April 4, 1967 S. HENIG 3,312,949

STACK-FORMING SYNCHRONIZER FOR A SORTING MACHINE

Filed April 6, 1964 9 Sheets-Sheet 1

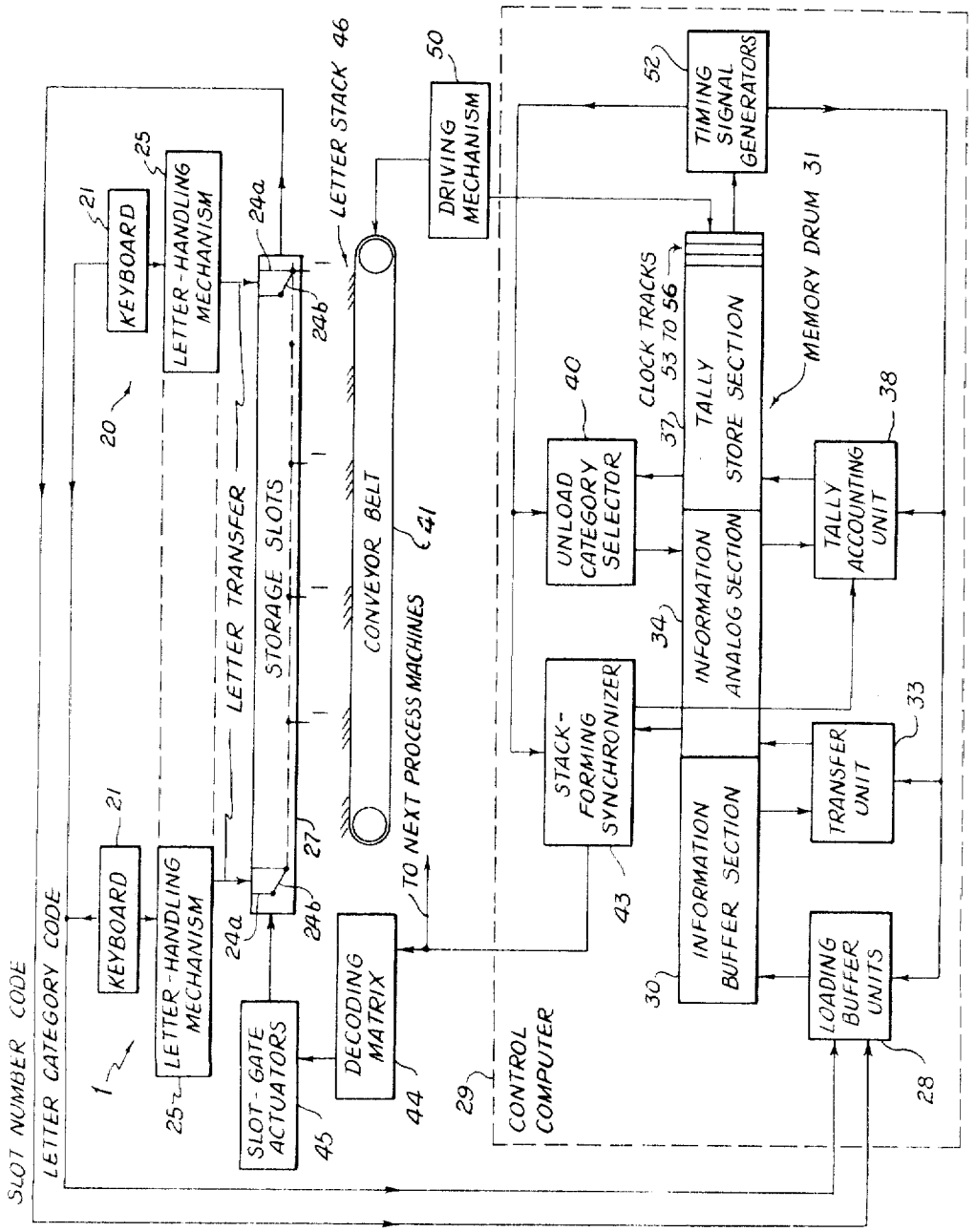


Fig. 1

INVENTOR

Seymour Henig

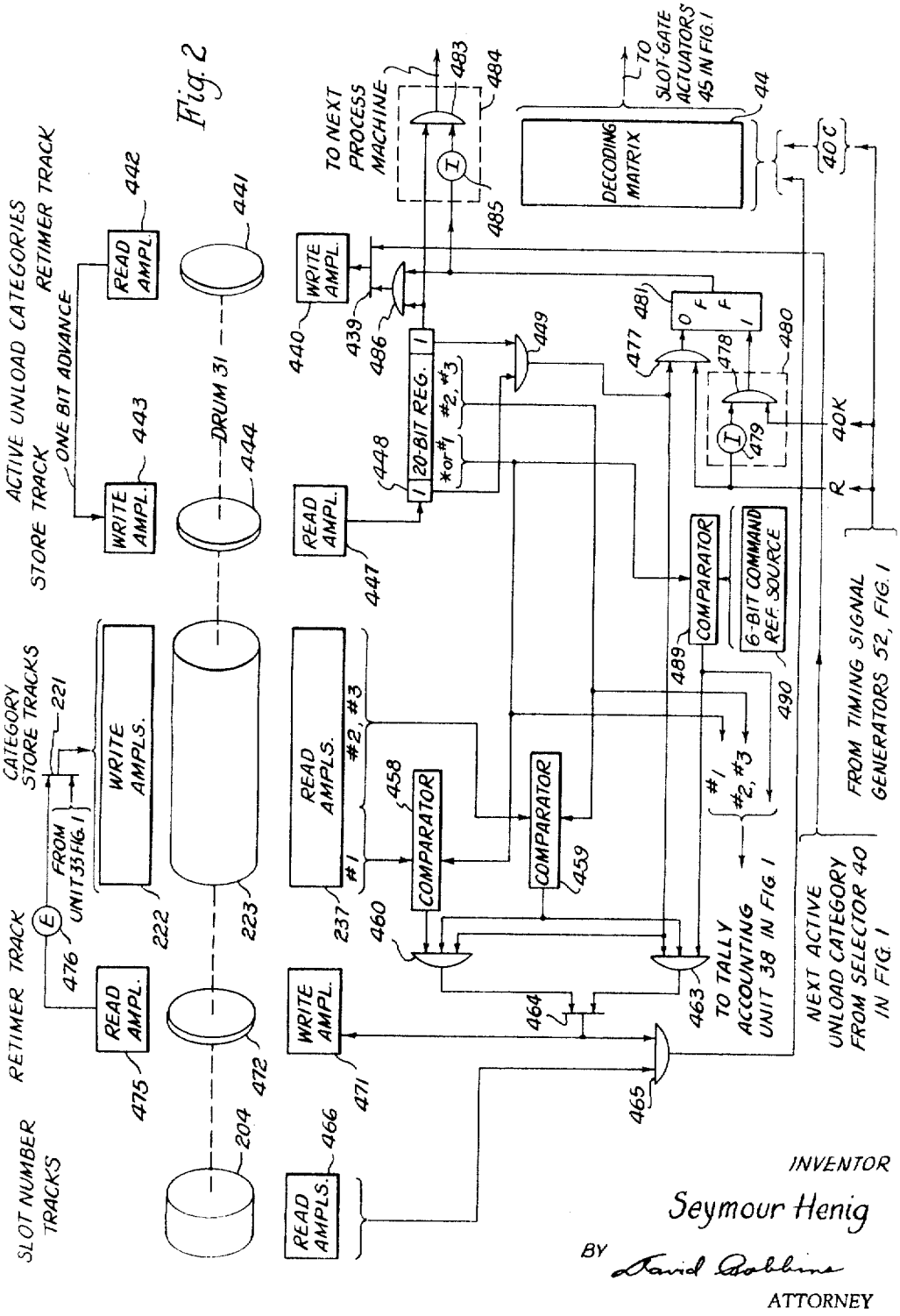
BY David Robbins

ATTORNEY

April 4, 1967 S. HENIG 3,312,949

STACK-FORMING SYNCHRONIZER FOR A SORTING MACHINE

Filed April 6, 1964 9 Sheets-Sheet 5



April 4, 1967 S. HENIG 3,312,949

STACK-FORMING SYNCHRONIZER FOR A SORTING MACHINE

Filed April 6, 1964 9 Sheets-Sheet 3

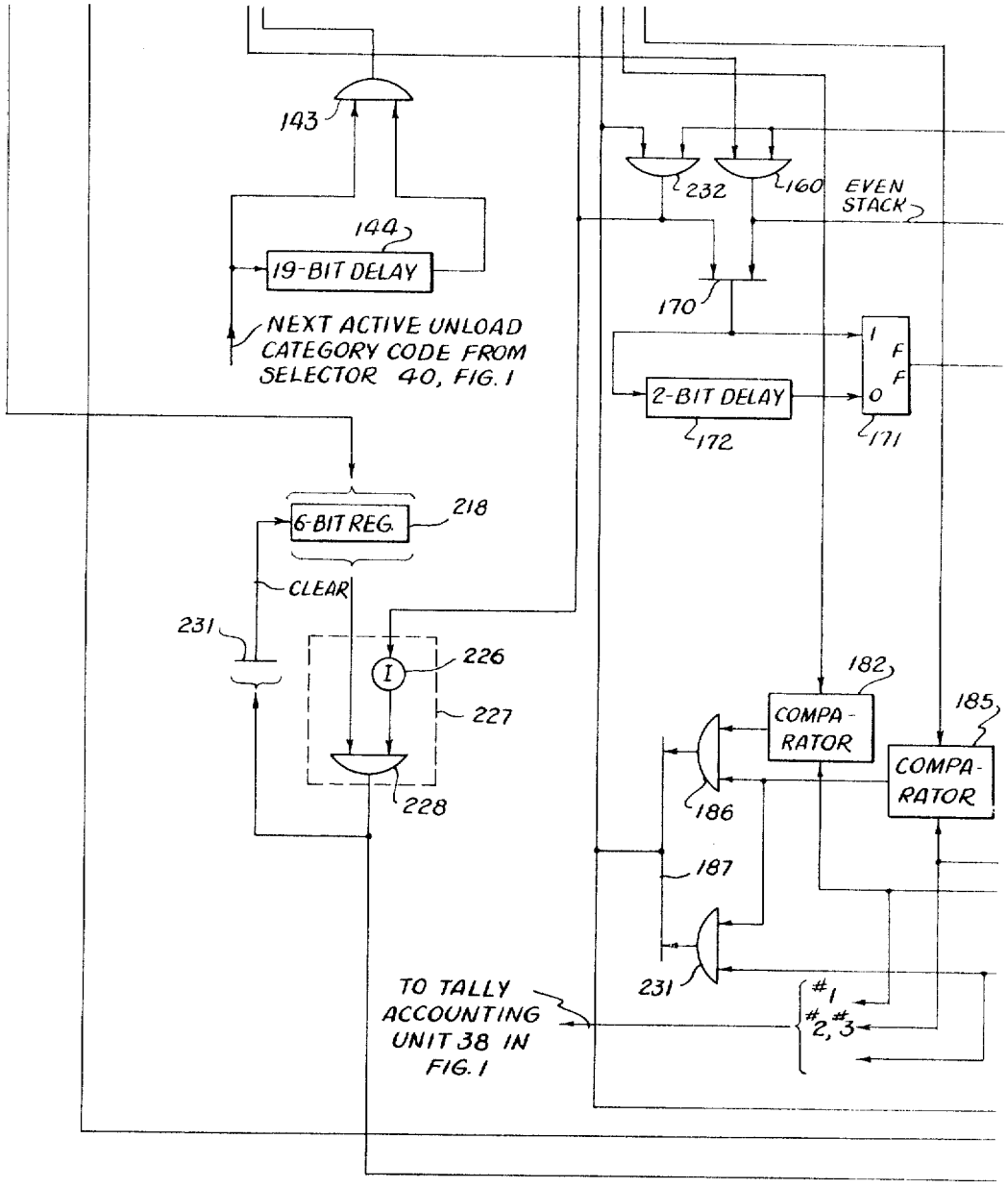


Fig. 3C

Fig. 3B	Fig. 3A
Fig. 3C	Fig. 3D

Fig. 3

INVENTOR

Seymour Henig

BY David Gabbine

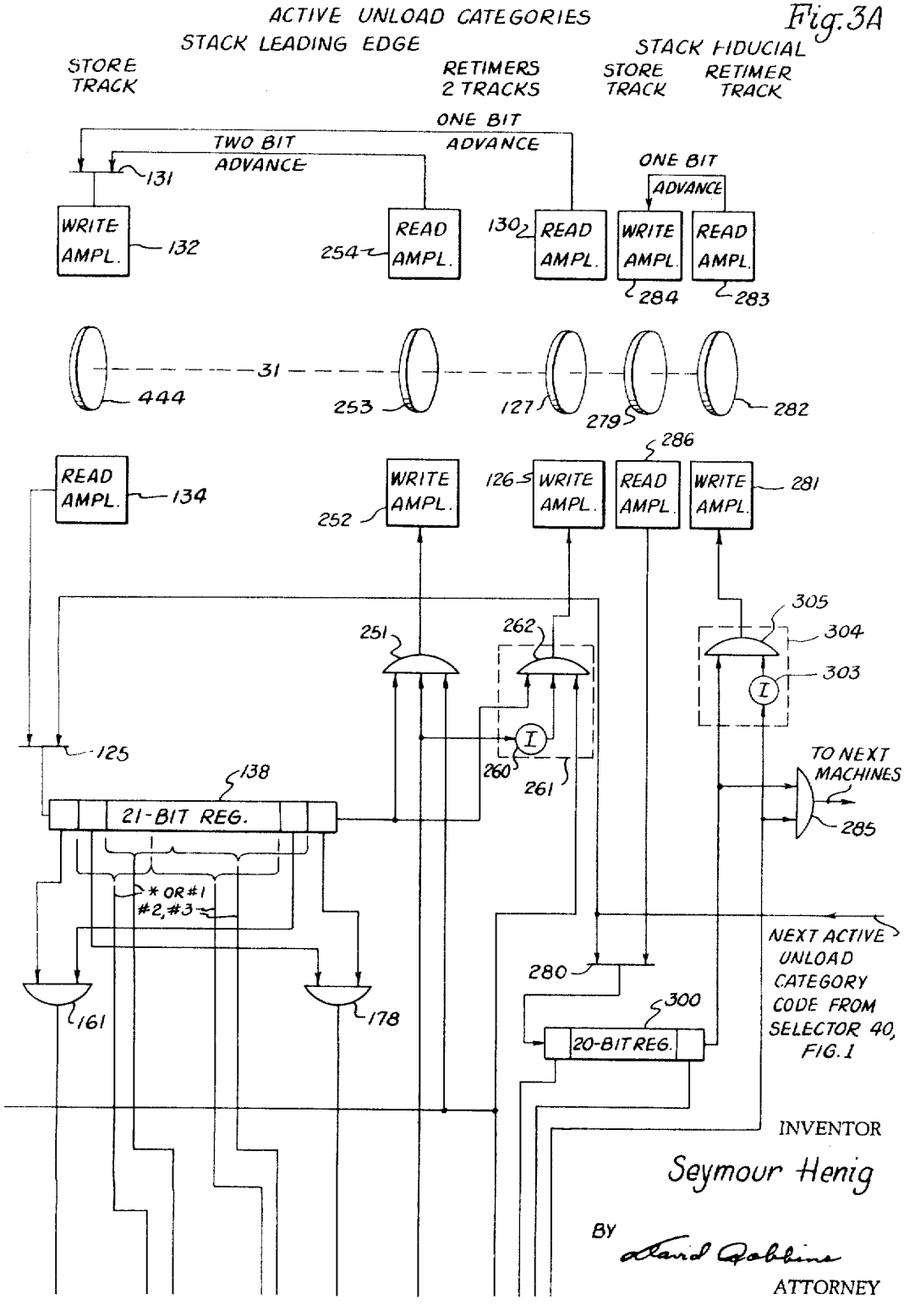
ATTORNEY

April 4, 1967 S. HENIG 3,312,949

STACK-FORMING SYNCHRONIZER FOR A SORTING MACHINE

Filed April 6, 1964 9 Sheets-Sheet 4

Fig. 3A



INVENTOR

Seymour Henig

BY David Gabbins

ATTORNEY

STACK-FORMING SYNCHRONIZER FOR A SORTING MACHINE

Filed April 6, 1964 9 Sheets-Sheet 5

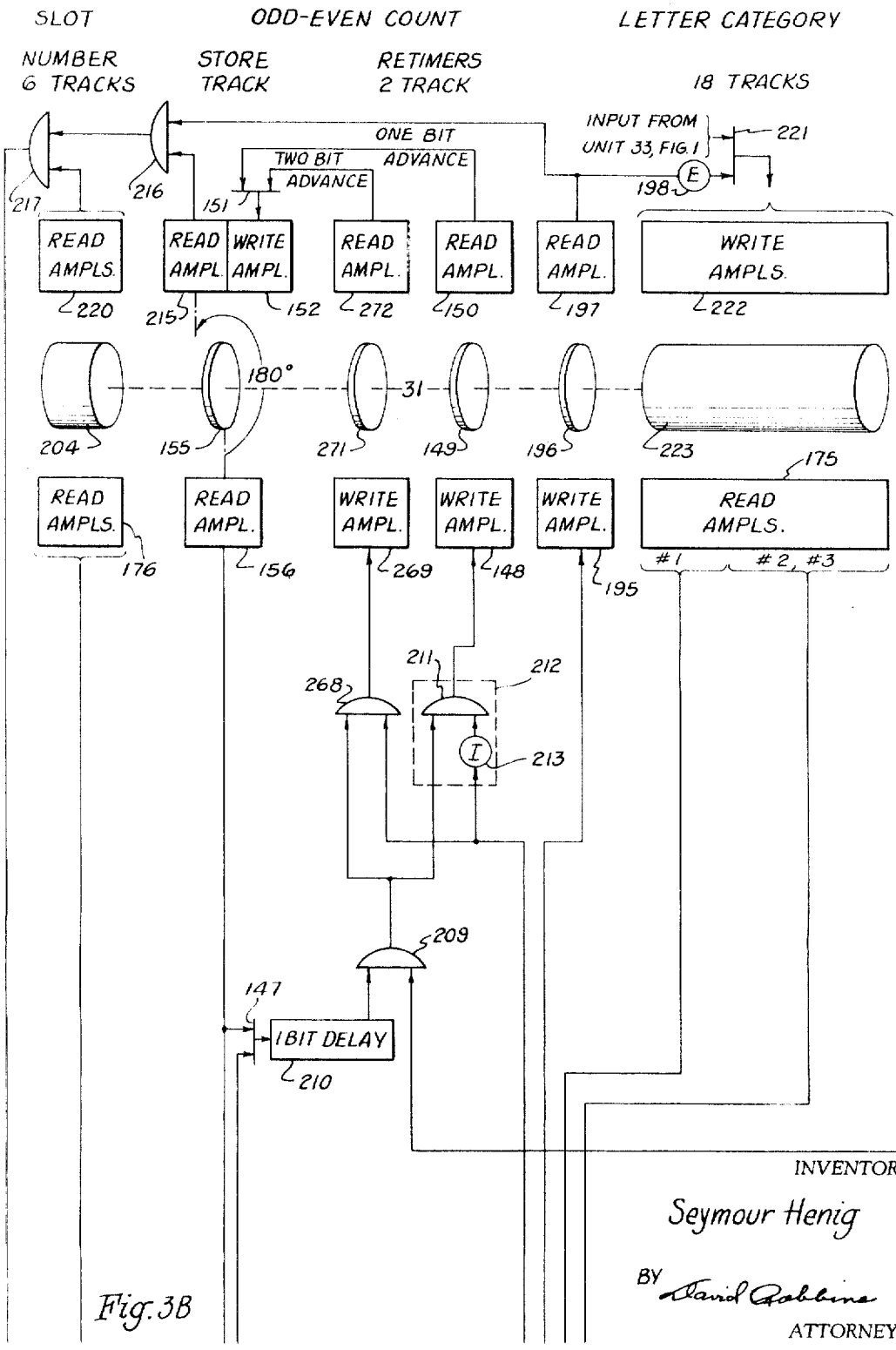


Fig. 3B

INVENTOR

Seymour Henig

BY David Goblirsch

ATTORNEY

STACK-FORMING SYNCHRONIZER FOR A SORTING MACHINE

Filed April 6, 1964 9 Sheets-Sheet 6

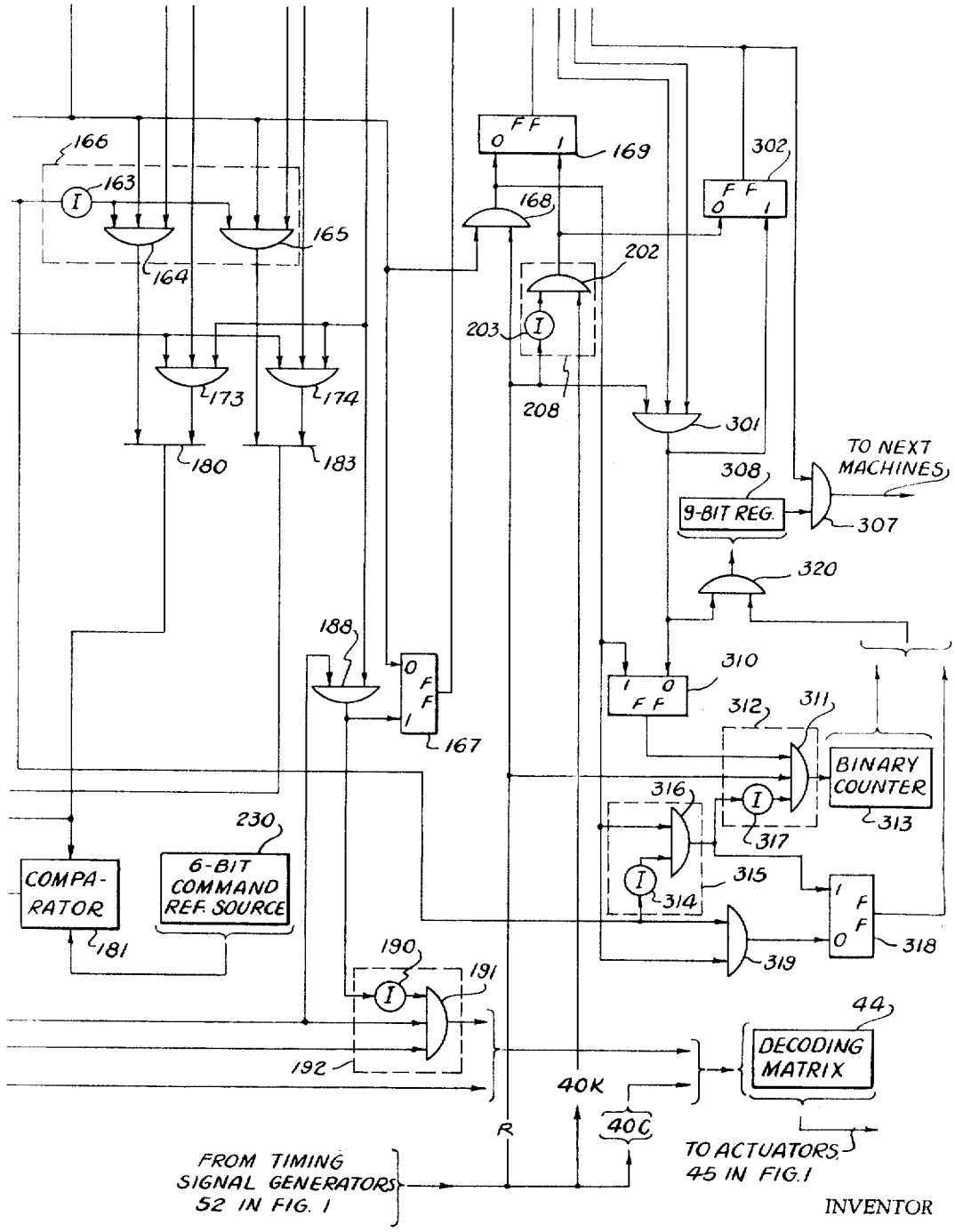


Fig. 3D

INVENTOR

Seymour Henig

BY David Gabbins

ATTORNEY

STACK-FORMING SYNCHRONIZER FOR A SORTING MACHINE

Filed April 6, 1964 9 Sheets-Sheet 7

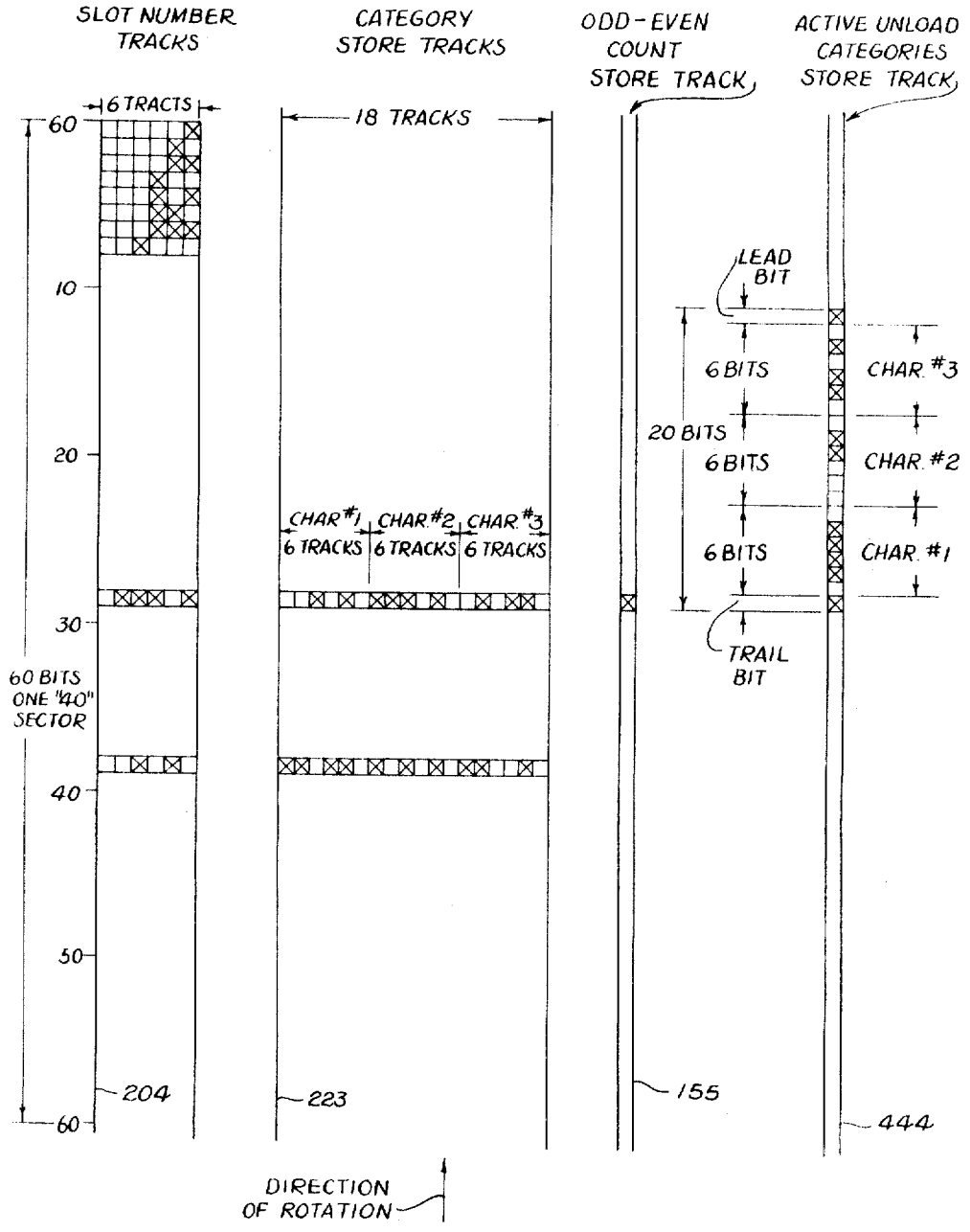


Fig. 4

INVENTOR
Seymour Henig

BY
Daniel Robbins
ATTORNEY

April 4, 1967 S. HENIG 3,312,949

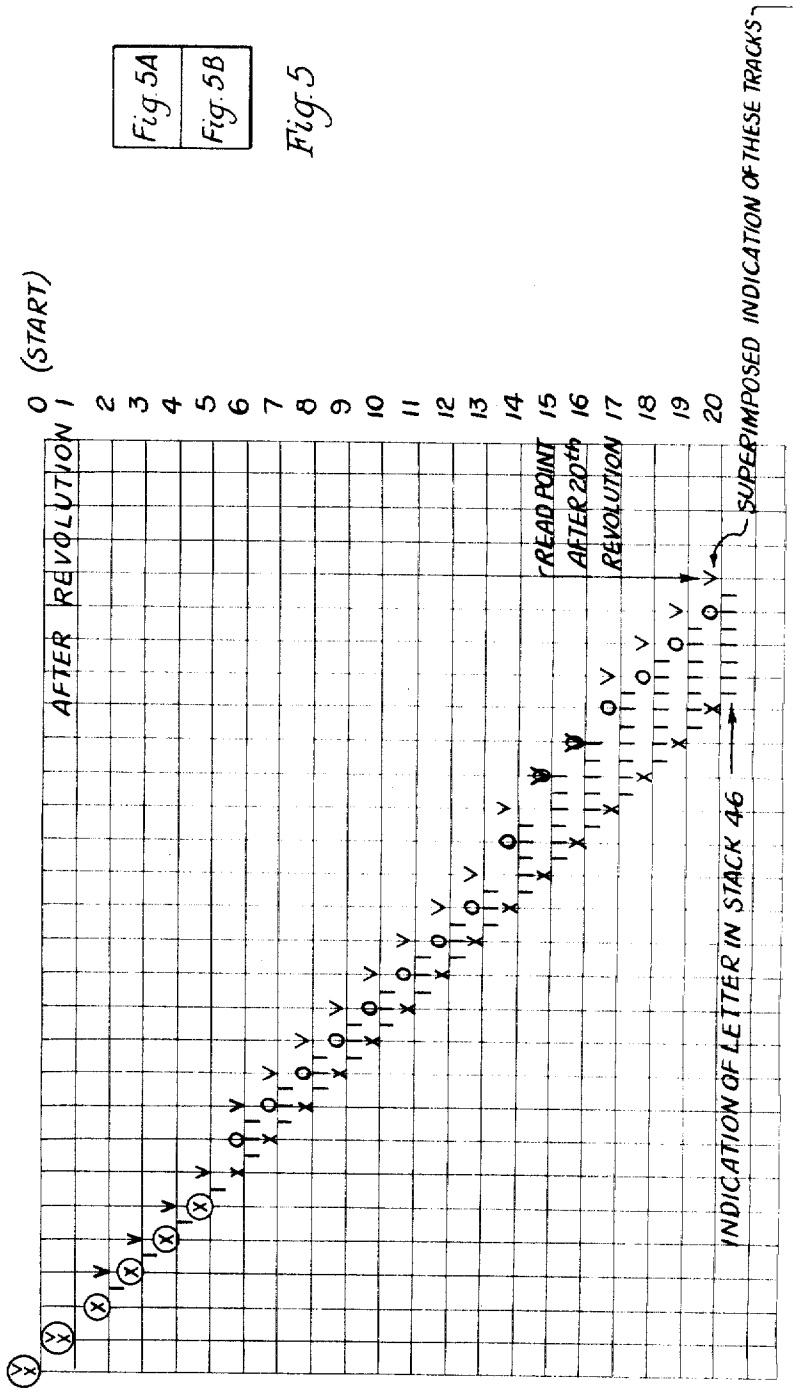
STACK-FORMING SYNCHRONIZER FOR A SORTING MACHINE

Filed April 6, 1964 9 Sheets-Sheet 8

Fig. 5

Fig. 5A
Fig. 5B

Fig. 5A



INVENTOR
Seymour Henig
BY David Gabbino
ATTORNEY

STACK-FORMING SYNCHRONIZER FOR A SORTING MACHINE

Filed April 6, 1964 9 Sheets-Sheet 9

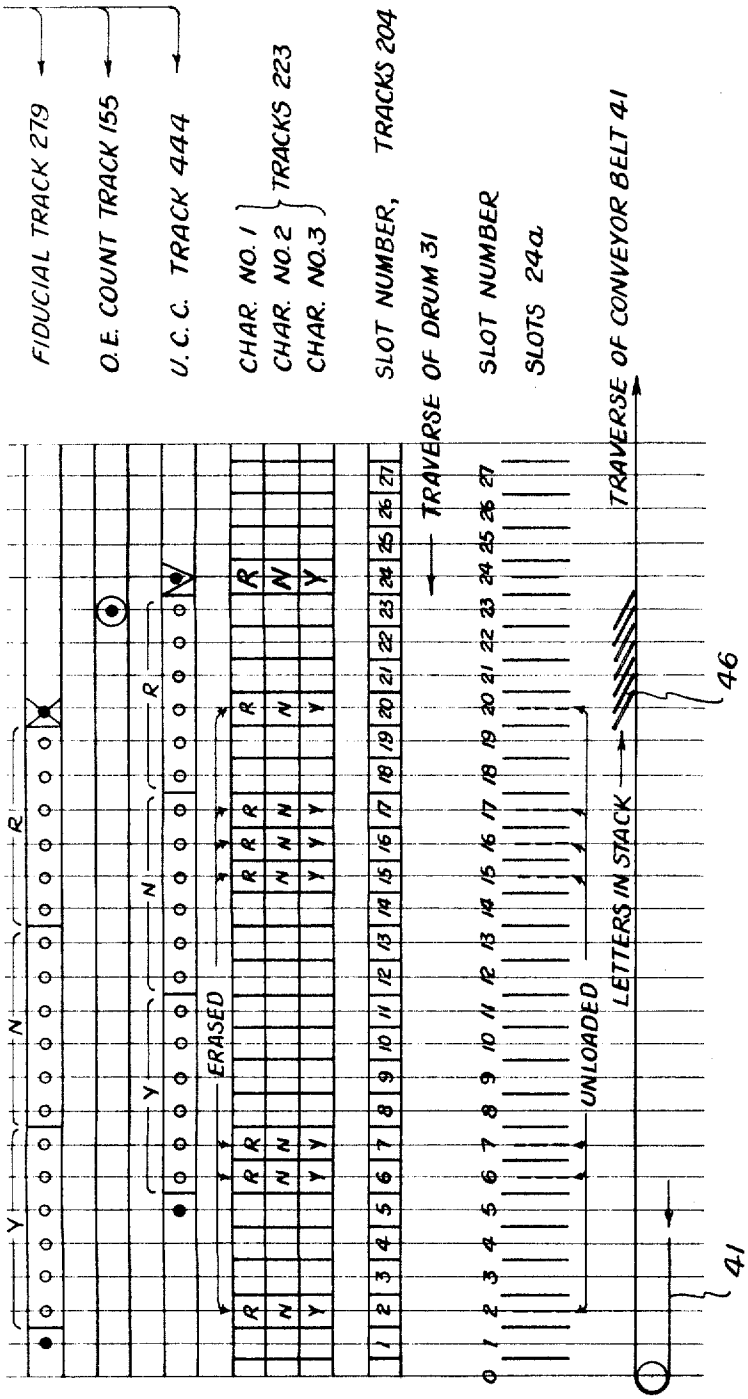


Fig. 5B

INVENTOR
Seymour Henig
BY David Gabbins
ATTORNEY

United States Patent Office

3,312,949

Patented Apr. 4, 1967

3,312,949

STACK-FORMING SYNCHRONIZER FOR A SORTING MACHINE

Seymour Henig, Kensington, Md., assignor to the United States of America as represented by the Secretary of Commerce

Filed Apr. 6, 1964, Ser. No. 357,846

12 Claims. (Cl. 340—172.5)

This invention relates to a stack-forming synchronizer and in particular to one that may be used in a sorting machine in which physically similar articles are deposited randomly into individual storage slots emplaced in a rack. In this type of machine, each article's category information along with its storage location is filed in a memory. In order to sort, the memory is searched periodically for articles of like category which are then unloaded to a stack on a single conveyor belt. The belt is used for different categories during successive periods and emptied slots are immediately available for reuse by new articles of any category.

The present application is a continuation-in-part of U.S. patent application Serial No. 289,761 entitled "Sorting Machine Providing Self-Optimizing Inventory Reduction," which was filed by Seymour Henig and Ervin C. Palasky on June 21, 1963.

The sorting machine, described in the above patent application, includes several stations, each having a rack containing a plurality of slots, each slot having a gate positioned above a common conveyor belt. An operator, reading the destination of a letter to be sorted, operates a keyboard to record the related letter category code in the memory of a control computer. The letter is then dropped into the first empty slot and the number code of the slot is generated and stored in the memory.

Using the generated number code, the computer places the category code for each letter in relation to an identical number code prerecorded in a section of the memory whose arrangement is analogous to the slots' physical locations in the rack. A running inventory with respect to categories is kept for the letters stored in the slots. Periodically, the computer makes a determination, premised on optimum inventory reduction, of the next category to be unloaded from the slots. The code of this category is then recorded in a section of the memory which retains a list of these active unload category codes of the stacks forming on the conveyor belt.

One embodiment of the present stack-forming synchronizer may be incorporated in a sorting machine of this type that uses a drum as the memory. The number of each slot in the rack is prerecorded in respective tracks on the drum and a letter category code may be written in tracks of the drum in alignment with the related slot number code. At the start of stack forming, each active unload category code is written on a track of this drum in alignment with the number for slot 1.

During one revolution of the memory drum, each active unload category code is read out and compared with the aligned letter category code, that is likewise read out of the memory drum. On the succeeding revolution of the drum, each active code is advanced to a new alignment with that letter category code and slot number code written in respective tracks of the drum for the next one of the slots in the rack. Thus, on starting at the first slot, each active category code is compared with each letter category code written in the tracks of the memory drum for each of the slots in the rack. When an identity is found, the related slot number code is read out of the drum and applied to a decoding matrix to control the gate of a slot containing the letter related to the category being unloaded on the conveyor belt. The stack obtains its spread of letters from the cumulative difference provided by running the belt at a speed slightly less than one slot advance for each revolution of the memory drum.

In another embodiment of the present invention, the stack-forming synchronizer is provided with an arrangement that regulates the spacing of the letters in each stack. Since letter-spacing can have many ramifications, it is necessary to define the kind of regulation that is being effected. Slot pitch is a convenient unit of linear measurement here, so the regulation is defined as follows: each letter added to the stack is positioned one-half slot ahead of the letter just previously added to the stack. This embodiment uses the same general method of active unload category code storage and traverse as the first embodiment. However, the conveyor belt, instead of lagging, runs in step with the memory drum in the nominal exact ratio of one slot advance per revolution. Further, the odd-count additions are used to fill the half-slot places in a stack, and the even-count additions are used to fill the whole-slot places. (The odd-count additions are the first, third, fifth, etc. letters added to a stack, while the even-count additions are the second, fourth, sixth, etc. letters added to the stack.)

In the latter embodiment, a record is kept of the leading edge and another record is kept of the zero-letter position of the stack. Utilizing these records, a count of the number of letters in each stack and information relating to each completed stack on the conveyor belt is sent to the next stack-processing machine.

Accordingly, it is an object of the present invention to provide a stack-forming synchronizer for a sorting machine.

Another object is to provide a stack-forming synchronizer incorporating an arrangement for regulating the spacing of articles in each stack.

Another object is to provide a stack-forming synchronizer that generates a count of the articles in each stack and data concerning the position of each stack formed on a conveyor belt.

Other objects and advantages of the invention will become apparent from the following description of the annexed drawings in which like reference numerals designate like parts throughout the figures thereof and wherein:

FIG. 1 is a block diagram of a sorting machine using the present invention;

FIG. 2 is a first embodiment of the present invention;

FIG. 3 is a block diagram showing the manner in which FIGS. 3A to 3D are assembled;

FIGS. 3A to 3D comprise a second embodiment of the present invention;

FIG. 4 shows selected tracks on the memory drum **31** in FIG. 1;

FIG. 5 is a block diagram showing the assembly of FIGS. 5A and 5B; and

FIGS. 5A and 5B comprise event schedule used to explain the operation of the embodiment illustrated in FIGS. 3A to 3D.

The block diagram (FIG. 1)

The number of sorting stations employed in a particular installation is determined by the volume of sorting to be accomplished. Here twenty stations are used and sorting stations **1** and **20** are shown in FIG. 1. Each station includes a keyboard **21** for controlling a signal generating mechanism that is generally used in a Teletypewriter to develop signal patterns in binary code. An operator, by striking selected keys on the keyboard, transcribes the address of each letter into a category code in accordance with an appropriate scheme list of categories.

The scheme list, in one example, may comprise two hundred direct and secondary categories. A direct category is one that requires no further sorting en route

or at the post office of origin, while a secondary category requires further sorting. In this example, the list includes twenty secondary categories, sixteen representing states and four representing groups of states, one hundred and seventy-five direct categories, and five spares.

Each secondary category is encoded in binary in a four-position code by means of two characters and a command symbol, represented by an asterisk. New York State, for example, is designated by *—NY. (Note that the second position of the code is blank.) Each direct category is encoded in binary in a four-position code by the characters representing a related secondary category and another prefix. Rochester, N.Y. is therefore designated at __RNY. (Note that here the first position of the code is blank.) The number of direct categories assigned to each secondary is determined by the statistical probability of the former.

After observing the address of a letter, an operator generates the related category code by striking the appropriate keys on keyboard **21**. When the address is in a direct category, the operator strikes the three characters of the category code, but when the address is a secondary category, the operator strikes the space bar and the two characters of the code. The command code * is pre-recorded in the scheme list categories track in memory drum **31** and is provided in control computer **29**, when required, as indicated in the above patent application.

After the category code is struck, the letter-handling mechanism **25** is activated and the letter is dropped into a pocket of a shuttle, not shown. The shuttle traverses a rack **27** of sorting slots **24***a* and deposits the letter in the first vacant slot encountered. When the last-mentioned operation is performed, a code representing the number of the slot into which the letter was deposited is generated.

The letter category code and slot number code are applied to loading buffer units **28** in control computer **29** and are read into the information buffer section **30** of memory drum **31**. The codes are then transferred from section **30** through transfer unit **33** to information analog section **34**.

The data stored in section **34** is considered to be "information analog," since the slot number and related letter category code are positioned, side-by-side, in successive sectors of memory drum **31** in the same sequence as the corresponding letters stored in slots **24***a*. It is understood, however, that the codes need not be written on the drum side-by-side so long as each letter category code is provided with a fixed displacement relative to its related number code and the number codes are read out of the memory drum in the same sequence as the slots **24***a* are positioned in rack **27**.

Each category code in the scheme list is pre-recorded in certain tracks in the tally store section **37**. A threshold code, representing the minimum number of letters required for each category before it is selected for unloading, is also recorded in a track in this section. The tally accounting unit **38** keeps a running tally of the number of letters in each category stored in each slot **24***a*, as indicated by the recordings in analog section **34**, and writes this tally in a track in section **37**. All the information used to provide self-optimizing inventory reduction is now present in memory drum **31**.

In the operations performed in obtaining optimum inventory reduction, the predominant categories are selected on an alternate basis with a programed category, i.e., the categories in the scheme list are unloaded in turn on an alternate basis with the categories having the greatest number of letters in slots **24***a* when an active unload category is selected. Accordingly, during certain revolutions of memory drum **31**, unload category selector **40** performs one of the following operations:

(a) The predominant category, either direct or secondary, is determined and becomes the next active unload category. (Instead of selecting an active unload category code purely by predominance, the selection may be further constrained by a threshold, so that the category code selected is related to a tally code of desired magnitude.)

(b) The magnitude of a tally of a programed, direct category is compared with the magnitude of its recorded threshold. At the same time, the predominant category is determined. If the tally exceeds its threshold, the direct category is selected. If not, the direct category is skipped and the predominant category becomes the next active unload category.

(c) A search is made for the predominant category. Concurrently, a determination is made of the sum of the tallies of a programed, secondary category and all its associated direct categories, which includes those skipped. If the magnitude of the sum exceeds the magnitude of the threshold recorded for the secondary category, the letters in the secondary and related direct categories are unloaded together. If the tally fails to exceed its threshold, the predominant category is selected.

After selector **40** has determined the next category of letters to be unloaded from slots **24***a*, the code representing this category is written on a track in information analog section **34**. Each such stack category code written on this track is compared in stack-forming synchronizer **43** with each letter category code in the analog section positioned adjacent the number code of the slot containing the letter. When an identity is found, the number code, related to the letter category code, is read out and passed in binary form, as one output of the stack-forming synchronizer, to decoding matrix **44**. The matrix converts the binary code to a pulse on a line connected to one of the slot-gate actuators **45**. The energized actuator opens the gate **24***b* of the related slot to drop the letter on the stack **46** forming on conveyor belt **41**.

To optimize inventory reduction, the stack-forming synchronizer **43** makes no exceptions for unloading the letters in slots **24***a*. The synchronizer therefore instructs tally accounting unit **38** to discount any letter in an active unload category that is added to a slot in advance of the related stack being formed on conveyor belt **41**. Thus, the letters that are dropped in slots positioned in advance of the stack will not affect the tally, which is used simply for the selection of the next active unload category.

Another output of synchronizer **43** includes a count of the letters in each stack formed on conveyor belt **41**. The count could be used, for example, on the label supplied to the next process machine. Still another output provides an indication of the end of each stack, which could be used in the following process machines to inform a labeller, bundler and poucher as to the location of each stack on conveyor belt **41**.

The driving mechanism **50** comprises a synchronous motor for memory drum **31** and a separate synchronous motor for conveyor belt **41**. The motors are driven from the same source, and in one embodiment of the present stack-forming synchronizer, the motor for the belt is provided with a gear reduction arrangement, so that the time required for a reference point on the belt to traverse the width of one slot **24***a*, plus a fixed time interval, is substantially equal to the time required for one revolution of memory drum **31**. In another embodiment, the requisite time for the reference point to traverse the width of one slot is nominally equal to the requisite time for one revolution of the memory drum **31**.

The bits engraved in clock tracks **53** to **56** are used to control the timing signal generators **52**, which provide the timing signals for control computer **29**, as explained in detail in the above patent application. There are twenty-four hundred bits engraved in track **53**, and there are forty equally spaced engraved bits on track **54**, dividing memory drum **31** into forty equal sectors, each sixty bits long. The start of each sector is aligned with one

of the bits engraved on track **53**. Track **54** is sensed by a read amplifier whose output is used to provide the **40K** pulses and whose output is also used to generate the time gates **40C**. Track **56** forms a tach clock containing one prerecorded bit or fiducial mark Y, not shown, which is aligned with a bit in each of the tracks **53** to **55**. This track is sensed by a read amplifier to provide an R-bit pulse during each revolution of memory drum **31**.

Stack-forming synchronizer, first embodiment (FIG. 2)

Briefly, the function of the stack-forming synchronizer **43** is to control the formation of the stacks of letters on conveyor belt **41**. In performing this function, the synchronizer is responsive to ten to thirty active unload category codes selected in the manner set forth in the above patent application.

After being selected by the selector **40** in FIG. 1, an active unload category is transferred through or-gate **439** in FIG. 2 to write amplifier **440**. The code is written in serial form in retimer track **441**, is sensed by read amplifier **442** and is sent to write amplifier **443**. The active unload category code is then written on store track **444** with a one bit advance so that the trail bit of the code is in line with that category code, if any, recorded in tracks **223** for slot **1** of the twenty-four hundred slots **24***a*. (See FIG. 4).

It will be recalled that a letter category code and slot number code are generated for each letter deposited in a slot **24***a*. These codes are transferred from the information buffer section **30** (FIG. 1) through the transfer unit **33** to the information analog section **34**. Tracks **223** and **204** (FIG. 2) are located in the analog section. Hence, as explained in the above-identified application, the slot number code, generated when a letter is deposited in one of the slots **24***a*, is used in such a manner that the related letter category code will be passed through or-gates **221** to write amplifiers **222** and will be recorded in a position on tracks **223** adjacent the slot number code prerecorded on tracks **204**, that is identical to the generated number code. (See FIG. 4). Hence, each number and related category code are positioned, side-by-side, in successive sectors of memory drum **31** in the same sequence and proportioanl location as the corresponding letters stored in slots **24***a*.

It is noted here that the twenty-four hundred bits of tracks **204** are divided into forty sectors, each sixty bits long. Two of these sectors are assigned to each station **1** to **20**, and each sector contains in parallel form the slot numbers, one to sixty, of half the slots of a rack **27**. One of the "**40**" sectors is shown in FIG. 4.

A reference point on conveyor belt **41** represents the trailing edge of the space allocated to the formation of a stack **46** on the belt. It will be recalled that the time required for a point on the belt to traverse the width of one slot **24***a*, plus a fixed interval of time, is equal to the time required for one revolution of memory drum **31**. Accordingly, an active unload category code is written on track **441** when the reference point passes a nominal 0-slot and is read for the first time by read amplifier **447** when the reference point, minus one twenty-four hundredth of the space allocated to the formation of a stack, approaches the 1-slot of slots **24***a*.

The active unload category code on track **444** is read continuously into 20-bit shift register **448** by means of read amplifier **447**. With no more than thirty twenty-bit letter category codes, regularly spaced on this twenty-four hundred bit track, the blank gaps between the category codes is invariably more than twenty bits. Because it is not possible for the beginning and end of successive codes to be in register **448** at once, it is possible to indicate that a complete code is contained in the register by applying the lead and trail bits to and-gate **449**. An output from this and-gate indicates that a complete category code is present in the register.

Assume that a direct, active unload category is read out of track **444** for the first time and is recorded in register **448**. Simultaneously, read amplifiers **237** read the category code in tracks **223** that is releated to slot **1** of slots **24***a*. Character No. **1** in the output of the read amplifiers and in register **448** are compared in comparator **458**, while character Nos. **2** and **3** are compared in comparator **459**. When identities are found, the output of comparator **458** is applied to and-gate **460** and the output of comparator **459** is branched to and-gates **460**, **463**. Since a "1" bit is located at both ends of register **448**, theoutput of and-gate **449** enables and-gate **460** to pass a signal through or-gate **464** to **465**, which represents six and-gates. The code number of slot **1** is now read from tracks **204** by read amplifiers **466** and is passed, in parallel form, through and-gates **465** to the decoding matrix **44**. The matrix is then controlled, in a manner described in detail in the above patent application, to effect the discharge of the letter in slot **1** to the conveyor belt **41** in FIG. 1, thereby starting the formation of a stack of letters on the conveyor belt.

Summarizing the above operations, the active unload category code is read out of store track **444** and is positioned in register **448**. The related category code is read out of tracks **223** and is compared in comparators **458**, **459** with the unload category code. Since an identity is found in each comparator, the slot number in tracks **204**, on line with the category code, is applied to the decoding matrix **44**, which generates a gate-release signal. The latter signal is applied to the pertinent one of the slot-gate actuators **45** in FIG. 1, so that the letter contained in the associated slot **24***a* is discharged to start the formation of a stack **46** on conveyor belt **41**.

Returning to FIG. 2, and-gate **478** and inverter **479** form and-not-gate **480**. In the absence of an R-bit pulse on inverter **479** and the presence of a **40K** pulse on and-gate **478**, the latter and-gate provides an output signal that is applied to terminal **1** of flip-flop **481**, turning the latter an. The flip-flop remains on until turned off in the manner described below.

The output of flip-flop **481** enables and-gate **486** and the active unload category code under consideration passes bit-by-bit through register **448**, the latter and-gate and or-gate **439** to write amplifier **440**. The code is then written on track **441**, sensed by read amplifier **442** and advanced one bit on track **444** from the position in which it was previously written on this track. The trail bit of the active unload category code is now in line with the letter category code, if any, recorded in tracks **223** and the number code, recorded in tracks **204** for the letter in slot **2** of slots **24***a*.

It should be noted that after a category code is read and before the code is rewritten on either track **441** or **444**, the category code is erased by an erase head not shown in FIG. 2.

The operations, described in detail above, are now repeated with the category code written in tracks **223** for slot **2** of slots **24***a*. If identities are obtained in comparators **458** and **459** or in comparators **459** and **489**, as indicated below, the slot number code, read from tracks **204**, is transmitted to decoding matrix **44** to permit the letter in slot **2** to be discharged into the stack **46** forming on conveyor belt **41** in FIG. 1. The same sequence of operations are performed for the remaining slots **24***a*.

The conveyor belt advances the width of one slot **24***a* minus a small, fixed distance for each revolution of memory drum **31**, so that each letter added to a stack **46** in FIG. 1 drops to a location on the belt slightly in advance of the previous letter. Ideally, each letter of a stack will partially overlap the previous letter so as to present an orderly feathered arrangement.

The example presented in detail above relates to a direct category code. When a secondary category code, say *__NY, is selected through predominance as an active unload category code, only the characters NY are written on track **444**. When this category code is read out of

track **444** and is shifted through register **448**, character No. **1** in the register is blank and characters Nos. **2** and **3** are N and Y, respectively. If ___NY is written in tracks **223** on line with this active unload category code, identities are found in comparators **458, 459**. As set forth in the example above, the outputs of these comparators enable and gates **465** to pass the slot number, read out of tracks **204**, to the decoding matrix **44**. This will effect the discharge of the letter in the related slot **24** to the appropriate stack **46** forming on the conveyor belt **41**.

In contrast, when the secondary category code is selected as an active unload category code through a programed opportunity, the characters *NY are written on track **444** and the command code and characters N and Y are then recorded in register **448**. Since the command code in the register is compared in comparator **489** with the command code provided by the 6-bit command reference source **490**, each time that a letter category code ending in NY is encountered on tracks **223** in line with this active unload category code, the associated letter in one of the slots **24***a* is discharged to the stack **46** forming on conveyor belt **41**. Thus, the stack comprises letters in the secondary code ___NY and in the direct categories associated with the same. It should be noted here that the output of command reference source **490** and the command code in register **448**, when applied to comparator **489**, permits ignoring character No. **1** of the associated direct category codes written on tracks **223**.

In view of the foregoing, it will be apparent that each time one of the active unload categories is read out of track **444**, it is returned to that track with a one bit advance in angular position. In effect, the read amplifier **447** senses each active unload category during one revolution of memory drum **31** and on the succeeding revolution each code is advanced into alignment with the category code and slot number code for the next slot **24***a*. Hence, starting at slot **1** each active unload category code traverses track **444** and is sequentially compared with a letter category code for each slot **24***a*. When identities are found in either comparators **458** and **459** or comparators **459** and **489**, a signal is generated and applied to decoding matrix **44** to control the formation of a stack of letters, associated with that unload category, on conveyor belt **41**.

In keeping a running inventory of the categories of the letters in slots **24***a*, a letter is not counted if it is in an active unload category and is positioned in a slot in advance of the stack **46** forming on conveyor belt **41** for this category. The arrangement for accomplishing this result is in the tally accounting unit **38** (FIG. 1) and is responsive to character Nos. **1** to **3** in register **448** or the output of comparator **489** and characters Nos. **2** and **3** in the register. For further details of this arrangement, reference is made to the above-identified patent application.

The arrangement for erasing the letter category codes from track **223** will now be discussed. When a letter category code is written on tracks **223** in line with an identical direct unload category code, identities are obtained in comparators **458, 459**. The output of and-gate **460** is then directed through or-gate **464** to write amplifier **471**, is written on retimer track **472**, sensed by read amplifier **475** and converted to an erase signal by eraser **476**. The output of **476** is passed through or-gate **221** to the eighteen write amplifiers **222**, erasing that category code which had produced the identity signal when read out of these tracks by read amplifiers **237** and applied to the comparators.

When a category code is written on tracks **223** in line with an identical secondary active unload category code, derived through predominance, identities are obtained in comparators **458, 459** and the code on tracks **223** is erased in the manner just indicated. Finally, when relevant category code on tracks **223** is in line with the secondary active unload category code, derived through a programed opportunity, identities are found in comparators **459, 489** and the category code on tracks **223** is erased, again in the manner indicated above. Here, it should be noted that the associated direct category codes on tracks **223** are included for erasure.

The output of and-gate **449** coincides with the R-bit pulse in and-gate **477** only upon the full circumferential traverse of track **444** by an active unload category code. At this time, a signal is applied to terminal **0** of flip-flop **481**, turning the latter off, so that the information in register **448** will not pass through and-gate **486** to write amplifier **440**, but will pass through and-gate **483**, enabled by the inverter **485**, in and-not-gate **484** to the next machine. The next **40K** pulse passes through and-gate **478** to terminal **1** of flip-flop **481**, turning the latter on. The output of the flip-flop is then applied to inverter **485**, and-gate **483** is inhibited, while and-gate **486** is enabled to pass the next information in register **448** to write amplifier **440**. Thus, a time gate, sixty bits long, is provided during which the output of register **448** is directed to the next process machine instead of to write amplifier **440**.

As described in detail in the above patent application, each sorting station **1** to **20** contains a rack divided into rack-half X and rack-half Y. Since there are sixty slots **24***a* in each rack-half, there are, in total, twenty-four hundred slots. Each slot is provided with a normally-closed discharge gate **24***b* (FIG. 1) that is adapted to be opened by a release mechanism described in detail in U.S. Patent No. 2,863,574. The release mechanism for each gate is controlled by a solenoid connected to a respective one of twenty-four hundred lines in the output of decoding matrix **44**. The release mechanisms and solenoids for all the gates **24***b* are designated by the slot-gate actuators **45** in FIG. 1.

Each slot number code in tracks **204**, when related to an active unload category, is passed through and-gates **465**, in parallel form, to decoding matrix **44**. Time gates **40C** comprise forty components, generated in sequence. The active time gates applied to decoding matrix **44** together with the six bits of the number code cause a slot unload signal to be applied to one of the lines in the output of the matrix. Thus, when the code of a slot number is applied to the matrix, a signal appears on one of its twenty-four hundred output lines to energize a solenoid which operates a release mechanism to open a discharge gate **24***b*. A letter is then discharged by gravity from the associated slot **24***a* to conveyor belt **41** (FIG. 1). Subsequent deenergization of the solenoid operates a reset mechanism, described in U.S. Patent No. 2,863,574, to cause the discharge gate to close.

Stack-forming synchronizer, second embodiment (FIGS. 3A to 3D)

With reference to FIG. 3B, tracks **223** and **204** are located in the analog section **34** in FIG. 1.

As previously explained, a slot number code is generated whenever a letter is deposited in one of the slots **24***a*. This code is used to control the routing of a related letter category code, which is passed through or-gate **221** to write amplifiers **222** and is then written on tracks **223** efficaciously adjacent to a number code that is prerecorded in tracks **204**. The prerecorded code is identical to the generated number code. (See FIG. 4.) Thus, each number and related category code are positioned side-by-side on drum **31** in the same sequence as the corresponding letters stored in slots **24***a*.

As will be described later, flip-flop **169** (FIG. 3D) is normally on, being turned off only for a time interval of sixty bits duration after an active unload category code has traversed track **444**. (See "Fiducial Record, infra.) Hence, the flip-flop applies a signal to and-gate **262** which is normally enabled to pass the contents of register **138**. On the other hand, flip-flop **167** (FIG. 3D) is normally turned off unless identities are found in the second comparing-position of register **138**. Hence,

and-gate **251** is normally blocked from passing the contents of register **138**.

With reference to FIG. 3A, the next active unload category code, received from selector **40** in FIG. 1, is passed through or-gate **125**, 21-bit register **138**, normally enabled and-gate **262**, write amplifier **126** and is recorded on retimer track **127**. The category code is sensed by read amplifier **130** and is sent through or-gate **131** to write amplifier **132**. The code is then recorded on store track **444** with a one bit advance, is sensed by read amplifier **134**, passed through or-gate **125** and is re-registered in **138**.

As will be set forth in detail below, a fiducial record, indicating the zero letter position of a stack **46**, is kept on store track **279**. To keep this record, the next active unload category code is passed through or-gate **280** to the 20-bit register **300** (FIG. 3A). The code is shifted through the register bit-by-bit, is passed through normally enabled and-gate **305** to write amplifier **281**. The code is then written on retimer track **282**, sensed by read amplifier **283** and is written by write amplifier **284** on track **279** with a one bit advance. This active unload category code, sometimes referred to hereinafter as the duplicate unload category code, is read by read amplifier **286** and is passed through or-gate **280** to register **300**.

Here it is noted that track **279** contains 2400 bit places, each aligned with a bit place in track **444**.

The spacing of stacks of letters **46** on conveyor belt **41** (FIG. 1), and thus the spacing between active unload category codes, is adjusted by means of Rev. SS device **102** in FIG. 3 of the above-mentioned patent application. This spacing is adjusted so that when the next active unload category code is passed through or-gates **125** or **280** to register **138** or **300**, respectively, there will be no interference between this code and one read out of track **444** or **279**. (It will be apparent that if the active unload category codes do not interfere in registers **138** and **300**, they will not interfere when written on tracks **444** and **279**).

To avoid interference, registers **138** and **300** must be cleared of information for an interval of time between the recording of each category code therein. This is accomplished primarily by setting the pitch of stacks **46** by means of device **102** at a value greater than twice the bit length of the longest register, in this case, **138**.

However, this expedient does not take into account the need for a space on tracks **444** and **279** that is unoccupied at the time that the next active unload category code is recorded in registers **138** and **300**. If this space on either track were occupied, the code recorded therein after being read out, would interfere with the code being recorded in the associated register. Thus, at the time that the trail bit of the next active unload category code passes through or-gate **125**, the lead bit of the next code to be read out of track **444** must be at least twenty bits from the latter or-gate. A corresponding unoccupied space must be provided on track **279**.

When the O-bit and the 2400-bit position coincide on memory drum **31**, as in the present embodiment, the unoccupied space on tracks **444** and **279** may be obtained by the proper selection of the stack pitch. More specifically, a pitch may be chosen by trial and error such that when divided into the total number of bit positions in a track on drum **31**, a whole number is not obtained, but a number is obtained with a remainder which represents the unoccupied space on the tracks required for recording the next active unload category code in registers **138** and **300** without interference. (The last space on tracks **444** and **279**, occupied by the oldest category code before passing to the next process machine, is the same as that first occupied by the next active unload category code.)

As an example, using **30** as a practical number of forming stacks on conveyor belt **41**, the stack pitch, disregarding interference, would be 2400/30 or 80. To obtain a remainder, the pitch is selected to be 78 so that 2400 divided by 78 equals 30 stacks forming, plus 60 bit-places left over to avoid interference. Subtracting 42 (twice the bit-places of register **138**) from 60 leaves 18-bit-places, which allows for a count of 36 letters, per stack. To enable a larger letter count, the number of forming stacks may be adjusted downward.

It is obvious that a set of operative stack or active unload category code pitches may be predetermined for a fully specified machine.

In another arrangement to avoid interference, memory drum **31** may be dimensioned so that the O-slot and 2400-slot positions do not coincide. The positions may be separated by an otherwise unused part of the drum, representing unused or nonexistent slots **24***a*.

It will be recalled that active unload category code comprises eighteen category information bits which are preceded by a lead bit and followed by a trail bit. When the trail and lead bits are in the first and twentieth places, respectively, of register **138**, the trail bit is in line with a number code in tracks **204** and with a letter category code in tracks **223**. (See FIG. 4). This particular alignment of the trail bit is referred to hereinafter as the read point V.

The next active unload category code, received from selector **40** in FIG. 1, is initially branched to and-gate **143** (FIG. 3C) and to 19-bit delay device **144**. The output of **144** enables and-gate **143** to pass a single bit through or-gate **147**, 1-bit delay device **210** and normally enabled and-gates **209** and **211** to write amplifier **148**. The bit is then recorded on retimer track **149**, sensed by read amplifier **150** and sent through or-gate **151** to write amplifier **152**. The bit is recorded on store track **155** with a one bit advance in line with the above trail bit (see FIG. 4) and is referred to hereinafter as the odd-even mark O. The read point and odd-even mark are discussed in greater detail in connection with the event schedule in FIGS. 5A, 5B.

Since it is desired that the initial reading of the odd-even mark O and the trail bits of the active unload category codes be obtained simultaneously on read amplifiers **156**, **134** and **286**, the information is entered intially into delay device **210**, and or-gates **125**, **280**.

It will be apparent that there are several information advancing circuits in FIGS. 3A and 3B, that are exemplified by one that comprises retimer track **253**, store track **444**, the related read and write amplifiers and register **138**. These circuits are characterized by a mode of operation wherein the information is read out of the store track, is written on the retimer track and repositioned on the store track with a predetermined advance from the position where it was last read out.

Now consider the initial synchronizing requirements relative to the several information advancing circuits. The position of the heads of the read and write amplifiers of the various retimer tracks are adjusted to reposition the information as well as to compensate for the delays introduced in the circuits. Hence, because of the delay introduced by register **138**, retimer tracks **127** and **253** are compensated twenty bits besides the adjustments for their respective one and two bit advances.

In general, considering the bit pulse as a measure of time and one revolution of drum **31** as requiring 2400 bit pulses, a unit of coded information on a store track may be repositioned one bit by providing a retimer track and read, write and erasing means positioned so that the sum of the code's traverse of the advancing circuit is 2399 or 2401 bit pulses; the latter alternatives govern the direction of the traverse of the code on the store track relative to the rotation of memory drum **31**. As a more specific example, assume the direction of rotation of drum **31** is such that the bit places of store track **444**, starting with the 1-bit place, pass in sequence under the read head of amplifier **134**. Assume further that the trail bit of an active unload category is recorded in the 2-bit place of

store track **444**. Then, if the sum of the traverse of the advancing circuit is 2399 bit pulses, the trail bit will be rewritten in the 1-bit place of the track, while if the sum of traverse of the advancing circuit is 2401, the bit will be rewritten in the 3-bit place.

In this connection, it should be noted that each of the retimer tracks **282**, **127**, **253**, **149**, and **271** are provided with conventional erasing means, not shown, positioned after the heads of their read amplifiers and immediately before the heads of their write amplifiers, so that the information recorded on each track is erased after being read out.

The odd-even mark O indicates the count status of a stack of letters **46**, and the control logic, described below, uses this indication of status. An even count is indicated by coincidence of the odd-even mark with the output of and-gate **161** (FIG. 3A). Thus, an even-stack count is indicated by the output signal of and-gate **160** (FIG. 3C). When an output of and-gate **161** occurs without an odd-even mark reinforcement, it is understood that the count of the stack is odd.

At the start of an example of operation to be considered now, there are no letters in a stack **46**, and the related odd-even mark O and read point V are in line with the O-slot. (See FIG. 4). Read amplifier **156** (FIG. 3B) reads track **155** and applies the odd-even mark O recorded therein to and-gate **160** (FIG. 3C). Simultaneously, a "1" bit of an active unload category code is registered in the first and twentieth places of register **138**. That is to say, the active unload category code is registered in the first comparing-position of the register, and this enables and-gate **161** to apply a signal to and-gate **160**.

The output of and-gate **160** is transmitted to inverter **163**, which together with the six and-gates **164** and the twelve and-gates **165** comprise eighteen and-not-gates **166**. Since a signal is applied to inverter **163**, the code in register **138** can not pass through and-gates **164**, **165**.

The output of and-gate **160** is branched through or-gate **170** to terminal **1** of the odd-count addition flip-flop **171**, turning the flip-flop on. The output of the or-gate is also branched through 2-bit delay device **172** to terminal O of the flip-flop, so that after a two-bit delay the flip-flop is turned off. When activated, the output of the flip-flop is applied in parallel to **173** and **174**, illustrating six and twelve and-gates, respectively.

After the active unload category code in register **138** shifts one place, the trail bit of the code is in the second bit place, efficaciously in line with the next slot number code in tracks **204**, and the lead bit is in the twenty-first bit place. The active unload category code is now in the second comparing-position of register **138**, and coincidence of the "1" bits and-gate **178** enables the latter to transmit a signal to and-gates **173**, **174**. Character No. **1** of the category code in register **138** is now passed in parallel form through and-gates **173** and or-gates **180** to comparators **181** and **182**. Likewise, characters Nos. **2** and **3** in register **138** are passed in parallel form through and-gates **174**, or-gates **183** to comparator **185**.

Assume that the category code read out of track **223** by read amplifiers **175** is a direct category code and is the same as the active unload category code in register **138**. Then, identities will be obtained in comparators **182**, **185** and the output of these comparators will enable and-gate **186** to send a signal through or-gate **187** to and-gate **188** (FIG. 3D). On coincidence of the outputs of and-gate **178** and or-gate **187** in and-gate **188**, the latter transmits a signal to inverter **190** to inhibit the eighteen and-not gates **192**. Hence the slot number code, read out of tracks **204** by read amplifiers **176**, can not pass through the six and-gates **191** to decoding matrix **44**.

The output of or-gate **187** is branched to write amplifier **195** and is written on retimer track **196**. This track is sensed by read amplifier **197**, which applies its output to eraser **198** to provide an erase signal that is passed through or-gate **221** to write amplifiers **222**, thereby erasing the category code that was identified by comparators **182** and **185** in the manner described above.

If the active unload category code in register **138** had been for a program-derived secondary category, the command code in the register would have been compared in comparator **181** with the command code provided by 6-bit command reference source **230**. Similarly, character Nos. **2** and **3** in **138** would have been compared in comparator **185** with the characters read from track **223**. On the occurrence of identities, the outputs of these comparators would have enabled and-gate **231** to transmit an identity signal through or-gate **187** to and-gate **188**. Again, the output of or-gate **187** is branched to write amplifier **195** and written on retimer track **196**.

It will be apparent that the cooperation among register **138**, comparators **181**, **182**, **185**, source **230** and track **223** in the second embodiment is essentially the same as the cooperation among register **448**, comparators **458**, **459**, **489**, source **490** and tracks **223** in the first embodiment (FIG. 2). Accordingly, for other modes of operation of the first group of components just listed, attention is directed to the corresponding components in the first embodiment.

The output of comparator **181** and the signals applied through or-gates **180**, **183** are transmitted to the tally accounting unit **38** (FIG. 1) and inhibit the advance of a counter contained therein. This is accomplished in the same way as the output of comparator **489** and the inputs to each comparator **458**, **459** derived from register **448** in the first embodiment of the stack-forming synchronizer presented above.

The read heads of amplifiers **220**, **215**, **197** and the write heads of amplifiers **222** are in timing alignment and 180° away from the timing aligned read heads for amplifiers **176**, **175**, **156** and the write head for amplifier **195**. An erase head is positioned, but not shown, near each retimer track **282**, **149** and **271** and store track **155** immediately before the write head of each write amplifier associated with the track.

It will be recalled that at the start of the present example the read point and the odd-even mark O were in line with the O-slot. The latter mark was read out of track **155** by read amplifier **156** and was erased just before reaching the write amplifier **152**. However, since identities were not found when the active unload category code was in the first comparing-position of register **138**, the odd-even mark is re-recorded by means of write amplifier **152** with a one bit advance on track **155** in the fashion described in detail below. The odd-even mark is, in effect, recorded one bit after the read point. Since identities were found when the unload category code was in the second comparing-position of register **138**, write amplifier **195** records the output of or-gate **187** on track **196** as a bit, occurring efficaciously one bit after the read point. Thus, read amplifier **215** reads the odd-even mark O on track **155** at the same time that read amplifier **197** reads the latter bit on track **196**, and there is coincidence of the outputs of the amplifiers in and-gate **216**. The output of and-gate **216** enables and-gates **217**, and a slot number code is read by read amplifiers **220** from tracks **204**. This number is the one in line with the letter category code under consideration, since both codes moved the same 180° between readings. The output of read amplifiers **220** is passed through and-gates **217** and stored in parallel form in 6-bit register **218**.

In the absence of a signal to inverter **226** in and-not gates **227**, the output of register **218** is passed through the six and-gates **228**, in parallel form, to decoding matrix **44**. As set forth in the patent application mentioned above and summarized in connection with the first embodiment, the matrix generates a slot unload signal that is used to release the letter from the slot **24***a* (FIG. 1) assigned the slot number code just read out of tracks **204**. This letter is the first in the stack, is discharged

to a half-slot place on conveyor belt **41** and comprises an odd-count addition.

The outputs of and-gates **228** are branched to or-gate **231** which will pass any signal present in register **218** and thereby clear the register.

Normally the output of register **218** is not inhibited by and-gates **228**. However, an even count addition may occur for another stack at the same time that a slot number code is contained in register **218**, and its identity signal passed through or-gate **187** to and-gate **232**. Since a "1" bit is in the first and twentieth bit place of register **138**, the latter and-gate is enabled by the concurrent output of and-gate **161**, to apply a signal of one bit duration to inverter **226**. And-gates **228** will then be inhibited for the time interval of a single bit. This inhibition will seldom coincide with the code present in register **218**, since such coincidence requires that identities be found at the exact angular offset of the two sets of aligned heads mentioned above, and on occurrence will produce only a minuscule positioning error of the related letter on its stack **46**.

From the example just considered, it is apparent that when an active unload category code is in the first comparing-position in register **138** and an odd-even mark O and read point V are in line, the letter-category code for the next slot **24***a* is compared with an active unload category code in the second comparing-position in the register. When identities are found in the second comparing-position, a letter, comprising an odd-count addition, is released from a slot **24***a* after memory drum **31** has made one-half revolution from the point at which the identities were found. The letter occupies a half-slot place in a related stack **46**.

As another example of operation, assume that stack **46** has an odd-count, the read point V and the odd-even mark O are not in line, and the active unload category code is in the first comparing-position in register **138**. Since the read point and odd-even mark are not in line, and-gate **160** is blocked and the output of and-gate **161** can not pass through the former and-gate to inverter **163**. Thus, character No. **1** in the register will pass through and-gates **164** and or-gates **180** to comparators **181** and **182**, and character Nos. **2** and **3** will pass through and-gates **165** and or-gates **183** to comparator **185**.

Assume further that the slot on line with the read point V contains a letter in the category designated by the active unload category code. Identities will be obtained in comparators **181** and **185** or **182** and **185**, and a signal will be applied through or-gate **187** to and-gates **191**. The latter will then pass the slot number code, read out of tracks **204**, to the decoding matrix **44**. A letter will be released from the slot, associated with the latter code, to form an even-count addition, so that the count of the stack is now even.

When the output of and-gate **161** and the output of or-gate **187** coincide in and-gate **232**, the latter transmits a signal through or-gate **170** to terminal **1** of flip-flop **171**. Thus, flip-flop **171** may be set either by the finding of an even-count addition to an odd-count stack, as just indicated, or by the output of and-gate **160**, representing an even stack count.

After the active unload category code is shifted one bit place to the second comparing-position in register **138** and flip-flop **171** is turned on, character No. **1** in the register is passed through and-gates **173** to comparators **181** and **182**, while character Nos. **2** and **3** are passed through and-gates **174** to comparator **185**. The letter category code for the next slot **24***a*, i.e., the one following the slot just checked, is read out of tracks **223** and applied to the comparators. If identities are found, the operation becomes the same as the first example above, and the letter in the associated slot is released to the stack, forming an odd-count addition. The count of the stack is now odd.

If identities had not been found in the first comparing-position in register **138**, the lack of output of or-gate **187** would not have enabled and-gate **232** to turn on flip-flop **171**. Consequently, a comparison could not have been made between the unload category code in the second comparing-position and the letter category code, if any, for the next slot **24***a*, read out of tracks **223**.

From the last example, it is seen that when an active unload category code is in the first comparing-position in register **138** at a time when this read point V is not in line with the odd-even mark O, and the letter category code for the related slot is the same as the active unload category code then, and only then, will the letter category code for the following slot be checked at the next bit pulse, rather than after the next revolution of memory drum **31**.

To summarize the description presented so far: each active unload category code is read, in turn, out of store track **444** and is moved bit-by-bit through register **138**, where the code may be compared at a first and then at a second comparing-position with first one letter category code and then another read out of tracks **223**. During the first comparison, the first place of **138** contains the code's trail bit at the same time that a slot number code on tracks **204** is sensed by read amplifiers **176**. During the second comparison, one bit later, after the code has shifted one place in register **138**, the next advanced slot number code in tracks **204** is sensed. The number and category codes sensed when the unload category code is in the first comparing-position are related to a particular one of the slots **24***a*. The codes sensed when the unload code is in the second comparing-position are related to the next advanced slot **24***a*.

Accordingly, during one revolution of memory drum **31**, each active unload category code, written on track **444**, may be compared separately with the letter category code of first one slot **24***a* and then the next advanced slot. These two comparisons are separated by a time delay of one bit pulse.

If the two slots were unloaded on the occurrence of their related comparisons, practically simultaneously, the two letters would be separated on conveyor belt **41** by substantially a whole slot place. For this reason, only the first comparison is permitted to produce a slot unload signal in decoding matrix **44** at the point of comparison. When identities are found during the second comparison, the slot unload signal is produced after a delay of half-a-revolution of memory drum **31**, when conveyor belt **41** has moved half-a-slot space toward the slot **24***a* containing the pertinent letter. Thus, the even-count additions fill the whole-slot places and the odd-count additions fill the half-slot places in a stack **46**.

Because each letter is dropped to build on the leading edge of a stack as it forms on conveyor belt **41**, the active unload category code must be properly advanced on track **444** to pace the build-up. Hence when an odd-count addition is made to a stack, the code is advanced two bit places on track **444**. This means that when the stack has an odd count, the code is, in effect, recorded on track **444** at half-a-slot advance from the leading edge of the stack.

When an even-count addition is made to a stack, the active unload category code on track **444** is advanced the regular one bit place to match the constant advance on conveyor belt **41**. Thus, when the stack has an even count, the codes efficaciously recorded on track **444** in line with the leading edge of the stack.

When an addition is not made to a stack **46**, because identities are not found in the first and second comparing-position in register **138**, the active unload category code is advanced one bit place on track **444**.

An odd-even count record is kept on store track **155** by an odd-even mark O associated with the active unload category code for a stack **46**. This mark records an even count when in line with its unload category code and records an odd count when it is one bit place behind its

unload category code. For this record, when an even-count addition is made to a stack **46**, the odd-even mark O is advanced two bits during the following revolution of memory drum **31**; otherwise, the mark is advanced one bit place on track **155** for each revolution.

The arrangement for achieving these operations will be described in detail just below.

Assume that a stack **46** has an even count and that an active unload category code is recorded in the second comparing-position of register **138**. Assume further that flip-flop **169** is turned on, flip-flop **167** is turned off, and identities are found either in comparators **181** or **182** and comparator **185**, so that an odd-count addition is made to the stack.

Then, flip-flop **169** applies a signal to and-gate **251**. The output of or-gate **187** and the output of and-gate **178** enable and-gate **188** to send a signal to terminal **1** of flip-flop **167**, turning the latter on. The output of flip-flop **167** is branched to inverter **260** in and-not gate **261**, thereby inhibiting the and-not gate. The output of flip-flop **167** is also branched to and-gate **251**, which is then enabled to pass the active category code in register **138** to write amplifier **252**. The code is written on retimer track **253**, sensed by read amplifier **254** and passed through or-gate **131** to write amplifier **132**. The code is then recorded on store track **444** with a two bit advance.

Thus, it is seen that when identities are found in the second comparing-position of register **138** an odd-count addition is made to stack **46**, and the active unload category code is advanced two bit places on track **444** during the revolution of memory drum **31** that occurs after the comparison. Since the unload category code is advanced two bit places, the read point V is advanced the equivalent of two bit places.

Assume that a stack **46** has an odd count and that an active unload category code is recorded in the first comparing-position of register **138**. And-gate **161** will then apply a signal to terminal **0** of flip-flop **167**, turning the latter off. And-gate **251** is blocked, while and-gate **262** is enabled. Assume further that flip-flop **169** is turned on, and identities are found in either comparators **181** or **182** and comparator **185**. Under these conditions, an even-count addition is made to the stack **46**.

The active unload category code is then shifted one bit position to the second comparing-position of register **138** where a comparison is permitted (by action of flip-flop **171**, as described earlier) and where, it is assumed further, no identity is found. The unload category code is then shifted bit-by-bit through the remainder of register **138** and through and-gate **262** to write amplifier **126**. The code is recorded on retimer track **127**, is sensed by read amplifier **130** and is re-recorded on store track **444** with a one bit advance.

Hence, when identities are found in the first comparing-position of register **138**, an even-count addition is made to a stack **46**, and the active unload category code is advanced one bit place on track **444** during the revolution of memory drum **31** that occurs after the comparison. Since the latter point is advanced one bit place, the read point V is likewise advanced the equivalent of one bit place. (This example assumes, of course, that identities will not be found in the second comparing-position in register **138**. If the latter occurs, the unload category code is advanced two bit places on track **444**, as set forth above.)

In addition when identities are not found in either the first or second comparing-positions of register **138**, flip-flop **167** is not turned on, and-gate **251** is blocked, but and-gate **262** is enabled to pass the active unload category code. The code is then re-recorded on track **444** with a one bit advance.

With reference to the odd-even mark, consider the following example. The active unload category code is recorded in the first comparing-position of register **138**. The odd-even mark and the read point V are not in line, so that the output signal of and-gate **161** does not coincide with the odd-even mark in and-gate **160**, indicating that the count of the related stack **46** is odd. Identities are found in comparators **181** and **185** or **182** and **185**, so that an even-count addition is made to the stack. Under these conditions, the read point V (signaled from and-gate **161**) and the output of or-gate **187** coincide in and-gate **232**. The output of the latter and-gate is applied to inverter **213**, blocking and-gate **211**. The output is also applied to and-gate **268** to enable the latter to pass the odd-even mark to write amplifier **269**. The mark is then written on track **271**, is sensed by read amplifier **272**, passed through or-gate **151** and is rewritten or store track **155** with a two bit advance.

Thus when the odd-even mark O and read point V are not in line, an odd count of the related stack **46** is indicated, and when identities are found in the first comparing-position of register **138**, the odd-even mark is advanced two bit places on store track **155** during the following revolution of memory drum **31**.

It will be apparent that when the odd-even mark and the read point are not in line and identities are not found in the first comparing-position of register **138**, the odd-even mark, read out of track **155**, will pass through and-gate **211**, instead of **268**, and will be advanced one bit place on track **155** during the following revolution of memory drum **31**. Likewise, when the read point V and odd-even mark O are in line (denoting an even count of stack **46**), the output of and-gate **160** will block and-gates **164**, **165**, so that no identities can be found in comparators **181**, **182** and **185**, and the odd-even mark will be advanced one bit place on track **155** during the next revolution of memory drum **31**.

At this point, it is of interest to consider the function of delay device **210**. With reference to FIGS. 5A and 5B, it is noted that the tracks of drum **31** traverse any stationary point, such as a read head, from right to left and that the odd-even mark O is either in line with or one slot to the left of the read point V. In the latter case, the odd-even mark will be read out of track **155** one bit ahead of the point at which the trail bit of the unload category code relative to read point V is read out of track **444**. When the two marks are not in line and identities are found in comparators **181** or **182** and **185**, it is desired that the output of and-gate **232** and the odd-even mark coincide in and-gate **268** to permit the odd-even mark to advance two bit places, as described above. To achieve this result, the delay device **210** provides the odd-even mark with the necessary one bit delay.

In summary, when the stack count is even and an identity is found with the active unload category code in the second comparing-position in register **138**, the odd-even mark O is advanced one bit place on store track **155**. When the stack count is odd and identities are not found with the active unload category code in either the first or second comparing-position in register **138**, the odd-even mark O is advanced one bit place on track **155**. Furthermore, only when the stack count is odd and an even-count addition is found at its only designated opportunity, read point V, then and only then, is the odd-even mark O provided with a two bit advance.

Fiducial record (FIGS. 3A and 3D)

A fiducial record, that indicates the zero letter position of a stack **46**, is kept on store track **279** and is passed to subsequent stack-processing stations.

As set forth earlier, a duplicate of each active unload category code is entered on track **279** at the time that the code is entered on store track **444**. During each revolution of drum **31**, the duplicate code is read out of track **279**, shifted through register **300** and through and-gate **305** to write amplifier **281**. The code is written on track **282**, sensed by read amplifier **283** and is rewritten on track **279** with a one bit advance.

It will be recalled that the time required for a reference point on conveyor belt **41** to traverse one slot **24***a* is essentially equal to the time required for one revolution of memory drum **31**. Since there are twenty-four hundred slots and the same number of bit places on track **279**, one bit advance per revolution provides the duplicate code with a uniform traverse of track **279**. This represents a uniform traverse of slots **24***a* by a reference point on conveyor belt **41**.

At the end of traverse of track **444** by an unload category code, the "1" bits, recorded in the first and twentieth places of register **138**, enable and-gate **161**, whose output coincides with an R-pulse in and-gate **168** (FIG. 3D). The output of the latter and-gate then turns off flip-flop **169** to block and-gates **251** and **262**. The flip-flop is turned on by the next **40K** pulse, passing through and-gate **202**. In this way, and-gates **251**, **262** are blocked for a period of sixty bits long, which is adequate to prevent the code in register **138** from being re-recorded on tracks **253** or **127**.

At the end of traverse of track **279** by the duplicate code, the "1" bits, recorded in the first and twentieth places of register **300**, coincide with the last-mentioned R-pulse in and-gate **301**, which is enabled to turn on flip-flop **302**. The output of the flip-flop is applied to inverter **303** in and-not gate **304** to block and-gate **305**, so that the duplicate code can not be rewritten on track **282**. At the same time, the output of **300** is branched to and-gate **285**, permitting transfer of the duplicate code, whose trail bit is the fiducial mark, to the next process station.

Count of letters in a stack (FIGS. 3A and 3D)

As previously described, an active unload category code and its duplicate are recorded simultaneously on tracks **444** and **279**, respectively. The duplicate represents the zero letter position of a related stack on conveyor belt **41**.

The letters in an unload category are discharged from slots **24***a* to the belt at successive half-slot places, advancing head of the zero letter position in a stack. To pace the build-up of letters, during each revolution of memory drum **31**, the unload category code is advanced either one or two bit places on track **444**. The duplicate code, however, is advanced only one bit place on track **279** per revolution, and remains at the zero letter position. Accordingly, the trail bit of the unload category code on track **444**, within half-a-slot, indicates the leading edge of a stack **46**, while the trail bit of the duplicate code on track **279** indicates the zero letter position of the stack.

Since a letter is dropped in each whole slot and half-slot place, the count of the letters in a stack is equal to twice the number of bit places that separate the trail bit of the code on track **444** and the trail bit of the related code on **279**, plus nothing for an even-count stack, or minus one for an odd-count stack. The number of letters in a stack **46** may, therefore, be obtained by counting the revolutions that occur between the end of the traverse of the category code on track **444** and the end of traverse of the code on track **279**; letting each revolution equal two; and using the odd-even information, obtained from track **155**, to let the first revolution equal one (instead of two) when the count of the stack is odd.

The operations performed to take this count will now be considered in detail.

When an active unload category code has completed a traverse of track **444**, an R-pulse and output of and-gate **161** coincide in and-gate **168**, which turns on flip-flop **310** (FIG. 3D). The output of this flip-flop, indicating the start of a count, is applied to and-gate **311** in and-not gate **312**.

Now when the stack has an odd count, the revolution of memory drum **31**, occurring when flip-flop **310** is turned on, is counted as one, instead of two. More specifically, an odd count of stack **46** is deduced from the absence of an even count, that is, when and-gate **160** fails to apply an output signal to inverter **314** (FIG. 3D) in and-not gate **315**. The output of and-gate **168** is then passed through and-gate **316** to inverter **317** to inhibit and-gate **311** in and-not gate **312**. Consequently, the R-pulse, generated at the end of traverse of track **444** by the unload category code, is not transmitted through and-gate **311** to advance binary counter **313**. Instead, the output of and-gate **316** is branched to terminal **1** of flip-flop **318**, which then applies a signal to one of nine and-gates **320**.

Since flip-flop **310** is turned on, each succeeding R-pulse, representing a revolution of memory drum **31**, is passed through and-gate **311** to binary counter **313**. Each time a pulse is applied to the counter, the value of the binary number recorded therein is advanced by two until the output of and-gate **301**, which signals the end of traverse of the duplicate unload category code on track **279**, is applied to terminal **0** of flip-flop **310**, turning the latter of. Subsequent R-pulses can not pass through and-gate **311** to advance binary counter **313**, whose parallel outputs, together with the output of flip-flop **318**, are applied to and-gates **320** to indicate the count of the number of letters in the completed stack **46**.

Flip-flop **318** is turned off if, at the end of traverse of track **444** by another active unload category code, the outputs of and-gates **168**, **160** coincide in and-gate **319** and the output of and-gate **160**, also applied to inverter **314**, inhibits and-gate **316**. In this way, a signal is applied to terminal **0**, and not to terminal **1**, of flip-flop **318**, turning the latter off, to register an even count.

Moreover, when the stack has an even count, the inhibited and-gate **316** applies no output to inverter **317** of and-not gate **312**. The normal output of inverter **317** and the output of flip-flop **310**, turned on as described above, enable and-gate **311** to pass the R-pulse, coincident with the end of traverse, to binary counter **313**. During each succeeding revolution of memory drum **31**, an R-pulse is generated and applied to the counter. Each pulse, applied to the counter, advances by two the value of the binary number recorded therein, until the output of and-gate **301** is applied to terminal **0** of flip-flop **310**, turning the same off. Subsequent R-pulses can not pass through and-gate **311** to advance the counter, which with flip-flop **318**, then indicates the number of letters in the completed stack **46**.

On the occurrence of the output of and-gate **301**, which is branched in parallel to and-gates **320**, the output of flip-flop **318** and the binary number in counter **313** are passed through the and-gates to 9-bit register **308**. When, as noted above, and-gate **285** is enabled by the output of flip-flop **302**, the same output enables and-gate **307** through which the information in register **308** is then transmitted in serial form to the next process machine, not shown, which may be an automatic labeller and bundler, or to a statistical accounting machine.

In this way, the active unload category code and count are synchronous with the end of the related stack of letters. The latter information is transferred to the next stack-processing machine at the time of exit of the end of the stack from the present sorting machine. More precisely, the zero letter of a completed stack and the start of transfer of the information are coincident with a position equal to one slot after the last slot of the present machine.

Operations using event schedule (FIGS. 5A and 5B)

As described earlier, the read point V represents the presence of a trail bit in the first bit place of register **138** when the latter contains an active unload category code in its first comparing-position. The event schedule in FIGS. 5A and 5B illustrates a progressive accounting record for a sample group of successive read points V of a single active unload category code. For the purpose of the following discussion, the "advance" of read point V is entirely dependent upon the physical advance (re-

positioning on track **444**) of an active unload category code with its accompanying trail bit.

Some of the rules of operation, illustrated above, are summarized as follows:

(1) The fiducial mark X is advanced on track **279** one bit position, the analog of one slot **24***a*, per revolution of memory drum **31**.

(2) During each revolution of memory drum **31**, both the odd-even mark O and the read point V are advanced one bit position, the analog of slot **24***a*, except: (a) When an identity is found in the first comparing-position of register **138**, the odd-even mark is advanced two bit positions; (b) When an identity is found in the second comparing-position of register **138**, the read point V is advanced two bit positions.

(3) When an active unload category code is in the first comparing-position in register **138** and the associated odd-even mark O and read point V are in line, the letter category code for the slot in line with the read point is not checked, but the letter category code for the slot ahead of the latter point is checked when the category code shifts to the second comparing-position of the register. (A letter category code is "checked" when it is compared with the active unload category code.)

(4) (a) When an active unload category code is in the first comparing-position in register **138** and the associated read point V is not in line with the odd-even mark O, the letter category code for the related slot aligned with this read point is checked; (b) if this letter category code identifies with the active unload category code, then in addition, the letter category code for the following slot will be checked in the second comparing-position.

(5) an even-count addition to a stack **46** is made at the point in the revolution of drum **31** where the related identity is found.

(6) An odd-count addition to a stack **46** is made at 180° of the revolution of drum **31** after the point where the related identity is found.

A typical example of operation will now be presented with the assistance of the above rules and the timing schedule illustrated in FIGS. 5A, 5B. At the start of operation, the letters of the pertinent active unload category code, shown dotted in FIG. 5B, are positioned in the slots **24***a*, selected at random, in the figure. Again, at the start of operation, the associated fiducial mark X, odd-even mark O and trail bit in tracks **279**, **155** and **144**, respectively, are in line at the read point V. The active unload category in this example is chosen to be the direct category **RNY**.

It is noted that in FIG. 5B the length of each letter slot **24***a* is illustrated as equal to the length of a bit place on memory drum **31**, so that the sequence of the slot numbers pre-recorded on tracks **204** can be aligned with a matching sequence of progressively designated letter slots. In this figurative relationship, with conveyor belt **41** shown traversing the stationary letter slots **24***a* from left to right, the drum traverses any structurally associated fixed point, such as a read head, from right to left. In the depiction of the progress of read point V, the odd-even mark O is recorded either on line or one bit place to the left.

At the extent of progress depicted in FIGS. 5A, 5B, the letters represented as dotted lines have been discharged to conveyor belt **41**. Each letter category code RNY in tracks **223**, associated with a discharged letter, has been erased from the tracks and is shown in thin black lines. The odd-even mark, unload category code and duplicate unload category code are written on tracks **155**, **444** and **279**, respectively, in the manner shown in FIG. 5B.

Starting the example at slot O, since the read point V and the odd-even mark O are in line, comparison is blocked at the first comparing-position of register **138**, and the letter category code for slot **1** is compared with the active unload category code RNY when the latter is in the second comparing-position of the register (Rule 3). Since this slot does not contain a letter in the unload category, identities are not found and the odd-even mark and read point are advanced one bit position or one slot (Rule 2).

The fiducial mark is also advanced one slot (Rule 1). In fact, the fiducial mark X is advanced one slot during each revolution of memory drum **31**. This is indicated in FIG. 5B and will be understood in the discussion below.

After revolution 1, since the read point V and the odd-even mark O are still in line, but now at slot **1** where a comparison is now blocked, the letter category code for slot **2** is compared with the active unload category code RNY in accord with Rule 3. (More precisely, the comparison, in this case, blocked at one revolution plus one bit occurs at one revolution plus two bits.) Because slot **2** contains a letter in this category, an identity is found in the second comparing-position of register **138**, and the first letter in stack **46**, an odd-count addition, is discharged to conveyor belt **41** during revolution **2** (Rule 6). The odd-even mark O is advanced one slot and the read point V is advanced two slots (Rules 2 and 2*b*).

After revolution 2, a comparison is permitted at the read point, now in line with slot **3**, for the reason that the odd-even mark O and the read point are not in line (Rule 4*a*). Slot **3** does not hold a letter in the unload category code; therefore, an identity is not found in the first comparing-position. The read point and odd-even mark are then advanced one slot (Rule 2).

After revolutions 3 and 4, slots **4** and **5**, respectively, are found empty; and during each following revolution, the read point and odd-even mark are advanced one slot.

After revolution 5, the read point V is in line with slot **6**; the odd-even mark O is still not in line at this point, so a comparison is permitted (Rule 4*a*). Since slot **6** contains a letter in the unload category, an identity is found in the first comparing-position in register **138**. The letter category code for slot **7** is then compared with the unload category code in the second comparing-position (Rule 4*b*). Identities are found in each comparing-position, and consequently, the letter in slot **6**, an even-count addition, is discharged to the stack **46** immediately, while the letter in slot **7**, an odd-count addition, is discharged during revolution 6 (Rules 5, 6), and stack **46** then contains three letters. The odd-even mark O is advanced two slots (Rule 2*a*); the read point V is advanced two slots (Rule 2*b*) during revolution 6.

After each of the revolutions 6 to 12, a respective one of the slots **8** to **14** is checked; and since each slot is empty, the read point and odd-even mark are advanced one slot per revolution. This is in accord with Rule 2.

After revolution **13**, the read point V is in line with slot **15**; and the odd-even mark O is not in line with the read point. In accord with Rule 4*a*, the category code for the letter in slot **15** is compared in the first comparing-position of register **138** with the unload category code. (This comparison is made at revolution **13** plus fifteen bits.) An identity is found, and consequently, the letter category code for slot **16** is compared with the unload category code in the second comparing-position of the register (Rule 4*b*). Because identities are found in the first and second comparing-positions, the letter in slot **15**, an even-count addition, is discharged at the read point (Rule 5), while the letter in slot **16**, an odd-count addition, is discharged half-a-revolution of drum **31** thereafter (Rule 6). Because identities are found in the first and second comparing positions, the odd-even mark O is advanced two slots (Rule 2*a*) and the read point V is advanced two slots (Rule 2*b*).

After revolution **14**, the read point V is in line with slot **17**, while the odd-even mark O is not. Thus, slot **17** is checked, and since it contains a letter in the category code, slot **18** is also checked (Rules 4*a*, 4*b*). Since the latter slot is empty, the read point V is advanced one slot and since a letter is identified in slot **17**, the odd-even

mark O is advanced two slots (Rules 2, 2*a*). The letter in slot **17**, an even-count addition, is discharged at the former read point (Rule 5) and the stack contains six letters.

After revolution **15**, the odd-even mark O and the read point V are in line; consequently slot **18** is skipped and **19** is checked (Rule 3). The latter slot is found empty and the mark and point are each advanced one slot (Rule 2).

After revolution **16**, the odd-even mark and read point are still in line at slot **19** which is skipped, but slot **20** is checked (Rule 3) and contains a letter in the unload category. The read point is therefore advanced two slots, and the odd-even mark, one slot (Rules 2*b*, 2). The letter in slot **20** is an odd-count addition and is released to the stack half-a-revolution of drum **31** after the point at which the identities are found. The stack comprises seven letters and the odd-even mark, having been set out-of-line with the read point, reflects the odd condition.

After each of the revolutions **17**, **18**, and **19**, a respective one of the slots **21**, **22** and **23** is checked; and because each slot is found empty, the odd-even mark and read point are each advanced one slot (Rule 2). At twenty-four bit places after revolution **20**, slot **24** is checked at the read point. Since the unload category code is identified, the letter in slot **24** is released at this point as an even-count addition to the stack, and the letter category code for slot **25** is checked in the second comparing-position.

What is claimed is:

1. A stack-forming synchronizer for articles, each associated with one of a plurality of article category codes and each positioned in a respective one of a plurality of slots arranged in a sequence, comprising:

first storing means containing said article category codes, each in a code position related to a respective one of said slots,

second storing means having a plurality of code positions, each related to a respective one of said slots,

means for recording at least one unload category code in one of the code positions in said second storing means,

first comparing means having a first and second comparing-position,

means for transferring said unload category code from said second storing means to the first comparing-position and then to the second comparing-position, the first comparing-position being associated with the slot related to the code position from which the unload category code was transferred and the second comparing-position being associated with the next slot occurring in said sequence of slots,

second comparing means,

first control means for selectively applying either the output of one of the comparing-positions or for applying the output of the first and second comparing-position in sequence to said second comparing means,

means for sequentially transferring to said second comparing means the article category codes in said first storing means related to the slots associated with the first and second comparing-position, respectively,

means for generating an output signal when an identity is found in the second comparing means,

means responsive to each output signal for releasing the article in the slot related to the article category code being applied to said second comparing means when the output signal is generated, and

second control means for re-recording the unload category code in said second storing means with a selected advance in code position.

2. The stack-forming synchronizer set forth in claim **2** wherein said first control means comprises:

means for generating an output signal dependent upon whether the count of the articles released from the slots is odd or even, and

means responsive to said output signal for selectively applying either the output of one of the comparing-positions or for applying the output of the first and second comparing-position in sequence to said second comparing means.

3. The stack-forming synchronizer set forth in claim **2** wherein said first control means comprises:

signal generating means for providing a count signal dependent upon whether the count of articles released from the slots is odd or even,

means responsive to the coincidence of said count signal and the presence of an unload category code in said first comparing-position for generating a first gating signal,

means responsive to the coincidence of an unload category code in said first comparing-position and an output signal of said second comparing means for providing a second gating signal,

means responsive to the absence of said first gating signal for applying the unload category code in said first comparing-position to the second comparing means, and

means responsive to either the first or second gating signal for applying the unload category code in the second comparing-position to the second comparing means.

4. The stack-forming synchronizer set forth in claim **3** wherein said signal generating means comprises:

count-storage means having a plurality of code positions arranged in a desired sequence,

means for generating an odd-even signal and for recording the same in one of said code positions,

means responsive to an output signal generated by said second comparing means and the presence of the unload category code in said first comparing-position for advancing said odd-even signal a selected number of code positions in said sequence,

means responsive to the absence of an output signal generated by said second comparing means when said unload category code is in the first comparing-position for advancing the odd-even signal at least one code position in said sequence,

means for reading said odd-even signal out of said count-storage means, and

means responsive to said odd-even signal for generating a count signal.

5. The stack-forming synchronizer set forth in claim **4** wherein said second control means comprises:

means responsive to an output signal generated by said second comparing means and the presence of the unload category code in said second comparing-position for advancing the latter code a predetermined number of code positions in said second storing means, and

means responsive to the absence of an output signal generated by said second comparing means when the unload category code is in the second comparing-position for advancing the latter code at least one code position in said second storing means.

6. The stack-forming synchronizer set forth in claim **1** including:

a conveyor belt positioned beneath said slots,

means for moving said belt at a speed such that in a predetermined time interval a reference point on the belt is advanced substantially the width of one of said slots,

means responsive to the output signal generated by said second comparing means when the unload category code is in one of the comparing-positions for discharging at substantially the occurrence of the output signal the article from the slot related to the article category code being applied to the second comparing means, and

means responsive to the output signal generated by the second comparing means when the unload category

code is in the other comparing-position for discharging the article from the slot related to the category code applied to said second comparing means, the time interval between the latter discharge and the associated output signal being equal to that required for said reference point to travel a portion of the width of one slot.

7. The stack-forming synchronizer set forth in claim 1 including:

third storing means having a plurality of code positions, each related to a respective one of said slots,

means for recording a duplicate unload category code in a position in said third storing means when the unload category code is recorded in a position in said second storing means, the latter positions in the second and third storing means being related to the same slot,

means for advancing the duplicate code at least one code position in said third storing means each time that the unload category code is recorded in said first comparing means, and

counting means for providing a signal dependent upon the number of positions occupied in said third storing means before the duplicate code completes the traverse of the third storing means after the unload category code has traversed all the positions in said second storing means.

8. The stack-forming synchronizer set forth in claim 1 including:

third storing means having a plurality of code positions, each related to a respective one of said slots,

means for recording a duplicate unload category code in a code position in the third storing means when the unload category code is recorded in a code position in said second storing means, the latter code positions in the second and third storing means being related to the same slot,

means for advancing the duplicate code in at least one code position in said third storing means each time the unload category code is recorded in said first comparing means,

signal generating means for providing a count signal dependent upon whether the count of articles released from said slots is odd or even, and

counting means for providing a signal dependent upon said count signal and upon the remaining number of code positions occupied in said third storing means before the duplicate code completes the traverse of said third storing means after the unload category code has traversed all the code positions in said second storing means.

9. The stack-forming synchronizer set forth in claim 1 including:

third storing means having a plurality of code positions, each related to a respective one of said slots,

means for recording a duplicate unload category code in a position in said third storing means each time that the unload category code is recorded in said first comparing means,

an output terminal, and

means responsive to the end of traverse of the code positions in said third storing means by said duplicate unload category code for applying the last-mentioned code to said output terminal and for preventing re-recording of the last-mentioned code in said third storing means.

10. The stack-forming synchronizer set forth in claim 1 wherein: the first storing means comprises:

at least one store track positioned on a rotatable memory drum, and

means for rotating said memory drum,

the stack-forming synchronizer includes:

a retimer track positioned on said memory drum,

means for recording on said retimer track each output signal generated by said second comparing means,

reading means positioned to sense said retimer track,

means for converting the output of said reading means to an erase signal, and

means responsive to said erase signal for erasing from said first store track the article category code that was applied to said second comparing means when the associated output signal was generated.

11. A stack-forming synchronized for articles, each associated with one of a plurality of article category codes and each positioned in a respective one of a plurality of slots arranged in a sequence, comprising:

first storing means containing said article category codes, each in a code position related to a respective one of said slots,

second storing means containing a plurality of slot number codes, each related to a respective one of said slots, whereby each article category code is related to a respective slot number code,

third storing means having a plurality of code positions, each related to a respective one of said slots,

means for recording at least one unload category code in one of the code positions in said third storing means,

means for comparing in sequence each article category code in the first storing means with the unload category code in the third storing means and for generating an output signal when the codes are identical,

means responsive to each output signal for reading the slot number code out of said second storing means that is related to the article category code being applied to the comparing means when the output signal is generated,

means responsive to the read-out slot number code for discharging the article positioned in the slot related to the latter code,

a conveyor belt positioned beneath said slots, and

means for moving said belt at a speed such that the time interval required for advancing a reference point on the belt the width of one slot is substantially equal to the time interval between the comparison of the unload category code with one of said article category codes and the comparison of the former code with another one of the latter codes.

12. The stack-forming synchronizer set forth in claim 11 wherein said comparing means comprises:

a comparator,

means for transferring said unload category code to said comparator,

means for transferring to said compartor the article category code in said first storing means related to the slot associated with the code position from which the unload category code was transferred,

means for generating an output signal when an identity is found in said comparator, and

means for re-recording the unload category code in said third means with a selected advance in code position.

References Cited by the Examiner

UNITED STATES PATENTS

2,975,402	3/1961	Miller	347—147
3,067,874	12/1962	Rigg et al.	209—111.5
3,108,694	10/1963	Crain et al.	209—111.5
3,177,471	4/1965	Rabenda et al.	340—172.5

ROBERT C. BAILEY, *Primary Examiner.*

R. M. RICKERT, *Assistant Examiner.*