of the accumulator word D location. Instruction execution is terminated and the Program Processor goes to W00 block.

As indicated in the above tables, if the multiplier is being multiplied by a multiplier character other than the least-significant character of the full multiplier, accumulator words C and D are not cleared from the M-Register during the W07 blocks but are transferred to the E-Register since they comprise partial products from the previous execution of the instruction which are to be added to the product to be generated. As described above, the partial product is shifted during each execution of the instruction to correctly position the partial product for addition to the next product. The signs of the multiplier and the multiplicand are not sensed and the sign of the product is not determined, if the execution of the instruction is not the first execution. Flip-flop MPL, which indicates whether or not the instruction is being executed for the first time, is reset to the 0-state after the first execution and the state is not thereafter changed until changed by the program or by completion of the multiplication.

The sequence of occurrence of the micro-operations required to execute Instruction 27 is illustrated in the timing diagram of FIG. 159. Referring to block W04R of FIG. 159, signal DABX issues during time period TL0 to transfer the address of accumulator word A through the B-Gates into the B-Register of Memory. Flip-flop MRD is set to the 1-state to initiate a read operation in Memory which transfers accumulator word A into the M-Register. Signals DCM0–DCM3 issue to clear the M-Register, preparatory to receiving accumulator word A from Memory during the read operation. Flip-flop MSX is set to the 1-state to apply accumulator word A to the Adder, as soon as it is received in the M-Register during the read operation. The first half-sequence of block W04R is terminated and the second half-sequence is initiated upon issuance of memory synchronizing signal DMDA. Signal DCAD issues to form the address of accumulator word D in the A-Register and the Accumulator Counter Register. The Program Processor goes to block W11S to continue execution of the instruction.

During time period TL2 of block W11S, flip-flop SIN is set to the 1-state if the character 3 zone bits of accumulator word A indicate that the sign of the Accumulator is minus. Working clock signal QSEX issues to transfer accumulator word A to the E-Register. Working clock signal QSCX also issues to transfer character 3 of accumulator word A, which is the first multiplier character, to the CC-Register. Signal DCLQ clears the Q-Register and signals DCM0–DCM3 issue to clear the M-Register. Flip-flop SR1 is set to the 1-state to initiate a shift operation and signal QCNX issues to transfer the multiplier character in the CC-Register to the N-Register. During time period TL4, working clock signal QSHE issues to shift accumulator word A during time periods TL4 and TL5. Working clock signal QSHE issues six times to shift accumulator word A in the E-Register right one character position. Block W11S terminates with signal DACU issuing to form the address of accumulator word A in the A-Register and the Accumulator Counter Register.

During block W04S, which follows block W11S, signal DABX issues during time period TL2 to transfer the address of accumulator word A from the A-Register and the Accumulator Counter Register to the B-Register of Memory. Signals QSMA, QSMB and QSMC issue to transfer the contents of the E-Register to the M-Register. Flip-flop MWR is set to the 1-state to initiate a write operation which stores characters 0, 1 and 2 of accumulator word A in character positions 1, 2 and 3 respectively of the accumulator word A location in Memory, a zero being temporarily stored in the character 0 position of the accumulator word A location. Flip-flop SP2 is set to the 1-state during time period TL4 to store partial parity if the characters stored in the character 1, 2 and 3 positions of the accumulator word A location contained an odd number of binary 1's. During time period TL0, signal DACU issues to form the address of accumulator word B in the A-Register and the Accumulator Counter Register. Block W04S terminates and the Program Processor goes to block W04R.

During time period TL0 of block W04R, signal DABX issues to transfer the address of accumulator word B into the B-Register of the Memory. Flip-flop MRD issues to initiate a read operation which transfers accumulator word B from Memory into the M-Register. Signals DCM0–DCM3 issue to clear the M-Register, preparatory to receiving accumulator word B from Memory during the read operation. Flip-flop MSX is set to the 1-state to apply accumulator word B to the Adder as soon as it is received in the M-Register during the read operation. Block W04R terminates with signal DCAD issuing to form the address of accumulator word A in the A-Register and the Accumulator Counter Register.

During time period TL2 of block W11S following block W04S, working clock signal QSEX issues to transfer accumulator word B to the E-Register. Signals DCLQ and DCM0–DCM3 issue to clear the Q- and M-Registers respectively. Flip-flop SRI is set to the 1-state to initiate a shift operation. During time periods TL4 and TL5, working clock signals QSHM and QSHE issue six times to shift accumulator word B in the E-Register through one character position, character 3 of accumulator word B being shifted into flip-flops M18–M23 of the M-Register. Block W11S terminates and the Program Processor goes to block W03S to continue execution of the instruction.

During time period TL2 of block W03S, signal DABX issues to transfer the address of accumulator word A to the B-Register of Memory. Flip-flop MWR is set to the 1-state to initiate a write operation which stores character 3 of accumulator word B into the character 0 position of the accumulator word A location. The state of partial parity flip-flop SP2 is employed to generate the parity bit for the accumulator word A location. Signal DACU issues during time period TL0 to form the address of accumulator word B in the A-Register and the Accumulator Counter Register and the Program Processor goes to block W04S.

During time period TL2 of block W04S, signal DABX transfers the address of accumulator word B into the B-Register of Memory. Working clock signals QSMA, QSMB and QSMC issue to transfer the contents of the E-Register into the M-Register and flip-flop MWR is set to the 1-state to initiate a write operation. The write operation initiated by signal FMWR stores characters 0, 1 and 2 into the character 1, 2 and 3 positions of the accumulator word B location in Memory, a zero being temporarily stored in the character 0 position. Flip-flop SP2 is set during time period TL4 to store partial parity for the accumulator word B location, if the information stored in character positions 1, 2 and 3 of the accumulator word B location contain an odd number of binary 1's. Signal DACU issues to form the address of accumulator word C in the A-Register and the Accumulator Counter Register and the Program Processor next performs block W04R.

Signal DABX issues during time period TL0 of block W04R to transfer the address of accumulator word C to the B-Register of the Memory. Signals DCM0–DCM3 issue to clear the M-Register and flip-flop MRD is set to the 1-state to initiate a read operation which transfers accumulator word C into the M-Register. If the instruction is being executed for the first time, i.e. if the multiplicand is being multiplied by the least-significant character of the full multiplier, flip-flop MPL is set to the 1-state and signal FMSX is inhibited. Accumulator word C is therefore not applied to the Adder. If flip-flop MPL were reset to the 0-state, accumulator word C would be a partial product from a previous execution of the instruction and would be applied to the Adder for transfer to the E-Register. Signal DCAD issues during time period TL2 to form the address of accumulator word B in the A-Register and the Accumulator Counter Register.

During time period TL0 of block W07 following block W04R, signal DDBX issues to transfer the address of the least-significant multiplicand word from the D-Register to the B-Register of Memory. Flip-flop MRD is set to the 1-state to initiate a read operation in Memory which transfers the least-significant multiplicand word into the M-Register. Signal QSEX issues at this time to either clear the E-Register if the multiplicand is being multiplied by the least-significant multiplier character or to transfer the partial product from the accumulator word C location into the E-Register, if the multiplicand is being multiplied by a higher-order multiplier character. During time period TL1, the multiplier character in the N-Register is transferred to the Q-Register and signals DCM0–DCM3 issue to clear the M-Register, preparatory to receiving the least-significant multiplicand word from Memory during the read operation. Signal DCLC also issues at this time to clear the CC-Register. Flip-flop MSX is set to the 1-state to apply the least-significant multiplicand word in the M-Register to the Adder, when it is received in the M-Register during the read operation. Flip-flop ESX is also set to the 1-state at this time. If the multiplicand is being multiplied by a multiplier character other than the least-significant multiplier character, signal FESX causes the partial product in the E-Register to be applied to the adder inputs, forming the sum of the partial product and the least-significant multiplicand word. The first half-sequence of block W07 terminates and signal DMDA issues from Memory to initiate the second half-sequence.

Flip-flop MOD is set to the 1-state during time period TL2 of the second half-sequence of block W07 to initiate a multiplication operation, unless the multiplier character in the N-Register is zero. Flip-flop MWR is set to the 1-state to initiate a write operation which restores the least-significant multiplicand word to Memory. If the multiplicand is being multiplied by the least-significant multiplier character, the state of flip-flop LSN is determined at this time, LSN being reset to the 0-state if the signs of the multiplier and the multiplicand are opposite. Signal FMOD causes signal DBCU to issue at this time. In response to working clock signal QTCK and signal DBCU during time period TL3, flip-flop TB0 is set to the 1-state to form a count of one in the TB-Counter which comprises flip-flops TB1 and TB0. In response to the count of one in the TB-Counter, Carry Remember flip-flop CRE is reset to the 0-state during time period TL4, preparatory to storing carries during the sequence of additions comprising the multiplication operation. Flip-flop TB1 is set to the 1-state while flip-flop TB0 is reset to the 0-state during time period TL4 to advance the count in the TB-Counter to two. During time period TL5, working clock signal QTLP is inhibited by signal FMOD, preventing the advance of the TL-Counter. During time period TL5, in response to a count of two in the TB-Counter, working clock signal QSEX issues to transfer the adder output to the E-Register. If the multiplication operation involves the least-significant multiplier character, the adder output at this time will comprise the least-significant multiplicand word. If the multiplication operation involves a higher-order multiplier character, the adder output at this time will comprise the sum of the least-significant multiplicand word and the partial product derived from the accumulator word C location. Working clock signal QNCD also issues at this time to reduce the multiplier count in the N-Register by one. Flip-flop CRE is set to the 1-state if a carry is propagated from character 0 of the adder output. Flip-flops TB1 and TB0 of the TB-Counter are both reset to the 0-state, preparatory to another addition operation. Signal DCCU also issues at this time to advance the count to the CC-Register by one, if a carry was detected during the addition operation.

The TL-Counter remains at TL5 and the TB-Counter is again advanced from a count of zero to a count of two by successive working clock signals QTCK. When the count in the TB-Counter reaches two, signals QSEX and QNCD again issue to effect another addition of the least-significant multiplicand word to the contents of the E-Register and to reduce the count in the N-Register. This sequence is repeated until the number of signals QSEX which have issued equals the count or the value of the multiplier character in the N-Register. At this time, flip-flop MOD is reset to the 0-state and the multiplication by successive additions is terminated. The Program Processor then goes to block W11S.

Signal DCCU issues during time period TL2 of block W11S if flip-flop CRE is set to the 1-state to complete the accumulation of carries in the CC-Register resulting from the multiplication operation of block W07. Signal QQNX also issues at this time to transfer the multiplier character

stored in the Q-Register to the N-Register, preparatory to the multiplication of the most-significant multiplicand word by the multiplier character. During time period TL3, flip-flop E05 of the E-Register is set to the 1-state if flip-flop LSN is reset to the 0-state and if the multiplication operation involves the least-significant multiplier character. The reset state of flip-flop LSN indicates that the sign of either the multiplier or the multiplicand is minus. The setting of flip-flop E05 inserts a minus sign into the product in the E-Register. If the multiplication operation involves a higher-order multiplier character, the sign of the product was determined during a previous execution of the instruction.

Signals DCLQ and DCM0–DCM3 issue at this time to clear the Q- and M-Registers respectively. Flip-flop SRI is set to the 1-state at this time to initiate a shift operation in the M- and E-Registers. During time periods TL4 and TL5, working clock signals QSHM and QSHE issue six times to shift the contents of the E-Register right one character position, the least-significant character of the product being shifted into flip-flops M18–M23 of the M-Register. Flip-flop ESX is set to the 1-state during time period TL4 to apply the shifted contents of the E-Register to the Adder. Block W11S is terminated and the Program Processor goes to block W03S to continue execution of the instruction.

Signal DABX issues during time period TL2 of block W03S to transfer the address of accumulator word B to the B-Register of Memory. Flip-flop MWR is set to the 1-state to initiate a write operation which stores the least-significant character of the product in the character 0 position of the accumulator word B location in the Memory. The state of partial parity flip-flop SP2 is employed to generate the parity bit for the accumulator word B location. Signal DACU issues during time period TL0 to form the address of the accumulator word C location in the A-Register and the Accumulator Counter Register. Block W03S is terminated and the Program Processor next goes to block W04S.

Signal DABX issues during time period TL2 of block W04S to transfer the address of the accumulator word C location to the B-Register of Memory. Working clock signals QSMA, QSMB and QSMC issue to transfer the three most-significant characters of the product into the M-register. Flip-flop MWR is set to the 1-state at this time to initiate a write operation in Memory which stores the three most-significant characters of the product in the character 1, 2 and 3 positions of the accumulator word C location in Memory. A zero is temporarily stored in the character 0 position of the accumulator word C location. Partial parity flip-flop SP2 is set to the 1-state during time period TL4 to store partial parity, if the characters stored in the accumulator word C location contain an odd number of binary 1's. Signal DACU issues, forming the address of the accumulator word D location in the A-Register and the Accumulator Counter Register. The Program Processor next performs block W04R.

During time period TL0 of block W04R, signal DABX transfers the accumulator word D address to Memory and flip-flop MRD is set to the 1-state to initiate a read operation which transfers accumulator word D into the M-Register. Signals DCM0–DCM3 clear the M-Register, preparatory to receiving accumulator word D during the read operation. Signal DDCD issues to reduce the count in the D-Register by one, forming the address of the most-significant multiplicand word. If the multiplication operation involves a multiplier character other than the least-significant multiplier character, flip-flop MSX is set to the 1-state to cause accumulator word C, which is a partial product from a previous multiplication, to be applied to the Adder for transfer to the E-Register. Signal DCAD issues to form the address of accumulator word C in the A-Register and the Accumulator Counter Register. Block W04R terminates and the Program Processor goes to block W06S.

Flip-flop CS2 is set to the 1-state during time period TL3 of block W06S to apply the carry count in the CC-Register to the inputs of full adder circuits S00–S03. Flip-flop DCE is set to the 1-state to enable the Adder to operate in the decimal mode. Block W06S is then terminated and the Program Processor goes to block W07 to effect multiplication of the most-significant multiplicand word and the multiplier character.

During time period T10 of block W07, signal DDBX transfers the address of the most-significant multiplicand word from the D-Register to the B-Register of Memory. Flip-flop MRD is set to the 1-state to initiate a read operation which transfers the most-significant multiplicand word into the M-Register. If the instruction is being executed for the first time, signal QSEX transfers the carries from the multiplication operation involving the least-significant multiplicand word to the E-Register. If the multiplication operation involves a multiplier character other than the least-significant multiplier character, signal QSEX transfers the sum of the carries from the multiplication operation involving the least-significant multiplicand word and the partial product stored in the accumulator word D location to the E-Register. Flip-flop CRE is set to the 1-state at this time if a carry is propagated from character 0 of the adder outputs.

During time period TL1 of block W07, the multiplier count in the N-Register is again stored in the Q-Register. Flip-flop MPL is reset to the 0-state at this time, if the multiplication operation involves the least-significant multiplier character. Otherwise, flip-flop MPL would have been reset to the 0-state during a previous execution of the instruction. Signals DCLC and DCM0–DCM3 issue to clear the CC- and the M-Registers respectively. Flip-flops MSX and ESX are set to the 1-state at this time to apply the most-significant multiplicand word in the M-Register and the contents of the E-Register to the Adder. Flip-flops TB1 and TB0 are reset to the 0-state, preparatory to the multiplication operation. The first half-sequence of the W07 block is terminated and the second half-sequence is initiated by synchronizing signal DMDA from Memory.

During time period TL2 of the W07 block, flip-flop MOD is set to the 1-state to initiate the multiplication operation, if the multiplier character in the N-Register is not equal to zero. Flip-flop MWR is set to the 1-state to initiate a write operation which restores the most-significant multiplicand word to Memory. The count in the TB-Counter is again repeatedly advanced from zero through two and the signals QSEX and QNCD repeatedly issue to perform successive additions of the most-significant multiplicand word and the contents of the E-Register and to reduce the count in the N-Register. The carries are stored in the CC-Register. Upon completion of the multiplication operation, with the product stored in the E-Register and the carries stored in the CC-Register, flip-flop MOD is reset to the 0-state and the Program Processor goes to block W11S.

Signal DCCU issues during time period TL2 of block W11S, if flip-flop CRE is set to the 1-state, to complete the accumulation of carries in the CC-Register. Signal DCLQ and DCM0–DCM3 issue to clear the Q- and M-Registers respectively. Flip-flop SRI is set to the 1-state to initiate the shift operation. During time period TL4, flip-flop CS3 is set to the 1-state to apply the carry count in the CC-Register to the inputs of full adders S18–S21. Working clock signals QSHM and QSHE issue six times during time periods TL4 and TL5 to shift the product in the E-Register through one character position, the least-significant character of the product being transferred to flip-flops M18–M23 of the M-Register. Upon completion of the shift operation, the Program Processor goes to block W03S.

Signal DABX issues during time period TL2 of block W03S to transfer the address of the accumulator word

C location into the B-Register of Memory. Flip-flop MWR is set to the 1-state to initiate a write operation which stores the least-significant character of the product in the character 0 position of the accumulator word C location in the Memory. The parity bit for the accumulator word C location is generated and stored at this time. Signal DACU issues to form the address of the accumulator word D location in the A-Register and the Accumulator Counter Register and the Program Processor next goes to block W04S to complete execution of the instruction.

Signal DABX transfers the address of the accumulator word D location to the B-Register of Memory during the time period TL2 of the W04S block. Working clock signals QSMA, QSMB and QSMC issue to transfer the carry count and the three most-significant characters of the product to the M-Register. Flip-flop MWR is set to the 1-state to initiate a write operation which stores the carry count and the three most-significant characters of the product in the character 0, 1, 2 and 3 positions respectively of the accumulator word D location in Memory. Flip-flop PSX is set to the 1-state during time period TL5 to apply the address of the next instruction in the P-sequence to the Adder. The Program Processor then goes to the W00 block, preparatory to the execution of the next instruction.

The following table illustrates the sequence of blocks performed to execute Instruction 27 in multiplying 56785678 by 54 and the contents of the indicated registers after the performance of each block:

INSTRUCTION 27.—MULTIPLY 56785678 BY 54

Multiplicand

5678 5678

Block	Accumulator	N-Register	CC-Register (Carries)	M-Register	E-Register
	XXXX XXXX 9234 1254	XX	XX	0000	0000
W04R	XXXX XXXX 9234 0000	XX	XX	1254	0000
W11S	XXXX XXXX 9234 0000	04	XX	4000	0125
W04S	XXXX XXXX 9234 0125	04	XX	0125	0125
W04R	XXXX XXXX 0000 0125	04	XX	9234	0125
W11S	XXXX XXXX 0000 0125	04	XX	4000	0923
W03S	XXXX XXXX 0000 4125	04	XX	4000	0923
W04S	XXXX XXXX 0923 4125	04	XX	0923	0923
W04R	XXXX 0000 0923 4125	04	XX	XXXX	0923
W07	XXXX 0000 0923 4125	00	02	5678	2712
W11S	XXXX 0000 0923 4125	04	02	2000	0271
W03S	XXXX 0000 2923 4125	04	02	2000	0271
W04S	XXXX 0271 2923 4125	04	02	0271	0271
W04R	0000 0271 2923 4125	04	02	XXXX	0271
W06S, W07	0000 0271 2923 4125	00	02	5678	2714
W11S	0000 0271 2923 4125	00	02	4000	0271
W03S	0000 4271 2923 4125	00	02	4000	0271
W04S	2271 4271 2923 4125	00	02	2271	0271
W04R	2271 4271 2923 0000	00	02	4125	0271
W11S	2271 4271 2923 0000	05	02	5000	0412
W04S	2271 4271 2923 0412	05	02	0412	0412
W04R	2271 4271 0000 0412	05	02	2923	0412
W11S	2271 4271 0000 0412	05	02	3000	0292
W03S	2271 4271 0000 3412	05	02	3000	0292
W04S	2271 4271 0292 3412	05	02	0292	0292
W04R	2271 0000 0292 3412	05	02	4271	0292
W07	2271 0000 0292 3412	00	03	5678	2661

INSTRUCTION 27.—MULTIPLY 50785078 BY 54—Continued

Block	Accumulator	N-Register	CC-Register (Carries)	M-Register	E-Register
W11S	2271 0000 0292 3412	05	03	1000	0266
W03S	2271 0000 1292 3412	05	03	1000	0266
W04S	2271 0266 1292 3412	05	03	0266	0266
W04R	0000 0266 1292 3412	05	03	2271	0266
W06S, W07	0000 0266 1292 3412	00	03	5678	0664
W11S	0000 0266 1292 3412	00	03	4000	0066
W03S	0000 4266 1292 3412	00	03	4000	0066
W04S	3066 4266 1292 3412	00	03	3066	0066

Instruction 26: Variable length divide (VLD)

This instruction causes the nine most-significant characters of the Accumulator to be divided by a two-word memory field stored in specified adjacent memory locations. The arithmetic division apparatus described herein includes the invention of Edwin W. Herron, Robert D. Hunter, and David E. Keefer. For each execution of the instruction, a single quotient character is generated and stored in the least-significant character position of the Accumulator, with any remainder, the balance of the dividend, and any previously-generated quotient characters occupying the remaining character positions in the Accumulator. The contents of the Accumulator comprise the dividend while the memory field comprises the divisor. The address field of the instruction word defines the location of the least-significant word of the divisor. The sign of the quotient is determined by the sign of the divisor and the sign of the least-significant character of the nine-character dividend. The working length of the Accumulator does not affect and is not changed by Instruction 26.

The full dividend may comprise from 9 to 16 characters, depending upon the number of quotient characters desired and hence the number of executions of the instruction. For N executions of the instruction, the effective dividend length is $8+N$ characters. For each execution of the instruction, the dividend is considered to be the nine digits held in the accumulator word D and C locations and the most-significant character of the accumulator word B location. The contents of the Accumulator are shifted left one character position during each execution of the instruction to properly position the dividend characters in the Accumulator. After each execution of the instruction, the accumulator word D and C locations contain the remainder generated by a single execution of the instruction. This remainder and the most-significant character of the accumulator word B location form the effective dividend for the next execution of the instruction.

After each execution of the instruction, the quotient is stored in the least-significant character position of the Accumulator. During each execution, the quotient from a previous execution is shifted left one character position so that the quotient will be contained in the N least-significant character positions of the Accumulator after N executions of the instruction. The last execution of the instruction determines the sign of the quotient which is stored in the zone bits of the least-significant quotient character. A single Instruction 26 can be used to form an $8+N$ character result, where N is the number of characters in the quotient, by controlling the re-execution of Instruction 26 with Instruction 16, Branch on Count, that has been set to allow N executions of Instruction 26.

Two restrictions are placed on the magnitude of the eight-character divisor. The divisor must have an absolute magnitude of at least 100,000. In addition, the eight-character divisor must have an absolute magnitude which is equal to or greater than the absolute magnitude of the eight most-significant characters of the dividend. If either of the above conditions is not satisfied, the Overflow flip-flop is set to the 1-state and the execution of the instruction is not completed.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered. The W01 block of the initial routine illustrated in FIG. 121 is unaltered, if development of the address field of the instruction word is required.

Upon completion of the initial routine, the Program Processor performs first a shift operation and then a divide operation to effect execution of the instruction. The W-Selector Register and TL-Counter are set to define blocks W15R, W04R, W11S, W03S and W04S, in that order, and the Program Processor performs the micro-operations of this sequence of blocks four times to shift the contents of the Accumulator left one character position. After the shift of accumulator word A, the Program Processor performs blocks W02R and W04S to store the most-significant dividend character, previously shifted out of the Accumulator into memory location 10 (octal), into the CC-Register to complete the shift sequence. The W-Selector Register and the TL-Counter are then set to define blocks W02, W07, W03S, W02, W07, W04S, W03R, W06S and W15S to continue execution of the instruction. During this sequence, the divide operation is effected and the trial quotient character and its sign is developed. If the number of subtractions of the divisor from the dividend was not excessive, the Program Processor next performs block W03 to store the quotient and goes to block W00, preparatory to execution of the next instruction. If the number of subtractions of the divisor from the dividend was excessive, the Program Processor next performs W04, W05, W15S, W04R, W05, W04S and W03, in that order, to correct the trial quotient and the remainder.

The following table summarizes the macro-operation which is performed during each block of Instruction 26, including the blocks of both the shift and the divide sequences.

Block:	Macro-operation
W15R	Count of three established in CC-Register to control left shift through one character position. Address of most-significant divisor word formed in D-Register and address of most-significant dividend word (accumulator word D) in A-Register and Accumulator Counter Register.
W04R	Accumulator word D read from Memory into M-Register. Shift count in CC-Register transferred to N-Register.

Block:	Macro-operation
W11S --	Accumulator word D transferred to E-Register and shifted right three character positions, shifting characters 1, 2, and 3 of accumulator word D into the character 0, 1 and 2 positions respectively of the M-Register.
W03S --	Characters 1, 2 and 3 of accumulator word D stored in the character 0, 1 and 2 positions respectively of the accumulator word D location in Memory. Partial parity for accumulator word D location stored in flip-flop.
W04S --	Character 0 of accumulator word D stored in memory location 10 (octal).
W04R -	Accumulator word C read from Memory into the M-Register.
W11S --	Accumulator word C transferred to E-Register and shifted right three character positions, shifting characters 1, 2 and 3 of accumulator word C into the character 0, 1 and 2 positions respectively of the M-Register.
W03S --	Characters 1, 2 and 3 of accumulator word C stored in the character 0, 1 and 2 positions respectively of the accumulator word C location in Memory. Partial parity for the accumulator word C location stored in flip-flop.
W04S --	Character 0 of accumulator word C stored in the character 3 position of the accumulator word D location in Memory and parity bit stored in accumulator word D location.
W04R -	Accumulator word B read from Memory into the M-Register.
W11S --	Accumulator word B transferred to E-Register and shifted right three character positions, shifting characters 1, 2 and 3 of accumulator word B to the character 0, 1 and 2 positions respectively of the M-Register.
W03S --	Characters 1, 2 and 3 of accumulator word B stored in the character 0, 1 and 2 positions respectively of the accumulator word B location in Memory. Partial parity for the accumulator word B location stored in flip-flop.
W04S --	Character 0 of accumulator word B stored in the character 3 position of the accumulator word C location in Memory and parity bit stored in the accumulator word C location.
W04R -	Accumulator word A read from Memory into the M-Register.
W11S --	Accumulator word A transferred to E-Register and shifted right three character positions, shifting characters 1, 2 and 3 of accumulator word A into the character 0, 1 and 2 positions respectively of the M-Register. If flip-flop MPL is set to the 1-state indicating that Instruction 26 is being executed for the first time for a given dividend and divisor, the shift is an alpha shift which causes the sign of the least-significant character of the dividend to be shifted. If flip-flop MPL is reset to the 0-state, the shift is a decimal shift which causes the zone bits of the least-significant accumulator character to be reset to zeros before shifting is initiated.

Block:	Macro-operation
W03S --	Characters 1, 2 and 3 of accumulator word A stored in the character 0, 1 and 2 positions respectively of the accumulator word A location in the Memory. Partial parity for the accumulator word A location stored in flip-flop.
W04S --	Character 0 of accumulator word A stored in the character 3 position of the accumulator word B location in Memory and parity bit stored in the accumulator word B location.
W02R -	Character 0 of accumulator word D (most-significant dividend character) read from memory location 10 (octal) into the M-Register.
W04S --	Flip-flop COR reset, indicating the end of the shift sequence and the beginning of the divide sequence of Instruction 26.
W02 ---	Most-significant dividend character transferred to CC-Register and from CC-Register to N-Register. Most-significant divisor word read from one of specified memory locations and stored in E-Register. Overflow flip-flop FVO set if absolute magnitude of divisor is less than 100,000.
W07 ---	Most-significant dividend character stored in Q-Register. CC-Register cleared. Most-significant divisor word repeatedly subtracted from most-significant dividend word, increasing count in CC-Register by one for each subtraction and reducing count in N-Register by one for each borrow. The repetitive subtraction is repeated until the count in the CC-Register equals 11 or until the count in the N-Register is equal to zero and another borrow occurs. If the count in the CC-Register goes to 11, the absolute magnitude of the divisor is greater than the most-significant eight characters of the dividend and instruction execution is terminated. When the count in the N-Register is equal to zero and another borrow occurs, the CC-Register contains the trial quotient and the most-significant remainder word is in the M-Register.
W03S --	Most-significant remainder word stored in the accumulator word D location in Memory. If Overflow flip-flop FVO is set to the 1-state, indicating that the trial quotient was 11 or that the divisor was less than 100,000, the Program Processor next goes to block W00, preparatory to execution of the next instruction. Otherwise, Program Processor goes to block W02.
W02 ---	Least-significant divisor word read from Memory and stored in the E-Register. Trial quotient transferred from CC-Register to N-Register. Sign of the divisor stored in Sign flip-flop SIN.
W07 ---	Least-significant dividend word read from Memory and stored in M-Register. Trial quotient stored in Q-Register and CC-Register cleared. Least-significant divisor word repeatedly subtracted from least-significant dividend word, reducing trial quotient count in the N-Register by one for each subtraction until the count in the N-Register is equal to zero. Count in the CC-Register advanced by one for each

Block:	Macro-operation
	borrow. When the count in N-Register is equal to zero, the least-significant remainder word is in the M-Register and the borrow count is in the CC-Register. The sign of the quotient is stored in Sign flip-flop SIN.
W04S --	The least-significant remainder word is stored in the accumulator word C location in Memory.
W03R -	The most-significant remainder word in the accumulator word D location is read from Memory into the M-Register.
W06S --	The borrow count in the CC-Register is subtracted from the most-significant remainder word to modify the most-significant remainder word.
W15S --	The result of the subtraction of the borrow count from the most-significant remainder word is stored in the M-Register and flip-flop CRE is set to the 1-state, if a carry was propagated from character 0 of the result, indicating that the most-significant remainder is greater than the borrow count. The modified most-significant remainder word is then stored in the accumulator word D location in Memory and the trial quotient is transferred from the Q-Register to the N-Register. The Program Processor then goes to block W03 to store the quotient and the sign in the least-significant character position of the Accumulator to complete execution of the instruction. If flip-flop CRE is reset to the 0-state, indicating that the carry count is greater than the most-significant remainder, the divisor must be added to the remainder once and the trial quotient reduced by one to obtain the correct quotient and remainder. The modified most-significant remainder word is stored in the accumulator word D location in Memory. The trial quotient in the N-Register is reduced by one, since FCRE indicates that the divisor was subtracted from the dividend one too many times.
W04R -	The least-significant remainder word is read from the accumulator word C location into the M-Register.
W05 ---	The least-significant divisor word is added to the least-significant remainder word.
W15S --	The corrected least-significant remainder word is stored in the accumulator word C location in Memory. Flip-flop CRE is set to the 1-state if a carry occurred.
W04R -	The most-significant remainder word is read from the accumulator word D location into the M-Register.
W05 ---	The most-significant divisor word and the previous carry, if any, is added to the most-significant remainder word.
W04S --	The corrected most-significant remainder word is again stored in the accumulator word D location in Memory.
W03 ---	The corrected quotient and the sign of the quotient is stored in the least-significant character position of the accumulator word A location in Memory. Instruction execution is terminated and the Program Processor goes to block W00.

As indicated in the above table, the Accumulator is shifted left through one character position during each execution of the instruction to correctly position the quotient, if the instruction has previously been executed, and the dividend. The sign of the divisor and the sign of the dividend character in the character 0 position of accumulator word B, during the last execution of the instruction.

The sequence of occurrence of the micro-operations in the blocks required to execute Instruction 26 is illustrated in the timing diagram of FIG. 160. The micro-operations of the blocks in the shift sequence of Instruction 26 will not be described in detail since they are the same as the micro-operations required to circular shift the four-word Accumulator left through one character position, as described in the section entitled "Instruction 22: Shift (SHG)." During block W15R at the beginning of the shift sequence, signal DDCD issues to reduce the count in the D-Register by one, forming the address of the most-significant divisor word. During the last W04S block of the shift sequence, flip-flop COR is reset to the 0-state to indicate the end of the shift sequence of Instruction 26 and the beginning of the divide sequence.

With reference to block W02R of FIG. 160, signal DDBX issues during time period TL0 to transfer the address of the most-significant divisor word from the D-Register to the B-Register of Memory. Flip-flop MRD is set to the 1-state to initiate a read operation in Memory which transfers the most-significant divisor word into the M-Register. Working clock signal QSCX issues at this time to transfer the most-significant character of the dividend (character 0 of accumulator word D), which is in the M-Register, through the Adder to the CC-Register. During time period TL1, signal DCLQ issues to clear the Q-Register and signals DCM0–DCM3 issue to clear the M-Register, preparatory to receiving the most-significant divisor word during the read operation. Flip-flop MSX is set to the 1-state at this time to apply the most-significant divisor word to the Adder as soon as it is received in the M-Register during the read operation. Working clock signal QCNX issues to transfer the most-significant dividend character in the CC-Register to the N-Register. Block W02R is terminated and the Program Processor goes to the second half-sequence of the W02 block, viz block W02S.

During time period TL2 of block W02S, flip-flop MWR is set to the 1-state to initiate a write operation which restores the most-significant divisor word to Memory. During time period TL4, flip-flop FVO is set to the 1-state if the absolute magnitude of the divisor in the specified adjacent memory locations is less than 100,000. Working clock signal QSEX issues at this time to transfer the most-significant divisor word through the Adder to the E-Register. Flip-flop MPL is reset to the 0-state at this time if Overflow flip-flop FVO is set to the 1-state. Flip-flop MSX is reset to the 0-state and block W02S is terminated. The Program Processor next goes to block W07.

During time period TL0 of block W07, signal DABX issues to transfer the address of accumulator word D from the A-Register and the Accumulator Counter Register to the B-Register of Memory. Flip-flop MRD is set to the 1-state to initiate a read operation which transfers the most-significant dividend word into the M-Register. Working clock signal QCER forms the complement of the most-significant divisor word in the E-Register. During time period TL1, flip-flop DCE is set to the 1-state to cause the Adder to operate in the decimal mode. Flip-flop MPL is set to the 1-state at this time if Overflow flip-flop FVO is reset to the 0-state. The most-significant dividend character is transferred from the N-Register to the Q-Register at this time. Signals DCM0–DCM3 and DCLC issue to clear the M- and CC-Registers respectively. Flip-flops MSX and ESX are set to the 1-state to apply the most-significant dividend word and the complement of the most-significant divisor word to the Adder, effecting the subtraction of the divisor word from the dividend

word. Carry flip-flop CAY is set to the 1-state to add one to the complement of the divisor word in the Adder. The flip-flops of the TB-Counter are reset to the 0-state and the first half-sequence of block W07 terminates.

During the second half-sequence of block W07, flip-flop MOD is set to the 1-state to initiate the division. Signal FMOD causes signal DBCU to issue. During time period TL3, signal DBCU sets flip-flop TB0 to the 1-state, advancing the count in the TB-Counter to one. During time period TL4, the MPL and CRE flip-flops are reset to the 0-state and the count in the TB-Counter is advanced to two. During time period TL5, working clock signal QTLP is inhibited by signal FMOD, inhibiting the advance of the TL-Counter. The first subtraction of the most-significant divisor word from the most-significant dividend word is completed at this time and signals QSMA, QSMB and QSMC issue to transfer the result to the M-Register, clearing the previous contents of the M-Register. Flip-flop CRE is set to the 1-state if a carry was propagated from character 0 of the result. Signal DCCU issues to advance the count in the CC-Register by one, to form the trial quotient. Working clock signal QNCD issues, if flip-flop CRE is reset to the 0-state indicating a borrow, to reduce the most-significant dividend character in the N-Register by one.

The flip-flops of the TB-Counter are reset to the 0-state and the subtraction of the most-significant divisor word from the contents of the M-Register is repeated. The count in the CC-Register is advanced by one for each subtraction and the most-significant dividend character in the N-Register is reduced by one for each borrow. Division by repeated subtraction continues until the count in the N-Register is reduced to zero and another borrow occurs. At this time, flip-flop MOD is reset to the 0-state and the count in the TL-Counter is advanced to TL0, terminating the W07 block. Flip-flop FVO is set to the 1-state during time period TL0 if the count in the CC-Counter is 11, indicating that the necessary condition of the divisor being greater than the most-significant eight characters of the dividend has not been satisfied. The Program Processor next performs block W03S.

Signal DABX issues during time period TL2 of block W03S to transfer the address of the accumulator word D location to the B-Register of Memory. Flip-flops ESX and MSX are reset to the 0-state. Flip-flop MWR is set to the 1-state to initiate a write operation which stores the most-significant remainder word in the M-Register into the accumulator word D location in Memory. Flip-flop DSX is set to the 1-state during time period TL3 to apply the address of the most-significant dividend word to the Adder. Flip-flop CAY is set to the 1-state to add one to the address in the Adder, forming the address of the least-significant dividend word. Working clock signal QSDX issues during time period TL5 to transfer this address to the D-Register. Signal DACD issues during time period TL0 to form the address of accumulator word C in the A-Register and the Accumulator Counter Register. Block W03S is terminated and the Program Processor goes to block W02 to continue execution of the instruction.

During time period TL0 of block W02R, signal DDBX transfers the address of the least-significant divisor word to the B-Register of Memory. Flip-flop MRD is set to the 1-state to initiate a read operation which transfers the least-significant divisor word into the M-register. The M-register is cleared, preparatory to receiving the least-significant divisor word, by signals DCM0–DCM3. Signal DCLQ clears the Q-register. Flip-flop DSX is reset to the 0-state. Flip-flop MSX is set to the 1-state to apply the least-significant divisor word to the Adder as soon as it is received in the M-Register during the read operation. Working clock signal QCNX issues to transfer the trial quotient in the CC-Register to the N-Register. Block W02R terminates and the Program Processor goes to block W02S.

During time period TL2 of block W02S, signal DDBX issues and flip-flop MWR is set to the 1-state to initiate a write operation which restores the least-significant divisor word to Memory. Sign flip-flop SIN is set to the 1-state at this time if the sign of the divisor, as contained in the zone bits of character 3 of the least-significant divisor word in the M-Register, is minus. Working clock signal QSEX during time period TL4 transfers the least-significant divisor word to the E-Register. Flip-flop MSX is reset to the 0-state and block W02 is terminated, the Program Processor next going to block W07 to continue the divide sequence.

Signal DABX issues and flip-flop MRD is set to the 1-state during time period TL0 of block W07 to address the least-significant dividend word and to initiate a read operation to transfer the least-significant dividend word to the M-Register. Working clock signal QCER complements the least-significant divisor word in the E-Register. During time period TL1, flip-flop DCE is set to the 1-state to establish the decimal mode of operation in the Adder. The trial quotient in the N-Register is transferred to the Q-Register and the M- and CC-Registers are cleared by signals DCM0–DCM3 and DCLC respectively. Flip-flops MSX and ESX are set to the 1-state to apply the least-significant dividend word and the complement of the least-significant divisor word to the Adder. Flip-flop CAY is also set to the 1-state to add one to the complement of the least-significant divisor word in the Adder. The flip-flops of the TB-Counter are reset to the 0-state, preparatory to performance of the division. Signal $\overline{DS55}$ issues during time period TL2 if the zone bits of character 3 of the least-significant dividend word in the M-Register establish the sign of the dividend as minus. Signal $\overline{DS55}$ causes the remainder resulting from the division to retain the sign of minus. Flip-flop MOD is set to the 1-state to initiate the division, causing signal DBCU to issue.

The sign of the quotient is stored in Sign flip-flop SIN during time period TL3 of block W07. If the signs of the divisor and dividend are both plus or if the sign of the divisor is minus and the sign of the dividend is plus, the state of the Sign flip-flop SIN, as previously determined during block W02, remains unchanged to establish the sign of the quotient. However, if the sign of the divisor is plus and the sign of the dividend, is minus, the Sign flip-flop SIN is set to the 1-state to render the sign of the quotient minus. Similarly, if the sign of the divisor is minus and the sign of the dividend is also minus, the Sign flip-flop SIN is reset to the 0-state to establish the sign of the quotient as plus.

Signal DBCU advances the count in the TB-Counter to one and flip-flop CRE is reset to the 0-state, preparatory to storing borrows during performance of the division. The count in the TB-Counter is advanced to two and working clock signal QTLP is inhibited by signal FMOD, preventing the advance of the TL-Counter from TL5. With a count of two in the TB-Counter, the result of the subtraction of the least-significant divisor word from the least-significant dividend word is transferred to the M-Register. This transfer clears the previous contents of the M-Register. Flip-flop CRE is set to the 1-state if a carry is propagated from character 0 of the result. Signal DCCU issues to advance the count in the CC-Counter by one each time that a carry is not propagated from character 0 of the result, i.e. each time that a borrow occurs. Working clock signal QNCD issues after each subtraction to reduce the trial quotient in the N-Register by one. The flip-flops of the TB-Counter are reset to the 0-state and another subtraction of the least-significant divisor word from the contents of the M-Register is performed. The repetitive subtractions continue until the count in the N-Register is reduced to zero, at which time flip-flop MOD is reset to the 0-state and the TL-Counter advances to TL0. Block W07 is terminated with the bor-

row count stored in the CC-Register, the trial quotient stored in the Q-Register, and the least-significant remainder word stored in the M-Register. The Program Processor goes to block W04S to continue execution of the instruction.

Signal DABX issues during time period TL2 of block W04S and flip-flop MWR is set to the 1-state to initiate a write operation which transfers the least-significant remainder word in the M-Register to the accumulator word C location in Memory. Flip-flops DCE, ESX, CAY and MSX are reset to the 0-state and the count in the Accumulator Counter Register is advanced by signal DACU to address the accumulator word D location. Block W04S terminates and the Program Processor goes to block W03R.

Signal DABX transfers the address of the accumulator word D location to the B-Register of Memory during time period TL0 of block W03R. Flip-flop MRD is set to the 1-state to initiate a read operation which transfers the most-significant remainder word in the accumulator word D location into the M-Register. Flip-flop CS2 is set to the 1-state to apply the carry count in the CC-Register to the Adder. The M-Register is cleared by signals DCM0–DCM3, preparatory to receiving the most-significant remainder word from Memory during the read operation. Block W03R terminates with the most-significant remainder word in the M-Register and the Program Processor goes to block W06S to continue instruction execution.

Working clock signal QSEX issues during time period TL2 of block W06S to transfer the borrow count from the CC-Register to the E-Register. Flip-flop DCE is set to the 1-state to cause the Adder to operate in the decimal mode and flip-flop CS2 is reset to the 0-state. Working clock signal QCER complements the borrow count in the E-Register. Flip-flop ESX is set to the 1-state to apply the complement of the borrow count in the E-Register to the Adder, flip-flop CAY being set to the 1-state to add one to the borrow count complement in the Adder. Flip-flop MSX is set to the 1-state to apply the most-significant remainder word in the M-Register to the Adder. If the borrow count is greater than zero, the borrow count is then subtracted from the most-significant remainder word. Flip-flop CRE is reset to the 0-state and the Program Processor goes to block W15S to continue performance of the divide sequence.

Signal DABX transfers the address of the accumulator word D location to the B-Register of the Memory and flip-flop MWR is set to the 1-state to initiate a write operation which transfers the modified most-significant remainder word to Memory during time period TL2 of the block W15S. Working clock signals QSMA, QSMB and QSMC issue to transfer the modified most-significant remainder word from the adder output to the M-Register, preparatory to storing it in the Memory during the write operation. Flip-flop CRE is set to the 1-state if a carry is propagated from character 0 of the adder output. Working clock signal QQNX issues to transfer the trial quotient from the Q-Register to the N-Register.

If flip-flop CRE is reset to the 0-state, indicating a borrow when the borrow count was subtracted from the most-significant remainder word, the borrow count in the CC-Register was greater than the most-significant remainder word. This indicates that the trial quotient is excessive by a quantity of one and that the divisor must be added to the remainder to obtain the true remainder. In this event, working clock signal QNCD issues to reduce the trial quotient by one and signal DACD issues to form the address of the accumulator word C location in the A-Register and the Accumulator Counter Register. If CRE is set to the 1-state, signal DACU issues to form the address of the accumulator word D location in the A-Register and the Accumulator Counter Register. Flip-flops ESC, MSX, PSX and CAY are reset to 0-state. Block W15S terminates. The Program Processor next performs block W03, if flip-flop CRE is set to the 1-state. If flip-flop CRE is reset to the 0-state, the Program Processor next performs block W04R.

Assuming that flip-flop CRE is set to the 1-state, the Program Processor goes to block W03 to complete execution of the instruction. Signal DABX transfers the address of the accumulator word A location to the B-Register of Memory during time period TL0 of block W03R. Flip-flop MRD is set to the 1-state at this time to initiate a read operation which transfers the contents of the accumulator word A location to the M-Register. Working clock signal QNCX transfers the final quotient from the N-Register to the CC-Register. Flip-flop CS2 is set to the 1-state to apply the final quotient in the CC-Register to full adder circuits S00–S03 of the Adder. Signal $\overline{D855}$ is a binary 1 if Sign flip-flop SIN is set to the 1-state, indicating that the sign of the quotient is minus, or is otherwise a binary 0 to insert a plus sign in the quotient. Working clock signal QSMA transfers the quotient character and the zone bits through the Adder to flip-flops M00–M05 of the M-Register. This transfer clears the former contents of flip-flops M00–M05. Flip-flop MWR is set to the 1-state to initiate a write operation which stores the quotient character and its sign along with the former contents of the character 0, 1 and 2 positions of the accumulator word A location into the accumulator word A location in Memory. Flip-flop PSX is set to the 1-state to apply the address of the next instruction in the P-sequence to the Adder and the Program Processor goes to block W00, preparatory to execution of the next instruction.

Assuming that flip-flop CRE is reset to the 0-state, the Program Processor next performs block W04R. Signal DABX transfers the address of the accumulator word C location to the B-Register of Memory during time period TL0 of block W04R. Flip-flop MRD is set to the 1-state to initiate a read operation which transfers the least-significant remainder word from the accumulator word C location into the M-Register of Memory. Signals DCM0–DCM3 clear the M-Register, preparatory to receiving the least-significant remainder word from Memory during the read operation. Flip-flop MSX is set during time period TL1 to apply the least-significant remainder word to the Adder, as soon as it is received in the M-Register during the read operation. Flip-flop ESX is reset to the 0-state and block W04R is terminated. The Program Processor next performs block W05.

Signal DDBX issues during time period TL0 of block W05 to transfer the address of the least-significant divisor word to the B-Register of Memory. Flip-flop MRD is set to the 1-state to initiate a read operation which transfers the least-significant divisor word into the M-Register. Working clock signal QSEX transfers the least-significant remainder word through the Adder to the E-Register. Flip-flop DCE is set to cause the Adder to operate in the decimal mode. The M-Register is cleared by signals DCM0–DCM3, preparatory to receiving the least-significant divisor word from Memory during the read operation. Flip-flop LSN is set to the 1-state so that the addition of the least-significant divisor word and the least-significant remainder word in the Adder is a like-signs addition. Flip-flop ESX is set to the 1-state to apply the least-significant remainder word in the E-Register to the Adder. The first half-sequence of the W05 block is terminated and the second half-sequence is initiated. Flip-flop MWR is set to the 1-state during time period TL2 of the second half-sequence to initiate a write operation which restores the least-significant divisor word to Memory. Block W05 is terminated and the Program Processor goes to block W15S.

Signal DABX transfers the address of the accumulator word C location to the B-Register of Memory during time period TL2 of block W15S. Signal $\overline{D855}$ issues at this time if the sign of the least-significant remainder word in

the E-Register is minus, to insert the proper sign in the corrected remainder word at the output of the Adder. Working clock signals QSMA, QSMB and QSMC transfer the corrected least-significant remainder word from the adder output to the M-Register, preparatory to storing the corrected least-significant remainder word in Memory during the write operation. Flip-flop MWR is set to the 1-state to initiate a write operation which stores the corrected least-significant remainder word in the accumulator word C location of Memory. Flip-flop CRE is set to the 1-state if a carry is propagated from character 0 of the adder output. Flip-flops ESX, MSX, PSX and CAY are reset to the 0-state and signal DACU issues to increase the count in the Accumulator Counter Register by one to form the address of the accumulator word D location. Block W15S terminates and the Program Processor next performs block W04R.

Signal DABX during time period TL0 of block W04R transfers the address of the accumulator word D location to the B-Register of Memory. Flip-flop MRD is set to the 1-state to initiate a read operation which transfers the most-significant remainder word from Memory to the M-Register. Signals DCM0–DCM3 clear the M-Register, preparatory to receiving the most-significant remainder word from Memory during the read operation. Flip-flop MSX is set to the 1-state to apply the most-significant remainder word to the Adder, when it is received in the M-Register during the read operation. Flip-flops DSX and ESX are reset to the 0-state and block W04R terminates. The Program Processor next performs block W05.

Signal DDBX issues and flip-flop MRD is set in the 1-state during time period TL0 of block W05 to address the most-significant divisor word and to initiate a read operation which transfers the most-significant divisor word from Memory to the M-Register. Working clock signal QSEX transfers the most-significant remainder word into the E-Register. Flip-flop DCE is set to the 1-state to establish the decimal mode of operation in the Adder. The M-Register is cleared by signals DCM0–DCM3 during time period TL1, preparatory to receiving the most-significant divisor word from Memory during the read operation. Flip-flop LSN is set to the 1-state, to render the addition of the most-significant divisor word and the most-significant word a like-signs addition. Carry flip-flop CAY is set to the 1-state at this time if flip-flop CRE is set to the 1-state, adding a carry from the earlier addition of the least-significant divisor word and the least-significant remainder word to the sum of the most-significant divisor word and the most-significant remainder word. Flip-flop ESX is set to the 1-state to apply the most-significant remainder word in the E-Register to the Adder. The adder output is the corrected most-significant remainder word. The first half-sequence in block W05 is terminated and the second half-sequence is initiated. Flip-flop MWR is set to the 1-state at time period TL2 of the second half-sequence to initiate a write operation which restores the most-significant divisor word to Memory. Block W05 is terminated and the Program Processor goes to block W04S to continue instruction execution.

Signal DABX issues and flip-flop MWR is set to the 1-state during time period TL2 of block W04S to address the accumulator word D location and to initiate a write operation which stores the corrected most-significant remainder word to Memory. Working clock signals QSMA, QSMB and QSMC issue at this time to transfer the corrected most-significant remainder word from the adder output to the M-Register, preparatory to performance of the write operation. Flip-flops DCE, ESX, CAY and MSX are reset to the 0-state and the count in the Accumulator Counter Register is advanced by signal DACU to address the accumulator word A location. Block W04S terminates and the Program Processor next performs block W03 to complete execution of the instruction.

During W03, the contents of the accumulator word A location are read from Memory into the M-Register and the final quotient and its sign are inserted in the flip-flops M00–M05. The contents of the M-Register are then stored in the accumulator word A location in Memory, as previously described following the first description of block W15S.

FIG. 161 is a flow diagram of Instruction 26, illustrating the sequence of operations performed in executing the instruction. Referring to FIG. 161, the instruction word is read from Memory and the state of flip-flop MPL is checked. If flip-flop MPL is set to the 1-state, indicating that the instruction is being performed for the first time with the given divisor and dividend, the Accumulator undergoes an alpha circular shift left through one character position. The alpha shift preserves the sign of character 3 of accumulator word A as it is shifted into the character 2 position of accumulator word A. If flip-flop MPL is reset to the 0-state, the left shift of the Accumulator through one character position is a decimal shift. The decimal shift causes the sign bits to be stripped from the quotient character in the character 3 position of the accumulator word A location prior to shifting the quotient character into the character 2 position. Since the sign of the last quotient character determines the sign of the entire quotient, the decimal shift removes unnecessary sign information from the quotient.

Character 3 of accumulator word D, which is shifted out of the Accumulator, is the most-significant character of the dividend and is stored in the CC-Register. The most-significant divisor word is then read from Memory and stored in the E-Register. The magnitude of the divisor is checked at this time. If the most-significant divisor word is less than 10, Overflow flip-flop FVO is set to the 1-state and the Program Processor terminates instruction execution. If the most-significant divisor word is equal to or greater than 10, the most-significant dividend word is read from Memory and stored in the M-Register. The most-significant divisor word is repeatedly subtracted from the most-significant dividend word, the count in the CC-Register being advanced by one for each subtraction and the most-significant dividend character in the N-Register being reduced by one for each borrow. If the count in the CC-Register reaches 11, the Overflow flip-flop FVO is set to the 1-state and the Program Processor terminates instruction execution. When the count in the N-Register reaches zero and another borrow occurs, the repetitive subtraction is stopped and the contents of the M-Register, which comprise the most-significant remainder word, are stored in the accumulator word D location in the Memory. The trial quotient in the CC-Register is stored in the N- and Q-Registers and the CC-Register is cleared.

The least-significant divisor word and the least-significant dividend word are next read from Memory and the least-significant divisor word is repeatedly subtracted from the least-significant dividend word a number of times equal to the value of the trial quotient in the N-Register. The sign of the divisor is stored in Sign flip-flop SIN. During the repeated subtractions, the count in the CC-Register is advanced by one for each borrow. When the count in the N-Register is reduced to zero, the M-Register contains the least-significant remainder word and the CC-Register contains a borrow count. The sign of the divisor and dividend are then compared and the sign of the quotient is stored in Sign flip-flop SIN.

The most-significant remainder word is read from the accumulator word D location in Memory and stored in the M-Register. The borrow count in the CC-Register is subtracted from the most-significant remainder word, with the result being stored in the accumulator word D location in Memory. If a carry occurred during the subtraction, the remainder in the two most-significant accumulator word locations and the quotient in the Q-Regis-

ter are final. Accumulator word A is read from Memory and the quotient and its sign are inserted in the character 3 position of accumulator word A prior to restoration of accumulator word A to Memory.

If no carry was propagated, i.e. if a borrow occurred, when the borrow count in the CC-Register was subtracted from the most-significant remainder word, the divisor has been subtracted from the dividend one too many times and the quotient and remainder must be corrected. The quotient in the N-Register is reduced by one to form the final quotient and the least-significant divisor word is added to the least significant remainder word. The result is stored in the accumulator word C location and any carry that occurred during the addition is remembered. The most-significant divisor word and the carry, if any, are then added to the most-significant remainder word. The corrected most-significant remainder word is stored in the accumulator word D location in Memory. Accumulator word A is then read from Memory and the corrected quotient and its sign is inserted in the character 3 position prior to restoration of accumulator word A to Memory. The Program Processor then goes to block W00, preparatory to execution of the next instruction.

Instruction 05: Edit (EDT)

This instruction causes the data characters of a memory field to be processed from right to left under control of format characters in the working Accumulator. The location of the least-significant data word of the memory field is specified by the instruction address field. The result, which contains a number of characters equal to the number of format characters (4, 8, 12 or 16) in the working Accumulator, is stored in the working Accumulator. The contents of the memory field and the accumulator working length are not changed.

Each format character causes the placement of a character or space in one character position of the result. The character placed in the character position of the result may be either a format character or a data character. Certain format characters can mark the corresponding result character and subsequent result characters as tentative candidates for suppression. Three types of suppression can occur under control of format characters:

(a) conversion of non-significant zeros, commas and periods to spaces (simple suppress).
(b) conversion of non-significant zeros, commas and periods to asterisks (suppress and asterisk protect).
(c) conversion of non-significant zeros, commas and periods to spaces, except that the right-most non-significant zero, comma or period is converted to a dollar sign (suppress and float $).

Each of the above types of suppression require a two-phase operation to execute the instruction. The first phase, called the edit mode, produces a tentative result. The second phase, called the suppress mode, converts the tentative result to the final result. If there are no suppression format characters in the format, only the edit mode is necessary to complete execution of the instruction. In this event, the tentative result is the final result and the suppress mode is not entered.

In the edit mode, editing of the data characters by the format characters proceeds from right to left, one character position at a time, with the right-most format character and the right-most data character being considered first. In each case, the result character is determined by the format character which may fall into one of four classes, viz. use, ignore, sign and insert. Flip-flops CL1 and CL2 are preset to particular states in response to each format character to define the class of the format character as indicated in the following table:

FORMAT CHARACTER CLASS

Class	Characters	FCL1	FCL2
Use.....	; (octal 56) ~ (octal 12) > (octal 16) # (octal 13)	0	0
Ignore...	I (octal 17)	1	0
Sign.....	− (octal 52)	1	1
Insert....	Remaining 58 characters of set.	0	1

The format character of the ignore class (I octal 17) causes the format character to be retained in the result in place of the data character. The data character is ignored and discarded, the next format character being considered with the next data character. The format character of the sign class (—octal 52) causes the sign information contained in the zone bits of the least-significant data character of the memory field to determine the character to be placed in the result. If the sign of the memory field is plus, a space is placed in the result. If the sign of the memory field is minus, a dash (—) is placed in the result. The zone bits of the least-significant data character of the memory field are set to zero in the Program Processor. The least-significant data character is retained for consideration by the next format character. If the data character being considered is not the least-significant data character of the memory field, the format character (—octal 52) is inserted in the result and the data character is retained for consideration by the next format character. A format character of the insert class (any one of the 58 characters of the 64 character set not used as a format character of the use, ignore or sign classes) causes the format character to be inserted into the result. The data character is retained for consideration by the next format character.

A format character of the use class causes the data character to be used in the tentative result in place of the format character. There are four distinct types of format characters in the use class, viz, use, simple suppress, suppress and float $ and suppress and asterisk protect. Flip-flops SP1 and SP2 are preset to predetermined states in accordance with the type of format character of the use class being employed to process a data character as indicated in the following table:

SUPPRESS FORMAT CHARACTER TYPE

Type	Character	FSP2	FSP1
Simple suppress.....	~ (octal 12)	0	1
Suppress and asterisk protect.	> (octal 16)	1	0
Suppress and float $..	# (octal 13)	1	1

The "use" format character (; octal 56) of the use class does not change the states of flip-flops SP1 and SP2. The "use" format character simply causes the data character to be used in the tentative result in place of the format character. The "simple suppress" format character (~octal 12), in addition to specifying that the data character be used in the tentative result in place of the format character, causes flip-flops SP2 and SP1 to assume the 0- and 1-states respectively, indicating that simple suppression is to occur, starting with this tentative

result character. Simple suppression causes a candidate for suppression in the tentative result which is a zero, a period or a comma to be replaced by a space during the suppress mode.

The "suppress and asterisk protect" format character (>octal 16) specifies that the data character be used in the tentative result in place of the format character and also causes flip-flops SP2 and SP1 to assume the 1- and 0-states respectively, indicating that "suppress and asterisk protect" is to occur starting with this tentative result character. "Suppress and asterisk protect" causes a zero, a period or a comma in the tentative result to be replaced by an asterisk during the suppress mode.

The "suppress and float $" format character (#octal 13), in addition to specifying that the data character be used in the tentative result in place of the format character, causes flip-flops SP1 and SP2 to be set to the 1-state, indicating that "suppress and float $" is to occur starting with this tentative result character. During "suppress and float $," the right-most non-significant zero, comma or period in the tentative result is converted to a dollar sign ($) and simple suppression thereafter occurs, causing remaining zeros, periods and commas to be replaced by spaces during the suppress mode.

FIGURE 162 illustrates how each of the format characters affects the tentative result during the edit mode.

Once one of the three suppress type format characters (~octal 12, >octal 16, or #octal 13) is employed during the edit mode, the suppression counter (CC-Register) initially set to zero, is affected by every character placed in the tentative result. One is added to the count in the suppression counter if the format character is octal 12, 13, 16 or 56 and the data character is a zero, or if the format character is not octal 12, 13, 16 or 56. The count in the suppression counter is reduced to zero if the format character is octal 12, 13, 16 or 56 and the data character is not a zero.

The suppress mode is not entered unless one of the three suppress type format characters is employed during the edit mode and the count in the suppression counter is not equal to zero. If more than one type of suppression format character occurs in the format, suppression begins during the suppress mode in the character position of the result specified by the first (right-most) suppression format character; the type of suppression to be performed during the suppress mode is determined by the last (left-most) suppression format character. At the beginning of the suppress mode, the count in the suppression counter indicates the number of characters in the tentative result, counting from the left, which are candidates for suppression. The characters in these positions are examined during the suppress mode from right to left and any zeros, commas or periods in these positions are suppressed according to the type of suppression being performed. No other characters in the candidate positions of the result are changed during the suppress mode.

The following table indicates the changes made in the tentative result during the suppress mode to obtain the final result:

SUPPRESS MODE

Tentative Result Character	Type of Suppression	Final Result Character	Other Actions
Zero, period or comma.	Simple.....	Space.....	
Zero, period or comma.	Float $.....	$.....	Change to Simple Suppression.
Zero, period or comma.	Asterisk Protect.	*.....	
Other.....		No change....	

The W00 block of the initial routine, illustrated in the FIG. 119, is unaltered. The W01 block of the initial routine illustrated in FIG. 121 is unaltered, if development of the address field of the instruction word is required.

Upon completion of the initial routine, the W-Selector Register and the TL-Counter are preset to define blocks W02, W14, W03S and W15S and the Program Processor performs the micro-operations of these blocks to execute the instruction. Immediately after completion of the initial routine, the Program Processor performs the micro-operations of blocks W02 and W14, in that order. During block W02, the least-significant data word is transferred from Memory to the E-Register. During block W14R, the least-significant format word from the accumulator word A location in Memory is transferred to the M-Register and the least-significant result character is determined, the format word in the M-Register and conditionally the data word in the E-Register being shifted right through one character position. The micro-operations of the W14 block are repeated until the last character of either the format word in the M-Register or the data word in the E-Register is processed. If the last character of the format word has been processed, the Program Processor goes to block W15S to store the result in the Accumulator. If the working Accumulator contains another higher-order accumulator word or if the last format word in the Accumulator has been processed and the suppress mode is required, the Program Processor again goes to block W14. If the last character of the data word or the last characters of both the data and the format words have been processed, the Program Processor performs the sequence of blocks W03S and W02 prior to returning to block W14. During the suppress mode, the same pattern of blocks is followed.

The sequence of micro-operations occurring during each of the above-identified blocks during execution of Instruction 05 is illustrated in the timing diagram of FIG. 163. Referring to block W02 of FIG. 163, signal DDBX issues during time period TL0 to transfer the address of a data word to the B-Register of Memory. Flip-flop MRD is set to the 1-state at this time to initiate a read operation which transfers the data word in the addressed memory location from Memory to the M-Register. Signals DCM0–DCM3 clear the M-Register during time period TL1, preparatory to receiving the data word from Memory during the read operation. Flip-flop MSX is set to the 1-state to apply the data word to the Adder, as soon as it is received in the M-Register during the read operation. The first half-sequence of the W02 block terminates and the second half-sequence is initiated by synchronizing signal DMDA from Memory.

Flip-flop MWR is set to the 1-state during time period TL2 of the second half-sequence of block W02 to initiate a write operation which restores the data word to Memory. Working clock signal QSEX issues during time period TL4 to transfer the data word through the Adder to the E-Register. The second half-sequence of block W02 terminates and the flip-flops of the W-Selector Register are preset to define the W14 block.

Signal DABX issues during time period TL0 of block W14 to transfer the address of an appropriate accumulator word location to the B-Register of Memory. Flip-flop MRD is set to the 1-state to initiate a read operation in Memory, which transfers the format word from the addressed accumulator word location to the M-Register. Signals DCM0–DCM3 clear the M-Register during time period TL1, preparatory to receiving the format word from Memory during the read operation. Signal DNS4 causes a count of four to be preset into the N-Register at this time, if the accumulator working length is single. If the accumulator working length is double, triple or quadruple and if the format word is being read from the accumulator word A location, indicating the first performance of block W14 during execution of the instruction,

signals DNS8, DNS2 or DNS6 respectively issue to preset counts of 8, 12 or 16 respectively into the N-Register. The count established in the N-Register at the beginning of instruction execution thus indicates the number of format characters to be employed and hence the number of result characters. Signal DCLQ also issues to clear the Q-Register. The first half-sequence of block W14 is terminated and the second half-sequence is initiated by synchronizing signal DMDA from Memory.

During time period TL2 of block W14, the states of flip-flops CL1 and CL2 are determined, in accordance with the class of the format character in flip-flops FM00–FM05 of the M-Register. If the Program Processor is in the edit mode, and if the format character is one of the four format characters of the use class, flip-flop ESX is set to the 1-state at this time to apply the data character in flip-flops E00–E05 to the Adder.

The states of flip-flops SP1 and SP2 are also determined at this time, if the format character is of the use class, in accordance with the type of the format character. If the Program Processor is in the suppress mode, the type of suppression is "suppress and float $" and the conditions are correct for inserting $ into the result, flip-flop SP2 is reset to the 0-state to change the type of suppression to "simple suppress."

Flip-flop SRI is set to the 1-state at this time during the suppress mode to initiate a shift operation. Working clock signal QSMA issues during the suppress mode if candidates for suppression are being processed and if the character in the tentative result is a zero, comma or period. Inputs DX00–DX05 to adders S00–S05 respectively provide the binary code for the appropriate character to be transferred to flip-flop M00–M05 by signal QSMA. Modification of the tentative result character is thus effected in accordance with the type of suppression. If simple suppression, the tentative result character is replaced by a space (010000), if suppress and asterisk protect, the tentative result character is replaced by an asterisk (101100) and if suppress and float $, the tentative result character is replaced with the dollar symbol (101011).

Signal DCLC issues during time period TL3 of block W14, during the edit mode, to reset the count of tentative candidates for suppression in the CC-Register to zero, if the format character is a suppression format character and the data character in flip-flops E00–E05 of the E-Register is not a zero. During the suppress mode, flip-flop Q00 of the Q-Register is set to the 1-state at this time to advance the count in the Q-Register to one. Signal DCCU issues during the edit mode to advance the count of candidates for suppression in the CC-Register by one if either flip-flop SP1 or SP2 has been set to the 1-state, indicating that the edit mode is to be followed by the suppress mode, and if either the data character in flip-flops E00–E05 of the E-Register is a zero or if the present format character in flip-flops M00–M05 of the M-Register is of the ignore, insert or sign class. Working clock signal QSHM issues at this time, during the suppress mode, to shift the contents of the M-Register right one character position. The state of flip-flop M23 is determined by the state of flip-flop M00 during the shift in the M-Register, causing the contents of the M-Register to be circular shifted.

During time period TL4, the count in the Q-Register is advanced to two during the suppress mode. During the edit mode, flip-flop SRI is set to the 1-state to initiate the shift operation. Working clock signal QSMA issues during time period TL4 during the edit mode to transfer the data character from flip-flops E00–E05 of the E-Register to flip-flops M00–M05 of the M-Register, if the format character is of the use class. If the format character is of the sign class, the data character is the least-significant data character and the sign in the zone bits of the data character is plus, signal QSMA transfers a space to flip-flops M00–M05. If the data character is the least-significant data character and the sign in the zone bits of the data character is minus, or if the data character is not the first, the format character (—octal 52) is used and no transfer from the E-Register to the M-Register is necessary. Working clock signal QSHM again issues, during the suppress mode, to circular shift the contents of the M-Register right one character position.

Signal DRE5 issues during time period TL5, when the edit mode is being performed, if the format character is of the sign class and if the least-significant data character is being processed to reset flip-flops E04 and E05 of the E-Register. Signal FSRI inhibits working clock signal QTLP at this time, preventing the advance of the TL-Counter. Flip-flop Q00 of the Q-Register is set to the 1-state to advance the count in the Q-Register to three, if the Program Processor is in the suppress mode, or to one, if the Program Processor is in the edit mode. QSHM issues to circular shift the contents of the M-Register right one character position. Working clock signal QSHE also issues during the edit mode to shift the contents of the E-Register right one bit position, if the format character is of the use or ignore class. If the format character is of the sign or insert class, the contents of the E-Register are not shifted and the data character in flip-flops E00–E05 is considered with the next format character.

Working clock signal QTCK continues to issue with the TL-Counter at TL5. For each clock signal QTCK, working clock signal QSHM and, under the conditions described above, QSHE issue to continue the shift operation in the M- and E-Registers. When the count in the Q-Register reaches five, working clock signal QNCD reduces the character count in the N-Register by one. The count in the TB-Counter has advanced by one, in response to signal DBCU, for each shift of the data word in the E-Register through one character position during the edit mode. During the suppress mode, the count in the CC-Register is advanced by one for each shift through one character position, if candidates for suppression are not being considered.

Flip-flop RNZ is set to the 1-state after the least-significant data character has been processed. During subsequent format characters of the sign class, signal $\overline{\text{FRNZ}}$ causes the format character (—octal 52) to be inserted in the result and the data character to be retained for consideration by the next format character. Flip-flop SRI is reset to the 0-state when the count in the Q-Register is equal to five. The delay in flip-flop SRI permits one more shift, reducing the count in the Q-Register to zero. The second half-sequence of block W14 is terminated and the Program Processor goes to block W03S, if the last character of the data word has been processed, or goes to block W15S, if the last format character in the M-Register has been processed. If neither of the two above conditions exists, the Program Processor returns to block W14S.

Referring to block W03S in FIG. 163, signal DABX issues during time period TL2 to transfer the address of the appropriate accumulator word to the B-Register of Memory. Flip-flop MWR is set to the 1-state at this time to initiate a write operation which transfers the result in the M-Register to the addressed accumulator word location in Memory. Signal DDCD issues during time period TL5 to reduce the count in the D-Register by one, forming the address of the next higher-order data word. Block W03S terminates and the Program Processor goes to block W02.

With reference to block W15S in FIG. 163, signal DABX issues during time period TL2 to transfer the appropriate accumulator word address to the B-Register of Memory. Flip-flop MWR is set to the 1-state at this time to initiate a write operation which transfers the result from the M-Register to the addressed accumulator word location. During time period TL3, flip-flop COR is set to the 1-state, during the edit mode, if the candidate-for-suppression count in the CC-Register is greater than zero when the format character count in the N-Register has been reduced to zero, indicating that the suppress mode

must be performed by the Program Processor. Flip-flop COR is also reset during time period TL3 of block W15S after the suppress mode has been completed. Flip-flop PSX is set to the 1-state during time period TL5 if the suppress mode has been completed or if the Program Processor is not to enter the suppress mode. Signal FPSX causes the address of the next instruction in the P-sequence to be applied to the Adder. Block W15S is terminated and the Program Processor goes to block W14S if the Program Processor is to enter the suppress mode or if the edit mode has not yet been completed. If the edit mode has been completed and the Program Processor is not to enter the suppress mode or if the suppress mode has been completed, the Program Processor goes to the W00 block, preparatory to execution of the next instruction. Signal DACU issues if the format character count in the N-Register is 0, 8 or 12, to form the address of the next higher-order accumulator word location in the A-Register and the Accumulator Counter Register.

FIGURE 164 is a flow diagram of Instruction 05, illustrating the sequence of operations performed in executing the instruction. Referring to FIG. 164, the least-significant data word is read from the memory location specified by the instruction address field and transferred to the E-Register. The least-significant format word is transferred from the accumulator word A location into the M-Register. A format character count equal to four times the number of words in the working Accumulator is established in the N-Register. This format character count is employed to ascertain when the edit mode and the suppress mode, if required, is ended. The Q- and CC-Registers are cleared and the class of the format character in flip-flops M00–M05 of the M-Register is checked. The operations performed while processing a format character of each class and its corresponding data character will be separately described.

Assuming that the format character in flip-flops M00–M05 of the M-Register is one of the 58 format characters of the insert class, flip-flop CL1 is set to the 1-state. If either flip-flop SP1 or SP2 is set to the 1-state, indicating that a previous format character employed during the edit mode was a suppression format character, the count in the CC-Register, which identifies the number of candidates for suppression, is advanced by one. The contents of the M-Register are circular shifted right through one character position, with the format character being retained in the result. The format character count in the N-Register is reduced by one. If the last character of the format word in the M-Register has been used, as indicated by the count in the N-Register, being 0, modulo 4, the Program Processor goes to sequence 6 in the flow diagram. Otherwise, the Program Processor re-enters the initial sequence at 5.

Assuming that the format character in flip-flops M00–M05 of the M-Register is of the use class, the Program Processor goes to sequence 2. Flip-flops CL2 and CL1 are both reset to the 0-state to identify the use class. The type of use class format character is next checked. If the format character is a "use" format character, the states of suppress mode flip-flops SP2 and SP1 are not affected. If the format character is of the "simple suppress" type, flip-flop SP1 is set to the 1-state. If the format character is of the "suppress and float $" type, both flip-flops SP2 and SP1 are set to the 1-state, whereas if the format character is of the "suppress and asterisk protect" type, flip-flop SP2 is set to the 1-state. If either SP1 or SP2 is set to the 1-state, indicating that the format character is a suppression format character, the Program Processor will enter the suppress mode upon completion of the edit mode, assuming that the candidate-for-suppression count in the CC-Register is greater than zero. The states of flip-flops SP1 and SP2 determine the type of correction to be performed during the suppress mode.

The format character in flip-flops M00–M05 being of the use class, the data character in flip-flops E00–E05 of the E-Register is transferred through the Adder to flip-flops M00–M05 of the M-Register. This transfer clears the format character from flip-flops M00–M05. If either flip-flop SP1 or SP2 is set to the 1-state, the data character in flip-flops E00–E05 is checked; if this character is a 0, the count in the CC-Register is advanced by one whereas if the character is other than a zero, the count in the CC-Register is cleared to zero, indicating that only subsequent result characters will be considered candidates for suppression. The contents of the M-Register are circular right shifted through one character position while the data word in the E-Register is open shifted right one character position. The format character count in the N-Register is reduced by one while the data character count in the TB-Counter is advanced by one. If reset to the 0-state, flip-flop RNZ is set to the 1-state at this time, indicating that the least-significant data character has been considered. If the count in the N-Register is 0, 4, 8 or 12, indicating that the last character of the format word in the M-Register has been processed, the Program Processor goes to sequence 6. If the last character of the data word in the E-Register has been processed, and if the count in the N-Register is greater than zero, the Program Processor goes to sequence 7 to read another word from Memory. If the last characters of both the data word in the E-Register and the format word in the M-Register have been processed, and if the count in the N-Register is greater than zero, the Program Processor goes to sequence 8. If neither the last character of the data word nor the last character of the format word has been processed, the Program Processor re-enters the initial sequence at 5.

Assuming that the format character in flip-flops M00–M05 of the M-Register is of the sign class, flip-flops CL2 and CL1 are both set to the 1-state. If either flip-flop SP1 or SP2 is set to the 1-state, indicating that an earlier format character was a suppression format character, the count in the CC-Register is advanced by one. If the least-significant data character is being processed and if the sign of the zone bits in the data character is plus, a space character is transferred to flip-flops M00–M05 of the M-Register. Otherwise, the format character is retained in the result. If the data character is the least-significant data character of the memory field, flip-flops E04–E05 of the E-Register are reset to the 0-state. The contents of the M-Register are circular shifted right through one character position and the format character count in the N-Register is reduced by one. The data character is retained in flip-flops E00–E05 of the E-Register for consideration with the next format character. If the last character of the format word in the M-Register has been processed, the Program Processor goes to sequence 6. Otherwise, the Program Processor re-enters the initial sequence at 5.

Assuming that the format character in flip-flops M00–M05 of the M-Register is of the ignore class, flip-flop CL2 is set to the 1-state and CL1 is reset to the 0-state. If either flip-flop SP1 or SP2 is set to the 1-state, indicating that a previous format character was a suppression format character, the count in the CC-Register is advanced by one. The contents of the M-Register are circular shifted right one character position and the contents of the E-Register are open shifted right one character position. The format character is thus retained in the result and the data character is discarded. The format character count in the N-Register is reduced by one while the data character count in the TB-Counter is advanced by one. If the last character of the format word has been considered, the Program Processor goes to sequence 6 whereas if the last character of the data word in the E-Register has been processed and if the format character count in the N-Register is greater than zero, the Program Processor goes to sequence 7. If the last characters of both the data and format words have been processed and the count in the N-Register is greater than zero, the Program Proces-

sor goes to sequence 8. Otherwise, the Program Processor re-enters the initial sequence at 5.

During sequence 6, the Program Processor stores the result word in the M-Register into the addressed accumulator word location from which the corresponding format word was read. If the format character count in the N-Register is greater than zero, the next format word from the next higher-order accumulator word location is read from Memory into the M-Register and instruction execution is continued by re-entering the initial sequence at 5. If the format character count in the N-Register is reduced to zero and if flip-flop COR is set to the 1-state indicating that a suppression format character was employed during the edit mode and the count in the CC-Register is greater than zero, the result word from the accumulator word A location is transferred to the M-Register and the Program Processor continues instruction execution in sequence 9. If flip-flop COR is reset to the 0-state, indicating that the suppress mode is not to be entered, the Program Processor goes to block W00, preparatory to execution of the next instruction in the P-sequence.

During sequence 7, the contents of the M-Register are stored in the accumulator word location from which the corresponding format word was read. The next higher-order data word is then transferred from Memory to the E-Register and the contents of the M-Register are restored to continue instruction execution with the remaining format characters. The Program Processor then re-enters the initial sequence at 5.

During sequence 8, the result in the M-Register is stored in the accumulator word location from which the corresponding format word was read. The next higher-order data word is transferred to the E-Register and the next format word is transferred from the next higher-order accumulator word location to the M-Register. The Program Processor re-enters the initial sequence at 5 to continue instruction execution in the edit mode.

During the suppress mode of sequence 9, the format character count in the N-Register is restored. The count in the CC-Register is checked. If the count in the CC-Register is not equal to the number of characters in the working Accumulator, indicating that the result character in flip-flops M00–M05 of the M-Register is not a candidate for suppression, the contents of the M-Register are circular shifted right one character position and the count in the CC-Register is advanced by one. The character count in the N-Register is also reduced by one. If the last character of the result word in the M-Register has been considered, the Program Processor goes to sequence 6 to obtain the next higher-order result word. Otherwise, the Program Processor re-enters the suppress mode sequence at 9. When the result character in flip-flops M00–M05 of the M-Register is a candidate for suppression, the count in the CC-Register will be equal to the number of characters in the working accumulator. In this event, the result character in flip-flops M00–M05 is checked to determine if it is a zero, comma or period. If so, the result character is replaced. If the states of flip-flops SP1 and SP2 identify the suppress mode as a "simple suppress," the result character is replaced by a space; if the suppress mode is "suppress and asterisk protect," the result character is replaced by an asterisk; if the suppress mode is "suppress and float $," the result character is replaced by a dollar sign. If the suppress mode is "suppress and float $," flip-flop SP2 is reset to the 0-state to change the suppress mode to "simple suppress." If the result character in flip-flops M00–M05 is not a zero, comma or period, the result character is retained. The result word in the M-Register is then circular shifted right one character position and the character count in the N-Register is reduced by one. If the last character of the result word has been processed, the Program Processor goes to sequence 6. Otherwise, the Program Processor re-enters the suppress sequence at 9.

The following examples illustrate the use of the Edit Instruction 05. In the examples, the format codes are represented in octal notation and a space is shown as "Δ." All other characters are shown as alphanumeric characters. The example illustrates single and double length operations; the rules illustrated are equally applicable to triple and quadruple length operations.

(1) *Use*, octal 56 (;):

Memory	4 3 2 1	Data
Accumulator	; ; ; ;	Format
Accumulator	4 3 2 1	Result

(2) *Insert*, all characters, except special format codes:

Memory	0 0 1 2 5 3 6 7	Data
Accumulator	; ; ; ; ; ; ; ;	Format
Accumulator	1 , 2 5 3 , 6 7	Result

(3) *Ignore*, octal 17 ( ):

Memory	7 3 2 4	Data
Accumulator	; ; ; 7	Format
Accumulator	7 3 2 7	Result

(4) *Sign*, octal 52 (—):

Memory	+ 0 3 2 1	Data
Accumulator	; ; ; -	Format
Accumulator	3 2 1 A	Result
Memory	8 5 3 1	Data
Accumulator	; ; ; -	Format
Accumulator	5 3 1 -	Result
Memory	5 8 3 1	Data
Accumulator	; - ; -	Format
Accumulator	3 - 1 -	Result

(5) *Suppression.*—Review of general rules:

Edit Mode

(1) Any of the three suppression codes (~ simple suppress, # suppress and float $, > suppress and asterisk protect) causes the CC-Register to function; as follows:
- (a) add one for each zero data character moved to the result.
- (b) add one for each format character moved to the result.
- (c) reset to zero when non-zero data character moves to result. Counting continues.

Suppress Mode

(1) The count identifies the number of leftmost positions that are candidates for suppression.
(2) Zero, comma or period candidates are replaced by some other character.
(3) The leftmost suppression code determines the type of suppression. The rightmost suppression code determines where suppression begins.

Memory	0 0 0 0	Data
Accumulator	; . ~ ;	Format
Accumulator	0 . 0 0	Result, Edit Mode
Counter	3 2 1 0	
Accumulator	Δ Δ Δ 0	Result, Final

Memory	0 0 1 0 0 2 0 0	Data
Accumulator	; ; ; . ; ~ ;	Format
Accumulator	1 0 . 0 . 2 0 0	Result, Edit Mode
Counter	0 4 3 2 1 0 1 0	
Accumulator	Same as Edit Mode Result. There are no suppression candidates.	

(6) *Simple Suppress*, octal 12 (~):

All zero, comma and period candidates are replaced by spaces.

Memory	0 0 0 1 2 3 4 5	Data
Accumulator	; ; ; ; ; ; ; ~	Format
Accumulator	Δ Δ Δ 1 2 3 4 5	Result

(7) *Suppress and Float $*, octal 13 (#):

A dollar sign replaces the rightmost zero, comma or period candidate. All other zero, comma and period candidates are replaced by spaces.

Memory	0 0 0 0 0 1 9 8	Data
Accumulator	; . ; ; # . ; ;	Format
Accumulator	Δ Δ Δ $ 1 . 9 8	Result

(8) *Suppress and Asterisk Protect*, octal 16 (>):

All zero, comma and period candidates are replaced by asterisks.

Memory	0 0 0 0 0 0 9 8	Data
Accumulator	; ; ; ; > . ; ;	Format
Accumulator	* * * * * . 9 8	Result

Instruction 67: Central Processor operations (CPO)

This instruction causes the states of predetermined indicator flip-flops, representing the status of the Central Processor, to be stored in Memory or causes these indicator flip-flops to assume predetermined states in accordance with a status word read from a specified memory location. The address of the specified memory location (second address), into which states of the indicator flip-flop is stored or from which the status word is read, is defined by employing SAS auxiliary words or is specified to be the address of a Fixed Location Index Word. The organization of the status word is illustrated in FIG. 2*m*. The relationship between the indicator flip-flops and the corresponding bit positions of the status word are as follows:

Indicator flip-flops	Bit positions
Comparison (GRE, LES)	1, 0
00-Equal	
01-Less	
10-Greater	
Program Interrupt (PIT)	2
First Time (MPL)	3
Overflow (FVO)	4
Overflow Mode (TVO)	5
Request Control (MPI)	19
Instruction Alert (PEF)	20
Control Word Alert (ICW)	21

Bit positions 6–18, 22 and 23 of the status word are not used during execution of the instruction.

The address field of the instruction word does not contain an address. However, the instruction address field may be developed, as described in the section "W01 Block of Micro-operations—Address Development." Bits 2–14 of the instruction field are ignored; bits 0 and 1 determine the type of operation to be performed by the Central Processor to execute the instruction. The relationship between the binary digits in bit positions 0 and 1 of the instruction word and the type of operation to be performed are as follows:

Instruction word bit position 1, 0	Type of operation
00	Request status of processor.
01	Set status by ORing.
10	Set status by ANDing.
11	Set status by loading.

If the contents of bits 0 and 1 of the instruction word require the "request status of processor" operation to be performed, the states of predetermined Central Processor indicator flip-flops are stored in predetermined bit positions of the specified memory location; a binary 0 is stored when the corresponding indicator flip-flop is reset to the 0-state and a binary 1 is stored when the corresponding indicator flip-flop is set to the 1-state. A binary 0 is always stored in the Program Interrupt bit position 2 and in bit positions 6–18, 22 and 23. The Request Control, Instruction Alert, Control Word Alert and Overflow flip-flops are reset to the 0-state during this type of operation and binary 0's are stored in the corresponding bit positions of the specified memory location.

During the "set status by ORing" operation, a binary 1 in any of bit positions 0–5 of the status word causes the corresponding indicator flip-flops to be set to the 1-state; a binary 0 stored in any of bit positions 0–5 of the status word causes the states of the corresponding indicator flip-flops to be unchanged. The binary digits in bit positions 6–23 of the status word are ignored. When Overflow flip-flop FVO is reset to the 0-state and the instruction is coded to set both the Overflow and the Overflow Mode flip-flops to the 1-state or to set the Overflow flip-flop to the 1-state when the Overflow Mode flip-flop is already set to the 1-state, Overflow flip-flop FVO remains reset to the 0-state.

During the "set status by ANDing" operation, a binary 0 in any of bit positions 0–5 of the status word causes the corresponding indicator flip-flops to be reset to the 0-state; a binary 1 in any of bit positions 0–5 of the status word causes the states of the corresponding indicator flip-flops to be unchanged. Bit positions 6–23 of the status word are ignored.

During the "set status by loading" operation, the contents of bit positions 0–5 of the status word are employed to determine the states of the corresponding indicator flip-flops. A binary 1 in any of bit positions 0–5 of the status word causes the corresponding indicator flip-flops to be set to the 1-state; a binary 0 in any of bit positions 0–5 of the status word causes the corresponding indicator flip-flops to be reset to the 0-state. Bits 6–23 are ignored. When the Overflow flip-flop FVO is reset to the 0-state, an instruction coded to set both the Overflow and the Overflow Mode flip-flops to the 1-state will cause only the Overflow Mode flip-flop to be set to the 1-state.

Block W00 of the initial routine, illustrated in FIG. 119, is unaltered. In executing the instruction, the Program Processor performs the micro-operations of blocks W01 and W03S in that order. The W01 block of the initial routine, illustrated in FIG. 121, is altered to the extent that the states of indicator flip-flops TVO, FVO, MPL, PIT, GRE and LES are determined if the instruction word requires "set status by ORing," "set status by ANDing," or "set status by loading" operations. During the W03S block, the states of the indicator flip-flops are stored in the specified memory location, if the instruction word requires the "request status of processor" operation.

The sequence of micro-operations of block W01, as illustrated in FIG. 121, is altered during execution of Instruction 67 as indicated in FIG. 165. With reference to FIG. 165, the setting of flip-flop MWR to the 1-state during time period TL2 is inhibited by signal DCPO to prevent restoration to memory of the word read from the second memory location. The setting of flip-flop ESX to the 1-state is also inhibited to prevent the application of the instruction word in the E-Register to the Adder. During time period TL3, signals DCM0–DCM3 issue to clear the word read from the second memory location from the M-Register, if the instruction word calls for a "request status of processor" operation.

During time period TL5 of block W01, signal DSCA issues, upon completion of development of the second address, if a binary 1 is stored in bit position 0 of the instruction word. Signal DSCZ also issues at this time if a binary 1 is stored in bit position 1 of the instruction word. If only signal DSCA issues, the operation to be performed is "set status by ORing" whereas if only signal DSCZ issues, the operation required for instruction execution is "set status by ANDing." If both signals DSCA and DSCZ issue, the "set status by loading" operation is required for instruction execution. If signal DSCA has issued, indicator flip-flops TVO, MPL, PIT, GRE and LES are set to the 1-state if bit positions 5, 3, 2, 1 and 0 respectively of the instruction word contain binary 1's. If bit position 4 of the instruction word contains a binary 1, flip-flop FVO is set to the 1-state only if Overflow Mode flip-flop TVO is reset to the 0-state.

If signal DSCZ has issued, flip-flops TVO, FVO, MPL, PIT, GRE and LES are reset to the 0-state, if bit positions 5, 4, 3, 2, 1 and 0 respectively of the instruction word contain binary 0's. Signals DSCA and/or DSCZ cause one of the operations "set status by ORing," "set status by ANDing," or "set status by loading" to be performed.

During time period TL2 of block W03S, signal DDBX issues to transfer the address of the second memory location from the D-Register to Memory, if the address control field in the instruction word has a value of zero or seven and a program interrupt subroutine is not being executed. If the address control field of the instruction word has a value in the range of 1–6, the address of the second memory location is transferred to Memory from the Q-Register. If bit positions 0 and 1 of the instruction word contain binary 0's, indicating that the "request status of processor" operation is to be performed, signal DSSC issues. Signal DGMM also issues during time period TL2. Signals DSSC and DGMM cause flip-flops M05, M04, M03, M01 and M00 to be set to the 1-state if the corresponding indicator flip-flops are set to the 1-state. The states of the indicator flip-flops TVO, FVO, MPL, GRE and LES are thus transferred to the appropriate bit positions of the M-Register. Flip-flop MWR is then set to the 1-state to initiate a write operation in Memory which stores the word in the M-Register containing the status of the Central Processor indicator flip-flops in the specified memory location.

During time period TL4, indicator flip-flops FVO and PEF are reset to the 0-state during the "request status of processor" operation. Overflow Mode flip-flop TVO is also reset to the 0-state at this time if flip-flop PIT is set to the 1-state. Control Word Alert flip-flop ICW and Request Control flip-flop MPI are reset to the 0-state at this time, if flip-flops PIT and PAD are set to the 1-state and if the operation is "request status of processor." During time period TL5 flip-flop PSX is set to the 1-state, applying the address of the next instruction in the P-sequence to the Adder. Block W03S is terminated and the Program Processor goes to block W00, preparatory to execution of the next instruction.

Instruction 00: Halt (HLT)

This instruction causes Manual flip-flop MAN to be set to the 1-state and the Central Processor to cease executing instructions. The Halt instruction establishes control conditions which inhibit the honoring of requests for program interrupts, but which allow data transfer sequences to occur if any peripheral subsystems have not terminated operations. Actuation of the Run Switch of the control Console causes execution of instructions to be resumed, starting with the next instruction in the P-sequence.

In executing the Halt instruction, the W00 block of the initial routine, illustrated in FIG. 119, is altered as indicated in FIG. 166, if modification of the instruction address field is not required. In this event, flip-flop MAN is set to the 1-state during time period TL4. If address development is required, the W00 block is unaltered.

The Program Processor then performs block W01. During time period TL4 of the W01 block, after address development is completed, flip-flop MAN is set to the 1-state. The Program Processor then goes to block W10S. During block W10S, certain control flip-flops are reset to the 0-state and flip-flop PSX is set to the 1-state to apply the address of the next instruction to the adder inputs. The Program Processor continues cycling in block W10S until Manual flip-flop MAN is reset to the 0-state. At this time, the Program Processor returns to block W00, preparatory to execution of the next instruction.

Instruction 07: General (GEN)

All input/output operations are initiated by means of Instruction 07 (GEN). The address field of the instruction word does not contain an address in Memory but rather specifies channel, device and command information for executing the instruction, as indicated in the following table.

Instruction word bit positions	Meaning
0–5	Specifies operation or function to be performed.
6–10	Specifies device (meaningful only for multi-device peripheral subsystems).
11–14	Specifies channel (and therefore peripheral subsystem).

The second address of the instruction specifies the memory location into which the status of the peripheral subsystem or channel is to be stored. The second address of the instruction may be specified as the address of a Fixed Location Index Word or may be developed by employing SAS auxiliary words.

In addition to initiating a peripheral input/output operation during which resettable status is reset and the remaining peripheral status is stored in a specified memory location, Instruction 07 (GEN) may be employed to initiate "request status" or "reset status" operations. The "request status" operation does not initiate any operation in the peripheral subsystem, but merely causes the status of the peripheral subsystem to be transmitted to Memory. Similarly, the "reset status" operation does not initiate an input/output operation in the peripheral subsystem, but resets certain status conditions before transmitting the status of the peripheral subsystem to Memory. The "request status" operation is initiated by code 00 (octal) in bit positions 0–5 of the instruction word while the "reset status" operation is initiated by code 40 (octal). The remaining codes in bit positions 0–5 of the instruction word are available as command codes to control the operation of the peripheral subsystem. A specific command code will have different effects, depending upon the peripheral subsystem which receives the command, as specified in bit positions 11–14 of the instruction word. Codes 00 and 40 are valid for all peripheral subsystems.

Execution of Instruction 07 (GEN) does not insure that an input/output operation has been initiated unless the peripheral subsystem in the channel is in a condition which permits the operation to be initiated. The condition of the peripheral subsystem or the channel comprises the status. As discussed above, one word of status information is stored into Memory, at a location determined by the second address of the instruction, every time the instruction is executed. By subsequent examination of the status information, it can be determined whether or not the desired operation was initiated, and if not, why not.

During execution of Instruction 07 (GEN), one of three conditions will exist:

(a) If the channel and peripheral subsystem are already busy, the operation required by the instruction cannot be initiated. However, peripheral subsystem status is stored into the second memory location indicating that the channel and the peripheral subsystem are busy.

(b) If the channel and the peripheral subsystem are not busy, but the peripheral subsystem is not able to initiate the operation required by the instruction, status indicating the general and specific reasons for the inability of the peripheral subsystem to initiate the required operation is stored in the second memory location.

(c) If the channel and peripheral subsystem are not busy and the peripheral subsystem can initiate the operation required by the instruction, status indicating that the channel and peripheral subsystem are ready is stored into the second memory location. If the instruction requires data transfer between Memory and the peripheral subsystem, the first word in the data control list, as specified by the address field of the channel List Pointer Word, is transferred into the channel DCW location.

The status word stored in the second memory location is organized as indicated in the following table:

Bit Positions	Meaning
0–3	Major status field which indicates eleven general status conditions applicable to all peripheral subsystems.
4	Special interrupt indicator which provides additional information for program control of some peripheral subsystems.
5–17	Ignored.
18–23	Substatus field which provides specific information about the indicated major status condition for the specific peripheral subsystem.

The major status conditions are defined as follows:

Bit Positions 3–0 of Status Word	Major Status Condition
0000	*Channel and Peripheral Subsystem Ready.*—The command has been accepted by the peripheral subsystem and the operation has been initiated. If the instruction requires a "request status" operation or a "reset status" operation, then the peripheral subsystem is ready to receive a subsequent command. No abnormal conditions are present in the channel or the peripheral subsystem.
0001	*Device Busy.*—The device is unable to accept a command because it is in the process of executing a previous command.
0010	*Attention.*—The peripheral subsystem is unable to accept a command because of an inoperable condition requiring operator intervention.
0011	*Data Alert.*—A data alert condition, such as a parity error, occurred during the last command executed by the peripheral subsystem.
0100	*End of File.*—An end of file condition was detected during the last command executed by the peripheral subsystem.
0101	*Command Rejected.*—An illegal command has been given to the peripheral subsystem.
0110	*Peripheral Subsystem Committed But Not Busy.*—The peripheral subsystem is not busy at this instant, but may not unqualifiedly accept a command because a prior command has already initiated procedures which will make the peripheral subsystem busy.
0111	*Load Operation Complete.*—The load console switch was depressed and the record of data was successfully transmitted to the channel associated with the peripheral subsystem.
1000	*Channel and Peripheral Subsystem Busy.*—The channel and peripheral subsystem are busy executing a previous command, which prevents the channel from transmitting another command at this time.

Bit Positions 3–0 of Status Word	Major Status Condition
1001	*Peripheral Subsystem Absent or Off-Line.*— The channel is unable to communicate with a peripheral subsystem because: (a) the peripheral subsystem is disconnected, (b) the peripheral subsystem is without power, (c) the peripheral subsystem is off-line, or (d) the channel is absent.
1010	*Channel Error.*—In executing the last command given to the peripheral subsystem, an error occurred in transmitting data to the channel from the peripheral subsystem.

Substatus conditions are defined specifically for each peripheral subsystem.

The W00 block of the initial routine, illustrated in FIG. 119, is unaltered during execution of Instruction 07 (GEN). The W01 block of the initial routine illustrated in FIG. 121 is unaltered, if address development of the instruction field is required.

The W01 block of the initial routine is altered, as indicated in the timing diagram of FIG. 167, during development of the second address of the instruction. Referring to FIG. 167, flip-flop BSY is set to the 1-state during time period TL1, the instruction cannot be executed for the selected channel. During time period TL2, signal DGEN inhibits the setting of flip-flop MWR to the 1-state to prevent the contents of the second memory location, which have been read into the M-Register, from being restored to Memory. The application of the instruction word in the E-Register to the Adder is also inhibited by signal DGEN.

During time period TL3, the word read from the second memory location is cleared from the M-Register. If flip-flop BSY is reset to the 0-state, indicating that an input/output operation can be initiated, signal DIOD issues and is transmitted to the channel to prepare the channel for the input/output operation. Flip-flop DFE is set to the 1-state in response to signal DIOD, transmitting device and function information from the E-Register through the input/output matrix to the selected channel. During time period TL0 of the W01 block, the W-Selector Register and the TL-Counter are preset to define the W03S block, if an input/output operation cannot be initiated, or are preset to define the W11 block if an input/output operation can be initiated.

Upon completion of the initial routine, assuming that flip-flop BSY is reset to the 0-state, the W-Selector Register and the TL-Counter are set to define blocks W11R, W12R, W12S and W03S, in that order, and the Central Processor performs the micro-operations of these blocks to initiate execution of the instruction. If flip-flop BSY is set to the 1-state, the Central Processor performs block W03S, before continuing execution of the next instruction. During block W11R, the device code of the instruction word is transmitted to the channel upon request from the channel. During block W12R, the function code of the instruction word is transmitted to the channel when the channel requests it. During block W03S, peripheral subsystem status is stored in the second memory location. After performance of the micro-operations of block W03S, if data transfer between Memory and the peripheral subsystem is required, the Central Processor leaves the W-sequence and enters the R-sequence, during which the first Data Control Word of the data control list is stored in the channel DCW location and the List Pointer Word in the channel LPW location is updated.

FIGURE 167 is a timing diagram illustrating the micro-operations performed during execution of Instruction 07 (GEN). Referring to FIG. 167, during time period TL0 of block W11R, flip-flop WRS is set to the 1-state when the peripheral subsystem acknowledges receipt of signal DIOD transmitted during block W01 and requests that the device code be transmitted to the peripheral subsystem. During time period TL1, signals DIDS and DWDO issue to effect the transfer of the device code from the channel to the peripheral subsystem. Signal DWST issues during time period TL2 to notify the peripheral subsystem that the device code was transmitted. If the peripheral subsystem does not request the device code prior to initiation of block W11R, block W11R is repeated until the device code is requested by the peripheral subsystem and transmitted to the peripheral subsystem.

The micro-operations of block W12R are performed after block W11R. During block W12R, signal DSH3 issues to transfer the function code of the instruction word from the E-Register through the input/output matrix to the channel. Flip-flop WRS is set to the 1-state when the peripheral subsystem requests the function code. In response to signal FWRS, signal DIDS issues during time period TL1 to effect the transfer of the function code to the peripheral subsystem. Signal DWST issues to notify the peripheral subsystem that the function code has been transmitted. If the peripheral subsystem does not request the function code during the first performance of block W12R, block W12R is again performed. After the function code has been transmitted to the peripheral subsystem, block W12S is performed.

During block W12S, DSH0, DSH3, DME0 and DME3 issue to apply the major status and substatus of the peripheral subsystem from the channel to the input/output matrix. Flip-flop WRS is set to the 1-state when the peripheral subsystem requests a release pulse. If the peripheral subsystem does not request a release pulse during the first performance of block W12S, block W12S is again performed. The Program Processor then goes to block W03S.

During time period TL2 of block W03S, signal DDBX or signal DQBX issues to transfer the address of the second memory location through the B-Gates to the B-Register of Memory. Signal DPIB may issue to address the channel PSW location if Instruction 07 (GEN) is being performed during a program interrupt and if the second memory location is the PSW location. Signal DGMM issues to transfer the major status and substatus of the peripheral subsystem to bit positions 0–4 and 18–23 respectively of the M-Register. Flip-flop MWR is set to the 1-state to initiate a write operation which stores the status word of the peripheral subsystem in the addressed memory location. Flip-flop IOR is set to the 1-state if signals $\overline{\text{FBSY}}$·DBYC exist, indicating that the channel was not busy but has become busy and will transfer data between Memory and the associated peripheral subsystem. During time period TL5 of block W03S, release signal DREL issues and is transmitted to the peripheral subsystem, if the channel did not go busy.

After block W03S, if data transfer between the peripheral subsystem and Memory is required, the output of the W-Selector Decode Logic is inhibited and blocks R01–R05 of the R-sequence are performed to transfer the first Data Control Word in the data control list, as specified by the address field of the List Pointer Word in the channel LPW memory location, into the DCW location for the channel. The List Pointer Word is updated prior to restoration to the LPW memory location of the channel. The release pulse DREL is sent to the peripheral

subsystem during time period TL4 of block R05 during the R-sequence.

FIGURE 168 is a flow diagram illustrating the sequence of operations performed during execution of Instruction 07 (GEN).

ALERTS

Three types of alert states can occur in the Central Processor, viz. instruction alert, control word alert and memory alert. An instruction alert occurs, and the corresponding indicator flip-flop PEF is set to the 1-state, upon detection of an invalid operation code or upon the detection of an invalid memory address, except during a data interrupt. A control word alert occurs, and the corresponding indicator flip-flop ICW is set to the 1-state, upon detection of an instruction other than Instruction 07 (GEN) or Instruction 17 (SPB) in the PIW location assigned to the processor channel or to an input/output channel, upon detection of an invalid address during a data interrupt, or upon detection of an invalid address in a LPW location.

A memory alert occurs, and the corresponding indicator flip-flop WJE is set to the 1-state, upon detection of a parity error in a word read from Memory. A memory alert causes the operation of the data processing system to halt. If the memory alert is caused by detection of a parity error in an instruction word, instruction execution by the Central Processor is halted immediately. If a word other than an instruction word causes a memory alert, the Central Processor halts after execution of the current instruction is complete. In either of the above cases, data interrupts continue until all currently initiated input/output operations are complete.

FIGURE 169 illustrates in summary form the conditions causing alerts and the resulting Central Processor actions. As indicated in FIG. 169, the Central Processor may be halted either immediately or at the end of the instruction being executed. If an invalid address is detected in a LPW location during execution of Instruction 07 (GEN) or in either a LPW or a DCW location during a data interrupt, and Program Interrupt flip-flop PIT is set to the 1-state, or if a parity error is detected in other than an instruction word, the Central Processor is not halted until execution of the current instruction is completed. Under all other conditions, the Central Processor is halted immediately upon detection of an invalid address or other error, if flip-flop PIT is set to the 1-state. If PIT is reset to the 0-state, a program interrupt occurs. For example, detection of an invalid operation code in a Program Interrupt Word causes the Central Processor to be halted immediately. Either type of halt is effected by setting Manual flip-flop MAN to the 1-state at the end of the W00 block. In the case of an immediate halt, the 1-output signal of flip-flop AIM causes signal DINP to issue, immediately resetting the flip-flops of the W-Selector Register to the 0-state, to define block W00.

PROCESSOR CHANNEL

The processor channel is associated with Central Processor operations and does not have input/output capability. The functions of the processor channel are:

(a) To place in Memory, under program control, the current status of specified Central Processor indicators by execution of an Instruction 67 (CPO) coded to perform a "request processor" operation.

(b) To set into the Central Processor indicators, under program control, status which has previously been placed in Memory by executing an Instruction 67 (CPO) coded to perform a "set status by loading," a "set status by ANDing" or a "set status by ORing" operation.

(c) To provide for interrupting the program and transferring control to a subroutine upon the occurrence of specific events in the Central Processor.

Reference is made to the section entitled "Instruction 67: Central Processor Operations (CPO)," for a detailed description of the request status and set status operations.

The following indicator flip-flops are employed in the Central Processor:

(a) Comparison indicator flip-flops GRE and LES,
(b) Program Interrupt indicator flip-flop PIT,
(c) First Time indicator flip-flop MPL,
(d) Overflow indicator flip-flop FVO,
(e) Overflow Mode indicator flip-flop TVO,
(f) Request Control indicator flip-flop MPI,
(g) Instruction Alert indicator flip-flop PEF, and
(h) Control Word Alert indicator flip-flop ICW.

The comparison indicator flip-flops GRE and LES show the result of the most recent compare operation during execution of one of the Instructions 01 (CMI), 02 (CDA), 03 (CAA) or 04 (CMM). The comparison indicator flip-flops are used to control conditional branching during execution of Instructions 12 (BRG), 13 (BRE) and 14 (BRL). The state of the comparison indicator flip-flops can also be determined by a set status operation during execution of Instruction 67 (CPO). Instruction 22, Shift (SHG), can also control the states of the comparison indicator flip-flops. The condition of "less-greater" (both GRE and LES set to the 1-state) can only occur as a result of a set status operation in executing Instruction 67 (CPO). The "less-greater" setting causes a branch to occur in executing either Instruction 12 (BRG) or 14 (BRL).

The Program Interrupt indicator flip-flop PIT is automatically set to the 1-state when a program interrupt request is honored by execution of an Instruction 17 (SPB) from the channel PIW location. While the Program Interrupt indicator is on, no other program interrupt request can be honored. Program interrupt requests by the input/output channels or by the processor channel will be remembered and serviced when the Program Interrupt indicator is turned off and when control has been returned to the interrupted program by means of an Instruction 10 (BRU). Because the processor channel has a higher program interrupt priority than any of the input/output channels, its program interrupt requests are honored first. Requests for a program interrupt from the input/output channels are then taken in the order of regular channel program interrupt priority. The Program Interrupt indicator can be turned on or off with an Instruction 67 (CPO) set status operation. The Program Interrupt indicator can also be turned off with either the Reset Computer or the Program Interrupt Reset console switches.

The First Time indicator flip-flop MPL is used to control operations during Instruction 26, Variable Length Divide (VLD), and Instruction 27, Variable Length Multiply (VLM). During execution of Instruction 27 (VLM), flip-flop MPL permits the signs of the multiplicand and multiplier to be recognized and the two high-order accumulator words to be cleared to zeros. During execution of Instruction 26 (VLD), the First Time indicator flip-flop preserves the sign of the dividend during the execution of the first VLD instruction and permits the VLD instruction to reset the Overflow flip-flop. When the First Time indicator flip-flop is reset to the 0-state, extraneous zone bits are prevented from appearing in quotient characters. First Time indicator flip-flop MPL is set to the 1-state by actuation of the Reset Computer console switch, by the control count during execution of Instruction 16 (BRC) going to zero, or by a set status operation during execution of Instruction 67 (CPO). Flip-flop MPL is reset to the 0-state by execution of either Instruction 26 (VLD) or 27 (VLM) or may be reset by a set status operation during execution of Instruction 67 (CPO).

The Overflow indicator flip-flop FVO is turned on by a predetermined condition arising during execution of one of the Instructions 26, 34, 35, 50–57 and 60–63. The set status operation during execution of Instruction 67 (CPO) can also be employed to turn the Overflow indicator on. The Overflow indicator flip-flop is reset to the 0-state by actuation of the Reset Computer console switch, by execution of the first Instruction 26 (VLD), by a set status operation during execution of Instruction 67 (CPO) when the status word is coded to turn off the indicator, and automatically when processor status is stored by a request status operation during execution of Instruction 67 (CPO).

The Overflow Mode indicator flip-flop TVO is used to control the effects of an overflow condition. If the Overflow Mode Indicator is off at the time of an overflow condition, the overflow condition turns on the Overflow Mode indicator and the next instruction is executed in normal sequence. If the Overflow Mode indicator is on at the time of an overflow condition, the overflow turns on the Overflow indicator and a program interrupt to the processor channel occurs. The Overflow Mode indicator can be turned on by a set status operation during execution of Instruction 67 (CPO) when the status word is properly coded. The Overflow Mode indicator flip-flop may be reset to the 0-state by actuation of the Reset Computer console switch, by a set status operation during execution of Instruction 67 (CPO) when the status word is coded to turn off the indicator, or by a request status operation during execution of Instruction 67 (CPO) when the Program Interrupt indicator is on.

Actuation of the Request Control console switch sets Request Control indicator flip-flop MPI to the 1-state, causing a program interrupt to the processor channel. The Request Control indicator flip-flop is reset to the 0-state by actuation of the Reset Computer console switch or automatically by a request status operation during execution of Instruction 67 (CPO) when the processor status is stored.

Instruction Alert indicator flip-flop PEF is set to the 1-state automatically when any one of the following three conditions exist:

(a) Invalid operation code—This condition occurs when any bit configuration is detected in bit positions 18–23 of the instruction word other than those specified in the table "INSTRUCTION DECODE."

(b) Invalid address—This condition occurs when a memory address in bit positions 0–14 of the instruction word is larger than the memory capacity. The address is tested for validity as it is used to access Memory. The Instruction Alert indicator is not affected by the detection of an invalid address during a data transfer sequence.

(c) Invalid accumulator location—This alert occurs when an accumulator location larger than the memory capacity is detected. The invalidity is not detected until an attempt is made to access the Accumulator.

If the Instruction Alert indicator is turned on while a program interrupt request is being serviced, execution of the current instruction is terminated and the Central Processor halts with the Instruction Alert indicator on. If the Instruction Alert indicator is turned on when no program interrupt is being serviced, execution of the current instruction is terminated and a program interrupt to the processor channel occurs. The Instruction Alert indicator is automatically turned off after Central Processor status is stored by a request status operation during execution of Instruction 67 (CPO). If the Instruction Alert indicator is turned on due to an invalid operation code in a Program Interrupt Word (PIW) execution of the current instruction is terminated and the Central Processor halts.

Control Word Alert indicator flip-flop ICW is set to the 1-state in response to an invalid address during a data interrupt, in response to an invalid address in the List Pointer Word (LPW) during execution of Instruction 07 (GEN), or in response to a program interrupt request when the instruction taken from the channel PIW location is not one of the Instructions 07 (GEN) or 17 (SPB). If the Control Word Alert indicator is turned on due to an invalid address during a data interrupt, the data interrupt is terminated and the End Data Transfer signal DEDX is transmitted to the channel whose LPW or DCW location contains the invalid address. Execution of the current instruction is completed. If the Central Processor is not servicing a program interrupt when the invalid address is detected, a program interrupt to the processor channel occurs and the Control Word Alert indicator is turned off after Central Processor status is stored by a request status operation during execution of Instruction 67 (CPO). If the Central Processor is servicing a program interrupt when the invalid address is detected, system operation is halted. When the Control Word Alert indicator is turned on due to the absence of an Instruction 07 (GEN) or an Instruction 17 (SPB) in a channel PIW location, execution of the instruction is terminated and system operation is halted.

The logical schematic diagrams for the processor channel are given by the flip-flop input diagrams of the indicator flip-flops in FIGS. 81–83, 85, 87, 88, 92 and 184 and by the special logic involved in executing Instruction 67, Central Processor Operations (CPO).

INPUT/OUTPUT CONTROL UNIT

The Input/Output Control Unit of the Central Processor controls all communications between the peripheral subsystems and Memory. The Input/Output Control Unit also controls communications between the peripheral subsystems and the Program Processor. Each peripheral subsystem is connected to an input/output channel of the Input/Output Control Unit. The input/output channels provide limited temporary storage for information being transferred between the associated peripheral subsystems and either the Program Processor or the Memory. The input/output channels enable the transmission of instructions from the Program Processor to the peripheral subsystems, the transfer of data between Memory and the peripheral subsystems, and the transfer of information about the operating conditions of the peripheral subsystems and the channels to the Program Processor.

An input/output operation is initiated by an Instruction 07, General (GEN), in the program. Once an input/output operation has been initiated, sequential execution of the instructions of the program continues. As it becomes necessary to transfer one character or one word of data between Memory and a channel of the Input/Output Control Unit, instruction execution is delayed for the period of time required to permit the character or word to be transferred. This interruption of program execution is termed a data interrupt. When the input/output operation is completed, sequential processing of the instructions of the program is terminated and a program interrupt occurs.

As illustrated in FIG. 170, Input/Output Control Unit **13** includes eight input/output channels designated channel #1 through channel #7 and channel #0. Each of the channels is connected to a peripheral subsystem which may be a specific single device or may include multiple devices. The channels communicate with Memory and the Program Processor through input/output matrix **700**. All data passing between input/output matrix **700** and Memory or the Program Processor on line **701** passes through M-Register **301** of the Program Processor. Input/output matrix **700** is a logic gating structure which may be controlled to gate any one of four characters at the

matrix input to any one of four character positions at the matrix output. Common logic 702 provides gating signals to control the transfer of data into and out of the channels. The operation of input/output matrix 700 is controlled by common logic 702. In addition, common logic 702 checks and generates parity for characters received and transmitted by a type of channel termed a "character" channel, as described in the section entitled, "Input/Output Channels." Common logic 702 also includes data interrupt logic, program interrupt logic and logic for executing Instruction 07 (GEN). The lines interconnecting the components of FIG. 170 represent symbolically paths of data and control signals.

A description of the operation of the Input/Output Control Unit is provided in the section entitled, "Instruction 07: General (GEN)" and in the succeeding sections. Details of the structure of the Input/Output Control Unit are provided in FIGS. 171, 172, 174 and 178–196.

Input/output channels

In input/output operations involving data transfers, the temporary storage capacity provided by a channel depends upon the type of channel employed, viz. a character channel, a single-word channel, or a double-word channel. All data transfer between a peripheral subsystem and its associated input/output channel is one character at a time. Data movement between a character channel and the input/output matrix is also one character at a time. However, data movement between a single-word channel or a double-word channel and the input/output matrix is one word at a time.

A character channel permits one character of temporary storage during read and write operations. During a read operation, the channel obtains a character from the peripheral subsystem and the character must be transmitted from the channel to Memory before the next character from the peripheral subsystem can be accepted by the channel. During a write operation, when the peripheral subsystem requests a character from the character channel, the channel must obtain the required character from Memory and transmit it to the peripheral subsystem before the subsystem can request another character.

During a read operation employing a character channel, the following sequence of events occurs:

(a) The peripheral subsystem transmits one data character and the parity bit for that data character to the character channel. The character channel stores this character while parity for the character is checked.
(b) The peripheral subsystem transmits a signal to the channel to indicate that a data character has been transmitted and the character channel employs this signal to request a data interrupt.
(c) When the data interrupt request from the channel is recognized, the data character is transferred from the channel to Memory through the input/output matrix.
(d) The character channel then notifies the peripheral subsystem that the channel is ready to receive another character.

During a write operation, when the peripheral subsystem is ready to receive a character, it requests a data character from the character channel. The character channel obtains the character from Memory, in a sequence similar to that described above for a read operation, and then transmits a signal to the peripheral subsystem indicating that the character is available.

A single-word channel provides four characters of temporary storage. During a read operation employing a single word channel, the channel receives characters from the peripheral subsystem, one at a time. The characters are stored in the single-word channel in the order received. When the single-word channel is filled, the channel issues a request for a data interrupt so that the entire word can be transferred from the channel to Memory through the input/output matrix. During a write operation, the single-word channel initially obtains a four-character word from Memory and stores it. The most significant character of the word in the single-word channel is transferred to the peripheral subsystem when the subsystem requests a character. The remaining characters are transmitted one at a time in the order of decreasing significance upon subsequent requests from the peripheral subsystem. When the last character is transmitted from the single-word channel to the peripheral subsystem, the channel issues a request for a data interrupt in order to obtain another four-character word from Memory.

A double-word channel provides two words, or eight characters, of temporary storage during data transfers. During a read operation employing a double-word channel, the channel receives characters from the peripheral subsystem one at a time and places the characters in the first word position of the double-word channel. When the first word position is filled, the channel issues a request for a data interrupt so that the data word assembled in the first word position of the double-word channel can be transferred to Memory. Due to the availability of the second-word position in the channel, additional characters can be received from the peripheral subsystem before the data interrupt occurs to transfer the four characters in the first word position of the channel to Memory.

During a write operation, the double-word channel initially obtains two four-character words from Memory, one word at a time, and places them in the first and second word positions in the channel. The most-significant character of the first word is transferred to the peripheral subsystem when the peripheral subsystem requests a character. The remaining characters of the first word and the characters of the second word are transmitted, in the order of decreasing significance, sequentially to the peripheral subsystem as the subsystem requests characters. When one of the word positions of the double-word channel is empty, the channel issues a request for a data interrupt to obtain another word from Memory. However, additional characters are available for transmission to the peripheral subsystem before a data interrupt occurs to transfer another four-character word from Memory to the double-word channel.

Data transfer

FIGURE 171 illustrates the transfer of data in the Input/Output Control Unit 13. Only a single channel 708 is shown in FIG. 171 since the data signal identification is common to all channels. During a read operation, data character signals REX0–REX5 and a parity bit are transmitted from the peripheral subsystem 709 to the associated channel 708 over line 710. During a write operation, data character signals RCX0–RCX5 are transmitted on line 711 from channel 708 to peripheral subsystem 709. Command signals to the channel and substatus signals from the channel are also transmitted over lines 710 and 711. Line 712 carries major status and program interrupt signals to the channel, control signals from the channel, and synchronizing clock signals.

Parity check and generation unit 714 is connected to channel 708 to check the parity of characters transmitted from peripheral subsystem 709 to channel 708 and to generate parity for characters transmitted from channel 708 to peripheral subsystem 709. Parity check and generation unit 714 is employed only with single-word or double-word channels. Parity check and generation for a character channel is performed by common logic 702.

Signals DJ00–DJ05, DJ10–DJ15, DJ20–DJ25 and DJ30–DJ35 are transmitted on line 715 from OR-gate 716 as input signals to input/output matrix 700. OR-gate

716 has applied to its input terminals data signals from channel 708, data signals from M-Register 301 and control signals from E-Register 303. Signals DK00–DK05, DK10–DK15, DK20–DK25 and DK30–DK35 on line 717 from channel 708 represent a character, if channel 708 is a character channel, or may represent from one to four characters, if channel 708 is either a single-word or a double-word channel. The data signals from M-Register 301 on line 718 are identified as signals DG00–DG23. Signals FE00–FE10 from E-Register 303 on line 719 represent device and function information during execution of Instruction 07, General (GEN).

The output signals of input/output matrix 700 are identified as signals DI00–DI05, DI10–DI15, DI20–DI25 and DI30–DI35. During a read operation, the matrix output signals are transmitted on line 720 to M-Register 301. During a write operation, the signals are transmitted on line 721 to channel 708.

Data Interrupt

Each input/output operation is initiated during program processing by the execution of Instruction 07, General (GEN), which designates:

(a) The channel (and thus the peripheral subsystem),
(b) The device (for a multi-device peripheral subsystem), and
(c) The operation to be performed (read document, punch card, rewind tape, etc.).

Once the input/output operation is initiated, it proceeds under the control of the input/output channel and the peripheral subsystem, and the Program Processor is free to continue executing program instructions. When an Instruction 07 (GEN) involving data transfer has been executed and the designated peripheral subsystem is ready for data transfer to begin, the associated input/output channel issues a request for a data interrupt. When the data interrupt request is granted to the requesting channel, sequential processing of program instructions stops and the Central Processor performs a data interrupt operation.

The primary purpose of a data interrupt is to effect a data transfer between the requesting input/output channel and Memory with a minimum delay in execution of program instructions. The amount of data transferred during a single data interrupt is a function of the channel type; for character channels, one character is transferred and for single- or double-word channels, four characters are normally transferred. As soon as the data transfer is completed, the data interrupt is terminated and execution of program instructions is resumed. When the channel is ready for the next data transfer, another request for a data interrupt is issued by the channel. A data interrupt is required for each memory access involved in a given input/output operation. Between program interrupts, and while data is being processed within the input/output channel and the peripheral subsystem, program execution continues.

During a data interrupt, access to Memory for data transfers is controlled automatically by two channel control words for the selected channel, viz. the List Pointer Word (LPW) and the Data Control Word (DCW). These channel control words, located in Memory at predetermined locations for each channel, are effective only during input/output operations involving data transfers. They control memory addressing and character counting during each transfer of data to or from Memory.

The organization of a Data Control Word (DCW) is illustrated in FIG. 2*o*. The address field (bits 0–14) of the Data Control Word contains the address of the first memory location to be used for data transfer between the peripheral subsystem and Memory. The count field (bits 15–23) of the Data Control Word contains a count of 512–N, expressed in binary, where N is the number of characters to be transferred between Memory and the peripheral subsystem. The two low-order bits of the count field of the Data Control Word also determine the character position within the word of the first character to be transferred between Memory and the peripheral subsystem.

The organization of a List Pointer Word (LPW) is illustrated in FIG. 2*n*. Before the initiation of an input/output operation, the address field (bits 0–14) of the List Pointer Word contains the address of the first word of a list of control words called the data control list. The count field (bits 15–23) of the List Pointer Word contains a count of 512–M expressed in binary, where M is a number of control words in the data control list.

During an input/output operation initiated by execution of an Instruction 07, General (GEN), the address field of the List Pointer Word of the appropriate channel is used to move the first Data Control Word of the data control list into the DCW location for that channel. As data characters are transferred between the input/output channel and Memory, the count field of the channel Data Control Word in the DCW location is incremented by the number of data characters transferred. The address field of the Data Control Word is also incremented by one to form the address of the memory location to which or from which the data is to be transferred. The address field of the Data Control Word is not incremented during the first data interrupt for each Data Control Word, since the address field already specifies the correct memory location.

The maximum count of the Data Control Word count field is 511 (in binary) and the count field overflows or goes to zero after the Data Control Word has caused the desired N characters to be transferred between Memory and the peripheral subsystem. The next Data Control Word in the data control list is then transferred to the DCW memory location for the channel, under control of the List Pointer Word.

As each Data Control Word of the data control list is placed in the DCW location for the channel, the count field of the List Pointer Word is incremented. The address field of the List Pointer Word is also incremented, to form the address of the next word in the data control list, except during the first use of the List Pointer Word to obtain a word from the data list. The successive words of the data control list are thus automatically accessed by the List Pointer Word. The input/output operation continues until the count field of the List Pointer Word overflows or goes to zero, indicating that the last word in the data control list has been read from Memory. This word is the LPW restore word which is transferred to the LPW location for the channel, preparatory to performance of another input/output operation in the channel.

DATA INTERRUPT LOGIC

Simultaneous requests for data interrupts from two or more of the eight input/output channels of the Input/Output Control Unit may occur. To assure orderly processing of data interrupt requests, each channel is assigned a priority. When more than one channel requests a data interrupt, the request of the channel having the highest priority is granted. All other data interrupt requests are serviced, in the order of the priority of the respective channels, before instruction processing is resumed.

FIGURE 172 illustrates the data interrupt logic which forms a part of the common logic of the Input/Output Control Unit. Referring to FIG. 172, the peripheral subsystem associated with each of the channels transmits a signal JEAX during a read input/output operation, when it is ready to transmit a character to the channel, or a signal JEWX during a write input/output operation, when it is ready to receive a character from the channel. In response to either signal JEWX or JEAX, the associated channel transmits a signal representing a request for a data interrupt. The data interrupt request signals for channels #1–#0 are DDI1–DDI0 respectively. The data interrupt request signals are applied to the input terminals of plugboard 740. The output terminals of plugboard 740 are

connected to the respective set input terminals of data interrupt priority flip-flops DSA–DSH.

Each of the data interrupt priority flip-flops DSA–DSH it set to the 1-state in response to signal FIRT and the corresponding signal ZDSA–ZDSH from the appropriate output terminal of plugboard **740**. By connection of conductors between selected input and output terminals of plugboard **740**, a given channel may be assigned any priority. Flip-flop DSA represents the highest priority while flip-flop DSH represents the lowest priority, the flip-flops between DSA and DSH representing successively decreasing priorities. As indicated in FIGURE 172, the interconnection of terminals **745** and **746** of plugboard **740** gives a data interrupt request from channel #2 priority over a data interrupt request from any other channel. The interconnection of plugboard terminals **747** and **748** gives channel #1 priority, second only to that of channel #2. Channels #3 through #7 may also be assigned an appropriate priority by suitable interconnection of the input and output terminals of plugboard **740**. Channel #0 is normally connected to the typewriter and retains lowest priority. The 1-output signal of flip-flop IRT, which is set to the 1-state in response to signal DIRS, permits the data interrupt priority flip-flops DSA–DSH to be set in response to a data interrupt request transmitted through plugboard **740** only if the Program Processor is at a point in the W-sequence when no significant data is contained in the M-Register.

The 1-output signals of data interrupt priority flip-flops DSA–DSH are applied to data interrupt priority gates **750**. One of the signals DDPA–DDPH issues from data interrupt priority gates **750** in response to the data interrupt priority flip-flop output signals applied to gates **750**, the signal which issues indicating the highest-priority flip-flop which is set to the 1-state. For example, if flip-flop DSA is set to the 1-state, signal DDPA issues from data interrupt priority gates **750**. The output signals of data interrupt priority gates **750** are applied to the input terminals of plugboard **752**. The signals ZDPA–ZDPH at the output terminals of plugboard **752** are decoded in B-Gates **317** to address the appropriate Data Control Word and List Pointer Word for the channel being granted a data interrupt. Plugboard **752** may be employed to control the List Pointer Word and the Data Control Word which are used with a particular channel. For example, with terminals **754** and **755** of plugboard **752** being connected to terminals **757** and **756** respectively, the List Pointer Word and the Data Control Word normally associated with channel #1 are actually used with channel #1 during a data interrupt to channel #1 while the List Pointer Word and the Data Control Word normally associated with channel #2 are actually employed to control data transfers during a data interrupt to channel #2.

When a request for a data interrupt is granted and one of the data interrupt priority flip-flops is set to the 1-state, data interrupt priority gates **750** generate signal DDIB, causing signal DDIE to issue. Signal DDIE is applied to R-Selector Register **758**. R-Selector Register **758** controls the operations occurring during the data interrupt by means of a series of blocks of micro-operations called the R-sequence. DDIE also issues in response to signal FIOR, initiating the R-sequence during execution of Instruction 07 (GEN) to update the channel control words.

Output signals ZDPA–ZDPH of plugboard **752** are applied to OR-gates **760**. The output signals DPC1–DPC0 of OR-gates **760** are channel select signals which issue to select a given channel during a data interrupt. Signals DPC1–DPC0 are applied to the input terminals of plugboard **765**. The output signals ZCN1–ZCN0 from the output terminals of plugboard **765** are applied to the respective channels. Plugboard **765** is employed to route the channel select output signal of OR-gates **760** to the channel having the highest data interrupt priority and being granted a data interrupt. In the plugboard wiring example illustrated in FIG. 172, terminals **770** and **771** of plugboard **765** are connected to terminals **772** and **773** respectively. Channel #1 is then selected by signal ZCN1 in response to data interrupt request signal DDI1 while channel #2 is selected by signal ZCN2 in response to data interrupt request signal DDI2.

Signals SLS1–SLS0 from the Load Channel console switch and signals DEL1–DEL0, which represent channel information from the E-Register during execution of Instruction 07 (GEN), are also applied to OR-gates **760**. The channel select output signals DPC1–DPC0 of OR-gates **760** may thus also select a given channel during initial program loading under control of the Console or during execution of Instruction 07 (GEN).

R-SELECTOR REGISTER

The R-Selector Register is a three-bit register comprising flip-flops RR0–RR2. Six of the eight possible states of the R-Selector Register flip-flops are decoded by logic to provide corresponding signals DR00–DR05. The state of the R-Selector Register defined by signal DR00 indicates that a data interrupt or input/output operation is not being performed. The remaining five of these six states of the R-Selector Register, represented by signals DR01–DR05, cause a corresponding block of micro-operations to be performed by the Central Processor. Each of the five blocks of micro-operations performs some basic macro-operation in the Central Processor, such as Data Control Word fetch, etc. The R-Selector Register defines the blocks of micro-operations in a sequence appropriate to the input/output operation being performed. The operation of the R-Selector Register is analogous to that of the W-Selector Register. The W-Selector Register defines blocks of micro-operations in a particular sequence to effect instruction execution whereas the R-Selector Register defines blocks of micro-operations in a particular sequence to control the transfer of data to or from a peripheral subsystem during a data interrupt. The logical schematic diagrams of the flip-flops of the R-Selector Register are illustrated in FIG. 178.

Input/Output Control Unit **13** can interrupt the execution of an instruction by the Program Processor for a data interrupt at predetermined times during predetermined blocks of the W-sequence. Signal DIRS defines the times when the Input/Output Control Unit **13** can interrupt the execution of an instruction by the Program Processor. Since input/output operations employ the M-Register, signal DIRS indicates the time at which the M-Register does not contain significant data.

In operation, if a request for a data interrupt occurs, the R-Selector Register remains in the state indicated by signal DR00, or the R00 state, until the signal DIRS issues, indicating that the Program Processor has reached a point in instruction execution at which the W-sequence may be interrupted. The R-Selector Register then changes to the R01 state and the W-Selector Register changes states to define the next block of micro-operations to be performed in the W-sequence. However, the outputs of the W-Selector Decode Logic and the outputs of the Adder are inhibited by signal $\overline{DR00}$. When the R-Selector Register advances to the R01 state, the TL-Counter is preset to define time period TL0 and, if the next block of the W-sequence is to be a second half-sequence, flip-flop LCP is set to the 1-state. The TL-Counter controls the timing within a block of the R-sequence just as it does within a block of the W-sequence. At the end of the R-sequence, the TL-Counter is preset to define time period TL0, if flip-flop LCP is reset to the 0-state, or is preset to define time period TL2, if flip-flop LCP is set to the 1-state, preparatory to performance of the micro-operations of the next block of the W-sequence.

The following tables indicate the signals issuing and the sequence of micro-operations occuring during each of the blocks R01–R05 of the R-sequence.

R01 BLOCK

TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS
TL0			
QTCK	$(\overline{FMNX}+FMNX \cdot FDCW)$ ΔMRD		
TL1		IRE	$(FMRD+FMNX \cdot \overline{FDCW} \cdot DCH0)$ $\Delta(DCM0, DCM1, DCM2, DCM3)$
			$FARM \cdot FMNX \Delta DAEX$
QTCK	MSX	RCK	
TL2			
QTRK	RCK		
QTCK	$\overline{FM15} \cdot \overline{FM16} \cdot \overline{DFTC} \cdot \overline{FMNX}$ ΔRCA		$DRIT \cdot FMNX(FPRM+FEMA)$ $\Delta(QSMA, QSMB, QSMC)$ $FARM \cdot FMNX\Delta(QSMA, QSMB)$
TL3	$MC23 \cdot \overline{FMNX} \cdot DDIB \Delta RVO$ $FM15 \cdot FMNX \Delta CK0$ $FM16 \cdot FMNX \Delta CK1$		$\overline{DRIT} \cdot FMNX \Delta DGMM$ $\overline{FMNX} \Delta DIDS$ $\overline{FMNX} \Delta DCTC$
QTCK			
TL4			$FMNX \Delta DRWR$ $\overline{FWRT} \cdot FMNX \Delta DWST$ $\overline{FWRT} \cdot FMNX \Delta DRRR$
QTCK	$(\overline{FMNX} \cdot \overline{FRV\emptyset}$ $+FMNX \cdot FDCW) \Delta MWR$		$\overline{FMNX} \Delta(QSMA, QSMB,$ $QSMC)$ $FPRM \cdot \overline{DRIT} \cdot FMNX \Delta QSPX$
TL5	$FMNX \cdot DCCS$ $\Delta(INCREMENT\ CK0,$ $CK1)$		
QTCK		MSX, RCK	

The following signals, if the indicated conditions are satisfied, are present during the entire R01 block:

$DRIT \cdot FMNX(FEMA+FARM) \Delta DEST$
$DCCS \cdot \overline{FMNX} \Delta DY15$
$DCCS \cdot \overline{FMNX} \cdot \overline{DCC5} \cdot \overline{DCC6} \Delta DY17$
$DCC5 \Delta DX15$
$DCC6 \Delta DX16$
$DCH0 \cdot FMNX \cdot FDCW \cdot \overline{DRIT} \Delta \overline{DAMR}$
$FPRM \cdot DRIT \cdot FMNX \Delta DPSR$
$DRIT \cdot DDIB \Delta DGDM.$

R02 BLOCK

TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS
TL0			
QTCK	MRD		
TL1			DCM0, DCM1, DCM2, DCM3
QTCK		RCK	
TL2			
QTRK	RCK		DMWS
QTCK			
TL3			$\overline{DRIT} \Delta DGMM$ DIDS
QTCK	MWR		
TL4			$\overline{FWRT} \Delta(DRWR, DWST)$ $\overline{FWRT} \Delta DRRR$
QTCK			
TL5	$\overline{FRV\emptyset} \Delta IRT$	CK0, CK1	
QTCK		RCK	

The following signals, if the indicated conditions are satisfied, are present during the entire R02 block:

DMBX
$\overline{DRIT}(DFTC+DCH0)\Delta\overline{DAMR}$
$DRIT \cdot DDIB\Delta DGDM$

R08 BLOCK

TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS
TL0			
QTCK	MRD		
TL1		RVO	DCM0, DCM1, DCM2, DCM3
QTCK	MSX	RCK	
TL2			
QTRK	RCK		
QTCK	$DDIB \cdot \overline{FMEM}$ ΔRCA		
TL3	$\overline{MC23} \cdot DDIB$ ΔRVO		
QTCK			
TL4	$\overline{FRVO}\Delta MWR$		
QTCK			QSMA, QSMB, QSMC
TL5			
QTCK		MSX, RCA, RCK	

The following signal, if the indicated condition is satisfied, is present during the entire R03 block:

$\overline{FMEM}\Delta DY15$

R04 BLOCK

TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS
TL0			
QTCK	MRD		
TL1			DCM0, DCM1, DCM2 DCM3
QTCK		RCK	
TL2			
QTRK	RCK		DMWS
QTCK	MWR		
TL3			
QTCK			
TL4			
QTCK			
TL5			
QTCK		RCK	

The following signals, if the indicated condition is satisfied, are present during the entire R04 block:

DMBX
FRVOΔDEDX

R05 BLOCK

TIME	FLIP-FLOPS SET	FLIP-FLOPS RESET	WORKING LOGIC TERMS
TL2		CKO, CK1	DSFT
QTCK	MWR		
TL3			DCTC
QTCK			
TL4			$\overline{\text{DDIBΔDREL}}$
QTCK			
TL5		RVO	
QTCK		RCK	

The following table indicates the conditions affecting the sequence of blocks and the sequence of blocks corresponding to each condition.

R-SEQUENCE

FROM BLOCK		TO BLOCK
R00		R01 IF DDIE
R01	R	R03 IF $\overline{\text{DDIB}}$
	S	R00 IF FMNX R02 IF $\overline{\text{FMNX}}$
R02		R00 IF $\overline{\text{DDIF}}\cdot\overline{\text{FRVO}}$ R01 IF DDIE R03 IF FRVO
R03		R04
R04		R05
R05		R00 IF $\overline{\text{DDIE}}$ R01 IF DDIE

Blocks R01 and R02 of the R-sequence are performed during every data interrupt. During block R01, the Data Control Word (DCW) is read from Memory, the desired incrementation of the address field and the count field is effected and the Data Control Word is restored to the appropriate memory location. During block R02, the address field of the Data Control Word is employed to address the memory location into which or from which data is transferred. If the input/output operation is a write operation, the data in the addressed memory location is restored.

Blocks R03, R04 and R05 of the R-sequence are performed only when the count field of the Data Control Word goes to zero, indicating that the number of characters required by the Data Control Word have been transferred between Memory and the peripheral subsystem. During block R03, the List Pointer Word (LPW) for the channel is read from Memory. The address field and the count field are incremented, if required, and the List Pointer Word is restored to the appropriate memory location. During block R04, a new Data Control Word for the channel is read from the memory location specified by the updated address field of the List Pointer Word. During block R05, the new Data Control Word is stored into the DCW location for the channel.

If the count field of the List Pointer Word goes to zero during block R03, indicating that the last word in the data control list has been read from Memory, the List Pointer Word is not restored to Memory. During block R04, the data control list word in the memory location specified by the address field of the List Pointer Word is read from Memory. This word is the LPW restore word which contains the address of the first word of a new data control list in the address field and a count of the number of words in the list in the count field. During block R05, the LPW restore word is stored into the LPW location for the channel.

FIGURE 173 is a flow chart illustrating the data transfers and other operations occurring during each block of the R-sequence and the conditions effecting the sequence of blocks. Referring to FIGS. 172 and 173, in response to a data interrupt request signal from one or more of the channels and signal FIRT, which indicates

that the R-sequence may be entered, the corresponding data interrupt priority flip-flops are set to the 1-state, transmitting their respective 1-output signals to the data interrupt priority gates 750. Data interrupt priority gates 750 provide an output signal corresponding to the highest priority data interrupt flip-flop set to the 1-state. The output signal of gates 750 is transmitted to B-Gates 317, to address the appropriate Data Control Word and List Pointer Word for the channel, and is also applied to OR-gates 760, to generate a channel select signal to select the highest-priority channel issuing a request for a data interrupt. Signal DDIE issues, presetting the R-Selector Register flip-flops to define block R01, and the W-sequence is inhibited. Signal DDIE also issues in response to signal FIOR, to obtain the Data Control Word for the selected channel during execution of Instruction 07 (GEN). In this event, channel selection is effected by signals DEL1–DEL0, representing channel information, transmitted from the E-Register to OR-gates 760.

During block R01, the Data Control Word for the selected channel is read from the DCW location for that channel. The Data Control Word is stored in the M-Register and applied to the Adder for updating. If signal DDIB is present, indicating that a data interrupt operation is being performed, the R-sequence is continued in block R01. During retrieval of the channel Data Control Word in executing Instruction 07 (GEN), the R-sequence is continued in block R03. In block R01, the contents of bit positions 15 and 16 of the Data Control Word are stored in flip-flops CK0 and CK1 before the Data Control Word is updated. The information in bit positions 15 and 16 determines which character position of the M-Register data is to be transferred to or from. The following table indicates the relationship between bits 16 and 15 stored in flip-flops CK1 and CK0 respectively and the character position of the M-Register to or from which data is to be transferred.

CK1	CK0	Character Position of M-Register
0	0	0
0	1	1
1	0	2
1	1	3

In updating the count field of the Data Control Word in the Adder, the count is incremented by one if a character channel is being serviced. If a single- or double-word channel is being serviced and if the channel contains four characters to be transferred, four is added to the count field of the Data Control Word. If the channel contains less than four characters, bits 15 and 16 of the count field are forced to equal the number of characters to be transferred. The address field of the Data Control Word is incremented in the Adder only when a character is to be transferred to or from the most-significant character position of the M-Register, as indicated by the signal combination $\overline{FM15}\cdot\overline{FM16}$, and the Data Control Word is not being used for the first time for a data interrupt operation, as indicated by signal $\overline{FFTC}$.

If the count field of the Data Control Word does not overflow, i.e. does not go to zero, the output of the Adder is transferred to the M-Register and a write cycle is initiated to store the updated Data Control Word into the DCW location for the channel, for use in the next data interrupt to the channel. If the count field does overflow, overflow flip-flop RVO is set to the 1-state, the output of the Adder is transferred to the M-Register, but a write cycle is not initiated, since a new Data Control Word is to be obtained. The DCW location for the channel thus remains cleared for subsequent storage of the new Data Control Word. Block R01 is terminated and block R02 of the R-sequence is next performed.

During block R02, the contents of the memory location specified by the updated address field of the Data Control Word in the M-Register is read from Memory into the M-Register. If signal DRIT issues, indicating an input/output write operation, the data from the addressed memory location is transferred to the selected channel. If signal $\overline{DRIT}$ issues indicating an input/output read operation, the data in the M-Register is replaced by data transmitted from the channel. A write cycle is then initiated to restore to the addressed memory location the data sent to the channel, if DRIT, or to store the data received from the channel, if $\overline{DRIT}$. Flip-flop FTC, if set to the 1-state, is reset to the 0-state at this time.

If Overflow flip-flop RVO is reset to the 0-state, the data interrupt priority flip-flops are checked to determine if another data interrupt request is present. If so, another R01 block is initiated; otherwise the R-Selector Register goes to R00 and program instruction execution continues. If flip-flop RVO is set to the 1-state, block R03 of the R-sequence is next performed. During block R03, the List Pointer Word is read from the LPW location for the selected channel into the M-Register. The List Pointer Word is applied to the Adder where the count field is incremented by one. If the List Pointer Word is being used for the first time to obtain the address of a Data Control Word for the channel, as indicated by signal $\overline{DDIB}$, the address field of the List Pointer Word is not incremented.

If the count field of the List Pointer Word does not overflow, the adder output is transferred to the M-Register and a write cycle is initiated to restore the updated List Pointer Word into the appropriate memory location. If the count field of the List Pointer Word does overflow during a data interrupt, flip-flop RVO is set to the 1-state and the adder output is transferred to the M-Register, but a write cycle is not initiated so that the LPW location for the channel remains cleared to receive a new List Pointer Word.

The micro-operations of block R04 are performed following block R03. During block R04, the new List Pointer Word, if flip-flop RVO is set to the 1-state, or the new Data Control Word, if flip-flop RVO is reset to the 0-state, is read from the appropriate memory location into the M-Register and restored. During block R05, either the List Pointer Word in the M-Register, if flip-flop RVO is set to the 1-state, is stored into the LPW memory location for the channel, or the Data Control Word in the M-Register, if flip-flop RVO is reset to the 0-state, is stored into the DCW memory location for the channel. If another R-sequence is to be initiated due to the existence of another data interrupt request, block R01 of the R-sequence is next performed. Otherwise, the R-Selector Register goes to R00 and program instruction execution is resumed by the Central Processor.

Program Interrupt

A program interrupt is defined as an unprogrammed transfer of control from the program being executed without altering the contents of the P-Register. As described in the section "Processor Channel," a program interrupt may be initiated upon detection of certain events in the Central Processor, e.g. an alert condition. Program interrupts may also be initiated by the input/output channels of the Input/Output Control Unit when either of the following conditions arises:

(a) Termination of any operation which previously placed an input/output channel and its associated peripheral subsystem into operation.

(b) The occurrence of a special event in the peripheral subsystem.

A program interrupt provides for the transfer of program control to an instruction placed in a Program Interrupt Word (PIW) location and a Program Interrupt Second

Address Word (PSW) location. Each input/output channel has its own PIW location and PSW location. Upon granting of a program interrupt request from a channel, the Program Processor takes its next instruction from the PIW and PSW locations for the channel requesting the program interrupt. Together, the PIW and PSW locations contain a two-address instruction that, depending upon the instruction, determines whether program interrupt requests for the associated channel are to be honored or not.

Upon completion of an input/output operation, the input/output channel always requests that the Central Processor interrupt the instruction processing sequence. When the Central Processor acknowledges the request, the instruction in the PIW location is executed as the next instruction. The PIW location must contain either Instruction 17, Store Program Counter and Branch (SPB), or Instruction 07, General (GEN), coded to initiate a request status operation. If an instruction other than one of the above is in the PIW location, the Central Processor halts with a control word alert.

The address field of the instruction in the PIW location cannot be modified. The word in the PSW location serves as the SAS word for developing the second address of the instruction in the PIW location. The word in the PSW location may be one of the following:

(a) An operand which causes the contents of the P-Register to be stored into the PSW location, if the instruction in the PIW location is Instruction 17 (SPB), or which indicates the location into which the peripheral subsystem status is to be stored, if the instruction in the PIW location is Instruction 07 (GEN).

(b) An operand pointer which indicates the location into which the P-Register contents are to be stored, if the instruction in the PIW location is Instruction 17 (SPB), or which indicates the location into which the peripheral subsystem status is to be stored, if the instruction in the PIW location is Instruction 07 (GEN).

(c) An operand link for developing the second address of the instruction in the PIW location.

When a program interrupt request is granted and the PIW location contains an Instruction 17 (SPB), the contents of the P-Register are stored in the location specified by the second address of the instruction. The address field of the Instruction 17 in the PIW location is transferred to the P-Register, effectively interrupting the program and transferring control to a subroutine designed to service the cause of the program interrupt. Execution of Instruction 17 (SPB) sets flip-flop PSA and the Program Interrupt indicator flip-flop PIT to the 1-state, preventing further program interrupts from being granted until the correct program interrupt is terminated.

The program interrupt subroutine will normally terminate the request for program interrupt by the channel and preserve the status of the Central Processor by performing a "request status of processor" operation [Instruction 67 (CPO)] during the subroutine, so that the Central Processor can be returned to that status upon conclusion of the program interrupt subroutine. The subroutine services the cause of the program interrupt and return control to the interrupted program by executing an Instruction 10 (BRU).

When a program interrupt request is granted and the PIW location contains an Instruction 07 (GEN), the status of the peripheral subsystem is stored in the location determined by the address field of the instruction in the PSW location. The request for program interrupt by the channel is terminated and the next instruction is obtained under control of the P-Register, which still refers to the program that was momentarily interrupted. Flip-flop PSA and the Program Interrupt indicator flip-flop PIT are not set to the 1-state. Thus, an Instruction 07 (GEN) in the PIW location provides a means for cancelling the request for program interrupt and continuing the processing of instructions in the main program.

PROGRAM INTERRUPT LOGIC

FIGURE 174 illustrates the program interrupt logic which forms part of the common logic of the Input/Output Control Unit. Referring to FIG. 174, each of the channels #1–#0 may request a program interrupt by issuance of signals DP11–DP10 respectively. The program interrupt request signals DP11–DP10 from the channels are applied to the input terminals of a plugboard 780. The signals ZP11–ZP10 from the output terminals of plugboard 780 are applied to gates 782.

Program interrupt scanner 784, comprising flip-flops PS0, PS1 and PS2, is a three-bit binary counter which has a unique output signal combination for each of the eight channels. Program interrupt scanner 784 repeatedly counts from one through zero when a program interrupt is not in progress. The output of program interrupt scanner 784 is applied to gates 782. When coincidence between a program interrupt request and its corresponding count in the program interrupt scanner occurs, signal DPSC issues and advance of the count in program interrupt scanner 784 is inhibited. For example, if program interrupt request signal ZP13 is applied to gates 780, program interrupt scanner 784 will stop when flip-flops PS0 and PS1 are set to the 1-state and flip-flop PS2 is reset to the 0-state. The output signals of program interrupt scanner 784 are applied to B-Gates 317 to address the PIW and PSW locations corresponding to the channel for which the program interrupt is granted.

Plugboard 780 enables the PIW and PSW locations of the channels to be grouped with the respective LPW and DCW locations. In the event two or more program interrupt requests occur simultaneously, priority is granted in accordance with the relative positions of the program interrupt request signals in the series ZP11–ZP10, ZP11 representing the highest-priority request and signal ZP10 representing the lowest-priority request. When flip-flop PIT is reset to the 0-state, program interrupt scanner 784 again starts counting from one. When the processor channel requests a program interrupt simultaneous with a request from an input/output channel, the processor channel request is given priority. Program interrupt requests are not granted until completion of execution of the current instruction in the main program.

Signal DPSC, which issues from gates 782 in response to coincidence of a channel program interrupt request and the count in program interrupt scanner 784, causes signal DPDC to issue, setting Program Interrupt Address Enable flip-flop PIA to the 1-state. Signal DPDC issues and flip-flop PIA is also set to the 1-state in response to the signal FPAD, which issues when the processor channel requests a program interrupt. Signal FPIA causes the instruction in the PIW location for the channel to be read from Memory during block W00 of the W-sequence. During the W01 block of the W-sequence, signal FPIA causes the word in the PSW location to be read from Memory, to develop the second address of the instruction in the PIW location. FIGURE 175 illustrates the variation in the micro-operations of block W00, illustrated in FIG. 119, when flip-flop PIA is set to the 1-state during a program interrupt. FIGURE 176 illustrates the variations in the micro-operations of block W01, illustrated in FIG. 121, when flip-flop PIA is set to the 1-state during a program interrupt.

Referring to FIG. 175, the 1-output signal FPIA of Program Interrupt Address Enable flip-flop PIA inhibits signal DDBX and causes signal DPIB to issue. Signal DPIB permits the B-Gates 317 to sample the output signals of program interrupt scanner 784 to obtain the address of the appropriate PIW location for the channel requesting the program interrupt. The instruction word in the PIW location is read from Memory into the M-Register. During time period TL3, signal FPIA inhibits the advance of the count in the P-Register. Signal FPIA also inhibits the transfer by signal DMQX of the address

control field of the instruction word in the M-Register to the Q-Register. During time period TL4, Manual flip-flop MAN is set to the 1-state if the instruction in the PIW location is neither Instruction .07 (GEN) or Instruction 17 (SPB).

During time period TL5 of a subsequent W00 block, occurring after execution of an Instruction 67 (CPO) which resets the Program Interrupt indicator flip-flop PIT to the 0-state, flip-flop PSA is reset to the 0-state when the instruction to be executed is an Instruction 10 (BRU) which returns the Central Processor to the interrupted program. Signal $\overline{\text{FPSA}}$ permits another program interrupt request to be granted.

Referring to FIG. 176, which illustrates the micro-operations of the W01 block peculiar to a program interrupt subroutine, signal FPIA inhibits signal DDBX and causes signal DPIB to issue. Signal DPIB permits the B-Gate 317 to sample the output signals of program interrupt scanner 784 to obtain the address of the appropriate PSW location for the channel requesting the program interrupt. Flip-flop PSA is set to the 1-state if the instruction in the PIW location is an Instruction 17 (SPB). Signal FPSA inhibits the setting of flip-flop PIA to the 1-state by subsequent program interrupt requests. During time period TL3, signal FPIA inhibits the advance of the count in the P-Register. During time period TL5, flip-flop PSA is reset to the 0-state upon execution of an Instruction 10 (BRU) which returns the Central Processor to the interrupted program.

FIGURE 177 is a flow diagram illustrating the operations occurring during a program interrupt.

What is claimed as new and desired to be secured by Letters Patent of the United States is:

1. In a data processing system including a memory, the combination comprising: first control means having a plurality of controllable states, each of said states causing a corresponding control signal to be generated, second control means responsive to each of said control signals generated by said first control means for performing a predetermined series of operations in said data processing system, a first portion of said predetermined series of operations including a memory read operation and a second portion of said predetermined series of operations including a memory write operation, a register for storing a command item, and first means responsive to the command item in said register for controlling the sequence of states of said first control means and second means responsive to the command item in said register for selectively causing one or both portions of the predetermined series of operations corresponding to each control signal generated by said first control means to be performed by said second control means.

2. In a data processing system, the combination comprising: a register for storing a command item, first control means for controlling execution of a data processing function required by the command item in said register, said first control means having a plurality of controllable states, each of said states causing a corresponding control signal to be generated, means responsive to the command item in said register for controlling the sequence of states of said first control means, second control means for controlling performance of an input/output data transfer, said second control means having a plurality of controllable states, each of said states causing a corresponding control signal to be generated, means responsive to conditions occurring during an input/output data transfer for controlling the sequence of states of said second control means, third control means normally responsive to each of said control signals generated by said first control means for performing a predetermined series of operations in said data processing system, and means responsive to said second control means for inhibiting the generation of control signals by said first control means and for causing said third control means to respond to each of said control signals generated by said second control means for performing a predetermined series of operations in said data processing system.

3. In a data processing system the combination comprising: control means having a plurality of controllable states, each of said states causing a corresponding control signal to be generated, timing means for providing timing signals, means responsive to each of said control signals generated by said control means and responsive to the timing signals generated by said timing means for performing a predetermined series of operations in said data processing system, a register for storing a command item, first means responsive to the command item in said register for controlling the sequence of states of said control means, and second means responsive to the command item in said register for selectively inhibiting the performance of predetermined operations within said predetermined series of operations.

4. In a data processing system, the combination comprising: a register for storing a command item, first control means having a plurality of controllable states, each of said states of said first control means causing a corresponding control signal to be generated, means responsive to the command item in said register for controlling the sequence of states of said first control means, second control means for controlling performance of a data transfer, said second control means having a plurality of controllable states, each of said states of said second control means causing a corresponding control signal to be generated, third control means normally responsive to each of said control signals generated by said first control means for performing a predetermined series of operations, and means responsive to said second control means for inhibiting the generation of control signals by said first control means and for causing said third control means to respond to each of said control signals generated by said second control means for performing a predetermined series of operations in said data processing system.

References Cited

UNITED STATES PATENTS

2,910,236	10/1959	Harper	340—172.5 X
3,061,192	10/1962	Terzian	340—172.5 X
3,067,937	12/1962	Hinkein et al.	340—172.5 X
3,094,610	6/1963	Humphrey	340—172.5 X
3,215,987	1/1965	Terzian	340—172.5 X
3,245,044	4/1966	Meade	340—172.5
3,302,183	1/1967	Bennett et al.	340—172.5

PAUL J. HENON, *Primary Examiner.*

UNITED STATES PATENT OFFICE

CERTIFICATE OF CORRECTION

Patent No. 3,368,205 February 6, 1968

Robert D. Hunter et al.

It is certified that error appears in the above identified patent and that said Letters Patent are hereby corrected as shown below:

Column 4, after line 75, insert -- Thomas J. Beatson, Frank J. Boyle, Byron F. Burch, Jr., Robert D. Hunter, and Daniel W. Scott, as defined by the claims of their application, Serial No. 448,194, filed April 14, 1965 --.

Signed and sealed this 12th day of August 1969.

(SEAL)
Attest:

Edward M. Fletcher, Jr.
Attesting Officer

WILLIAM E. SCHUYLER, JR.
Commissioner of Patents