

External Memory Interfacing Techniques for the PIC18F8XXX

*Author: Tim Rovnak
Microchip Technology Inc.*

INTRODUCTION

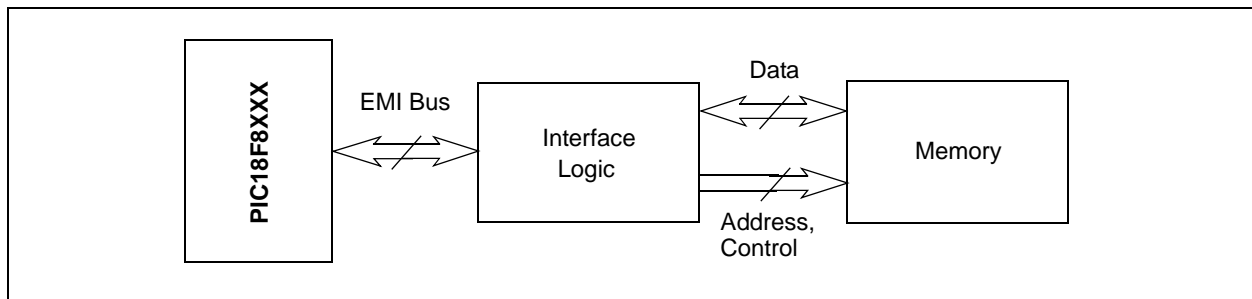
The PIC18FXXXX family offers the largest range of on-chip enhanced FLASH program memory and the richest selection of peripherals in the current line of Microchip microcontrollers. The PIC18F8XXX subset is made up of 80-pin parts that further extend the capabilities by providing access to external memory devices. Through the addition of external memory devices, an 8-bit application has the power to utilize unprecedented amounts of code or data; up to 2 Mbytes for an 8-bit microcontroller!

This application note describes the methodology to utilize the External Memory Interface on the PIC18F8XXX family of parts, and elaborates on the information provided in the data sheet. Connection diagrams are provided to demonstrate implementing various memory configurations. C and assembly code examples are included to assist in software development. It is expected that the reader be familiar with the PIC18 architecture and instruction set.

This application note contains the following main sections:

- **External Memory Interface (EMI) Overview**
Describes the Operating modes, pin implementation, registers, and control bits that determine the functionality of the External Memory Interface.
- **EMI Functional Implementation**
Discusses the mechanics behind the PIC18F8XXX 16-bit EMI. The most common operations of program fetching, user controlled reads, and user controlled writes are described.
- **16-bit EMI Operating Modes**
Details the timing and connection of the three EMI modes available to the PIC18F8XXX.
- **8-bit EMI Solutions**
Explains hardware and software concepts that allow access to byte-sized memories.
- **The Chip Enable Line and EMI Memory Mapped Peripherals**
Proposes a simple solution to using memory mapped peripherals in a PIC18F8XXX system.

FIGURE 1: EXTERNAL MEMORY INTERFACE DIAGRAM



AN869

EXTERNAL MEMORY INTERFACE (EMI) OVERVIEW

External Memory Interface offers the user many options, including:

- Operating the microcontroller entirely from external memory
- Using combinations of on-chip and external memory up to the 2-Mbyte limit
- Using external FLASH or EEPROM memory for reprogrammable application code or large data tables
- Using external RAM devices for storing large amounts of program or variable data
- Using external memory mapped devices and peripherals

EMI Operating Modes

There are four distinct EMI Operating modes available to the PIC18F8XXX devices. The EMI mode is determined by setting the two Least Significant bits of the CONFIG3L configuration byte. The function of the WAIT bit is described later in this application note. For more information on programming CONFIG bits, please see the "Special Features of the CPU" section in the respective data sheet.

Following is a summary for each of the External Memory Interface modes:

MC – The **Microcontroller Mode** accesses only on-chip FLASH memory. External Memory Interface functions are disabled. Attempts to read above the physical limit of the on-chip FLASH causes a read of all '0's (a NOP instruction).

MP – The **Microprocessor Mode** permits execution and access only through external program memory; the contents of the on-chip FLASH memory are ignored. The 21-bit program counter permits access to a 2-Mbyte linear program memory space.

MPBB – The **Microprocessor with Boot Block Mode** accesses on-chip FLASH memory within only the boot block. The boot block size is device dependent and is located at the beginning of program memory. Beyond the boot block, external program memory is accessed all the way up to the 2-MByte limit. Program execution automatically switches between the two memories as required.

EMC – The **Extended Microcontroller Mode** allows access to both internal and external program memories as a single block. The device can access its entire on-chip FLASH memory; above this, the device accesses external program memory up to the 2-MByte program space limit. As with Boot Block mode, execution automatically switches between the two memories as required.

REGISTER 1: CONFIG3L CONFIGURATION BYTE

R/P-1	U-0	U-0	U-0	U-0	U-0	R/P-1	R/P-1
WAIT	—	—	—	—	—	PM1	PM0
bit 7							bit 0

- bit 7 **WAIT:** External Bus Data Wait Enable bit
1 = Wait selections unavailable, device will not wait
0 = Wait programmed by WAIT1 and WAIT0 bits of MEMCOM register (MEMCOM<5:4>)
- bit 6-2 **Unimplemented:** Read as '0'
- bit 1-0 **PM1:PM0:** Processor Data Memory Mode Select bits
11 = Microcontroller mode
10 = Microprocessor mode
01 = Microcontroller with Boot Block mode
00 = Extended Microcontroller mode

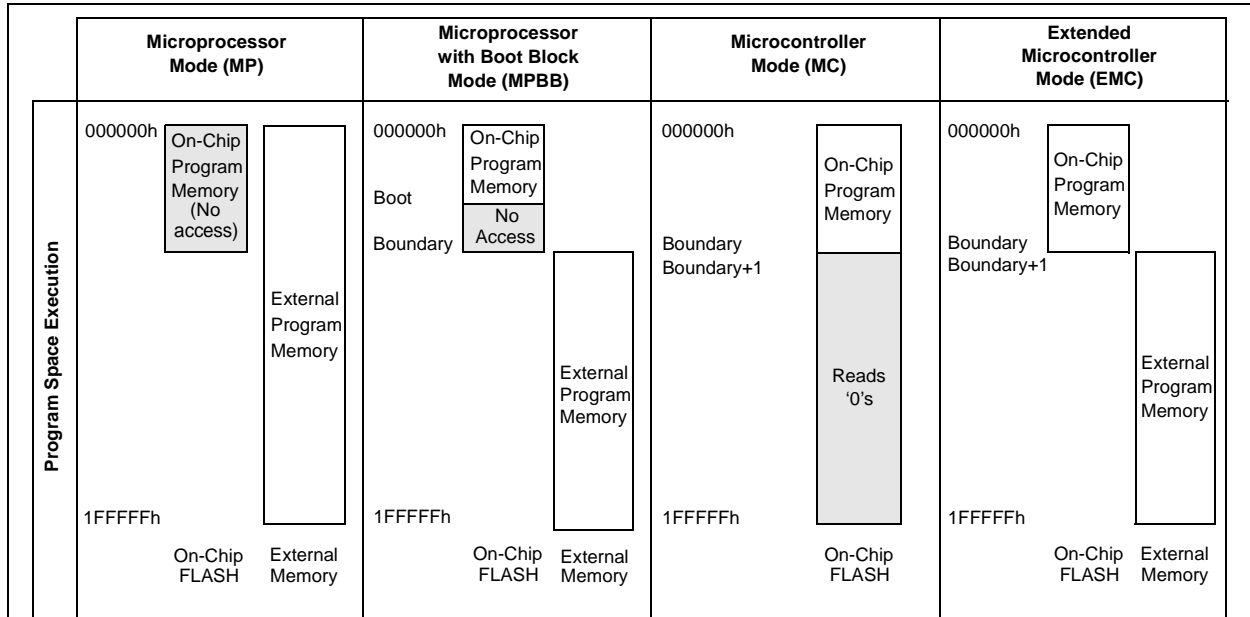
Legend:

R = Readable bit P = Programmable bit U = Unimplemented bit, read as '0'
- n = Value after erase '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

The External Memory Interface mode determines the memory mapping for the PIC18F8XXX. Figure 2 shows the internal and external memory mappings for the PIC18F8XXX.

In all modes, the microcontroller has complete access to internal data RAM and EEPROM.

FIGURE 2: MEMORY MAPS FOR PIC18FXXX PROGRAM MEMORY MODES



Boundary Values for Microprocessor with Boot Block, Microcontroller and Extended Microcontroller modes⁽¹⁾

Device	Boot	Boot+1	Boundary	Boundary+1	Available Memory Mode(s)
PIC18F6520	0007FFh	000800h	007FFFh	008000h	MC
PIC18F6525	0007FFh	000800h	00BFFFh	00C000h	MC
PIC18F6620	0001FFh	000200h	00FFFFh	010000h	MC
PIC18F6621	0007FFh	000800h	00FFFFh	010000h	MC
PIC18F6720	0001FFh	000200h	01FFFFh	020000h	MC
PIC18F8520	0007FFh	000800h	007FFFh	008000h	MP, MPBB, MC, EMC
PIC18F8525	0007FFh	000800h	00BFFFh	00C000h	MP, MPBB, MC, EMC
PIC18F8620	0001FFh	000200h	00FFFFh	010000h	MP, MPBB, MC, EMC
PIC18F8621	0007FFh	000800h	00FFFFh	010000h	MP, MPBB, MC, EMC
PIC18F8720	0001FFh	000200h	01FFFFh	020000h	MP, MPBB, MC, EMC

Note 1: PIC18F6X2X devices are included here for completeness to show the boundaries of their boot blocks and program memory spaces.

AN869

EMI Port Pin Implementation

The External Memory Interface is implemented across 4 ports (D,E,H,J) and 28 pins on the PIC18F8XXX. These pins are used for External Memory Interface address, data, and control lines, and are multiplexed with port and peripheral functions. They are mapped in a similar manner for all members of the PIC18F8XXX family, offering maximum compatibility (see Figure 3). Table 1 lists the pin designations and EMI descriptions for your reference.

The port pins listed in Table 1 are dedicated either to the EMI or to port/peripheral functions based on the EBDIS bit in the MEMCON register and the Operating mode defined by CONFIG3L. The MEMCON register map and the function of EBDIS are shown in Register 2 and Table 2, respectively. The additional bits found in MEMCON are described in later sections of this application note.

FIGURE 3: PIC18F8XXX EMI PIN ORIENTATION

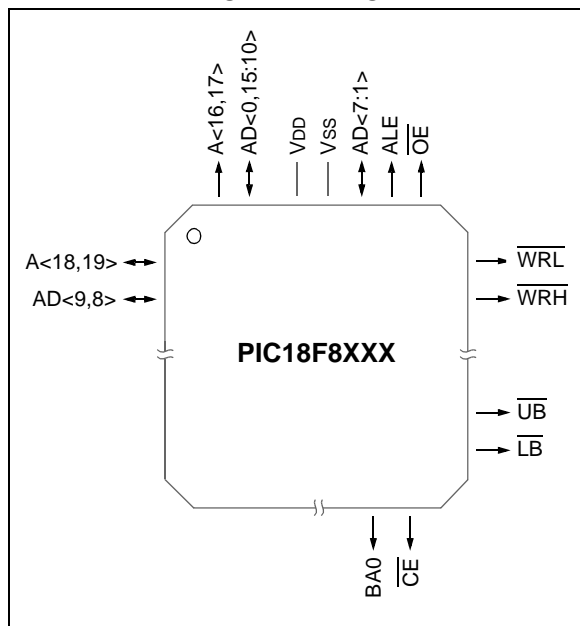


TABLE 1: PIC18F8XXX EMI BUS - I/O PORT FUNCTIONS

Name	Function
RD0/AD0	EMI Address bit 0 or Data bit 0.
RD1/AD1	EMI Address bit 1 or Data bit 1.
RD2/AD2	EMI Address bit 2 or Data bit 2.
RD3/AD3	EMI Address bit 3 or Data bit 3.
RD4/AD4	EMI Address bit 4 or Data bit 4.
RD5/AD5	EMI Address bit 5 or Data bit 5.
RD6/AD6	EMI Address bit 6 or Data bit 6.
RD7/AD7	EMI Address bit 7 or Data bit 7.
RE0/AD8	EMI Address bit 8 or Data bit 8.
RE1/AD9	EMI Address bit 9 or Data bit 9.
RE2/AD10	EMI Address bit 10 or Data bit 10.
RE3/AD11	EMI Address bit 11 or Data bit 11.
RE4/AD12	EMI Address bit 12 or Data bit 12.
RE5/AD13	EMI Address bit 13 or Data bit 13.
RE6/AD14	EMI Address bit 14 or Data bit 14.
RE7/AD15	EMI Address bit 15 or Data bit 15.
RH0/A16	EMI Address bit 16.
RH1/A17	EMI Address bit 17.
RH2/A18	EMI Address bit 18.
RH3/A19	EMI Address bit 19.
RJ0/ALE	EMI Address Latch Enable (ALE) Control pin.
RJ1/ \overline{OE}	EMI Output Enable (\overline{OE}) Control pin.
RJ2/ \overline{WRL}	EMI Write Low (\overline{WRL}) Control pin.
RJ3/ \overline{WRH}	EMI Write High (\overline{WRH}) Control pin.
RJ4/BA0	EMI Byte Address bit 0.
RJ5/ \overline{CE}	EMI Chip Enable (\overline{CE}) Control pin.
RJ6/ \overline{LB}	EMI Lower Byte Enable (\overline{LB}) Control pin.
RJ7/ \overline{UB}	EMI Upper Byte Enable (\overline{UB}) Control pin.

REGISTER 2: MEMCON REGISTER

R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
EBDIS	—	WAIT1	WAIT0	—	—	WM1	WM0
bit7						bit0	

- bit 7 **EBDIS:** External Bus Disable bit
 1 = External system bus disabled, all external bus drivers are mapped as I/O ports
 0 = External system bus enabled and I/O ports are disabled
- bit 6 **Unimplemented:** Read as '0'
- bit 5-4 **WAIT<1:0>:** Table Reads and Writes Bus Cycle Wait Count bits
 11 = Table reads and writes will wait 0 T_{CY}
 10 = Table reads and writes will wait 1 T_{CY}
 01 = Table reads and writes will wait 2 T_{CY}
 00 = Table reads and writes will wait 3 T_{CY}
- bit 3-2 **Unimplemented:** Read as '0'
- bit 1-0 **WM<1:0>:** TBLWRT Operation with 16-bit Bus bits
 1x = Word Write mode: TABLAT<0> and TABLAT<1> word output, \overline{WRH} active when TABLAT<1> written
 01 = Byte Select mode: TABLAT data copied on both MS and LS Byte, \overline{WRH} and $(\overline{UB}$ or $\overline{LB})$ will activate
 00 = Byte Write mode: TABLAT data copied on both MS and LS Byte, \overline{WRH} or \overline{WRL} will activate

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
 - n = Value at POR '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

TABLE 2: EBDIS CONTROL AND PORT PIN FUNCTION FOR EMI MODES

Mode	Pin Function	EBDIS Control
Microcontroller	Port Functions	The EBDIS bit has no effect.
Microprocessor	External Memory	The EBDIS bit has no effect.
Microprocessor with Boot Block	Shared	While fetching instructions <i>externally</i> or executing table read/table write operations <i>externally</i> , the EBDIS bit has no effect.
AND Extended Microcontroller		While fetching instructions <i>internally</i> or executing table read/table write operations <i>internally</i> , EBDIS control bit is capable of changing the pins from external memory to I/O port functions. When EBDIS = 0, the address and data pins are tri-stated and the control lines are pulled to inactive states. When EBDIS = 1, the pins function as I/O ports.

In summary, the Memory Operating mode is determined by the CONFIG3L register and the functionality of the port pins is determined by the EBDIS bit. However, for the three modes that are specific to external memory, the EBDIS is controlled manually only when execution occurs internally.

AN869

EMI FUNCTIONAL IMPLEMENTATION

The three most common functions of the External Memory Interface are:

- Program Fetches
- Data Reads
- Data Writes

This section describes how these operations are executed by the EMI. As will be shown, the timings for program fetches and data reads are almost identical. Data writes are presented generically here and specifics are detailed in a later section.

16-Bit Bus Overview

The PIC18F8XXX is defined as a 16-bit bus because the interface has 16 data lines for word-wide (2-byte) access. These data lines are shared with address lines and are labeled AD<15:0>. Because of this, 16 bits of latching are necessary to demultiplex the address and data. There are four additional address lines labeled A<19:16>. The capability of the PIC18F8XXX product is not limited to 16-bit memory configurations. Single byte external memory bussing is possible. This is described later.

The PIC18 architecture provides an internal program counter of 21 bits, offering a capability of 2 Mbytes of addressing. However, as noted above, the address lines of the external memory interface number only 20

A<19:0>. This is because the 16-bit bus is based on a word boundary. Only 20 bits are necessary in this type of access. However, if needed, an additional address line exists that provides the even/odd byte boundary. It is called BA0 and it reflects the state of the least bit in the program counter. BA0 is typically not used in PIC18F8XXX external connections.

Note: If BA0 is not needed for the application then it should be left unconnected. This is because any time external memory functions are active, BA0 will be active.

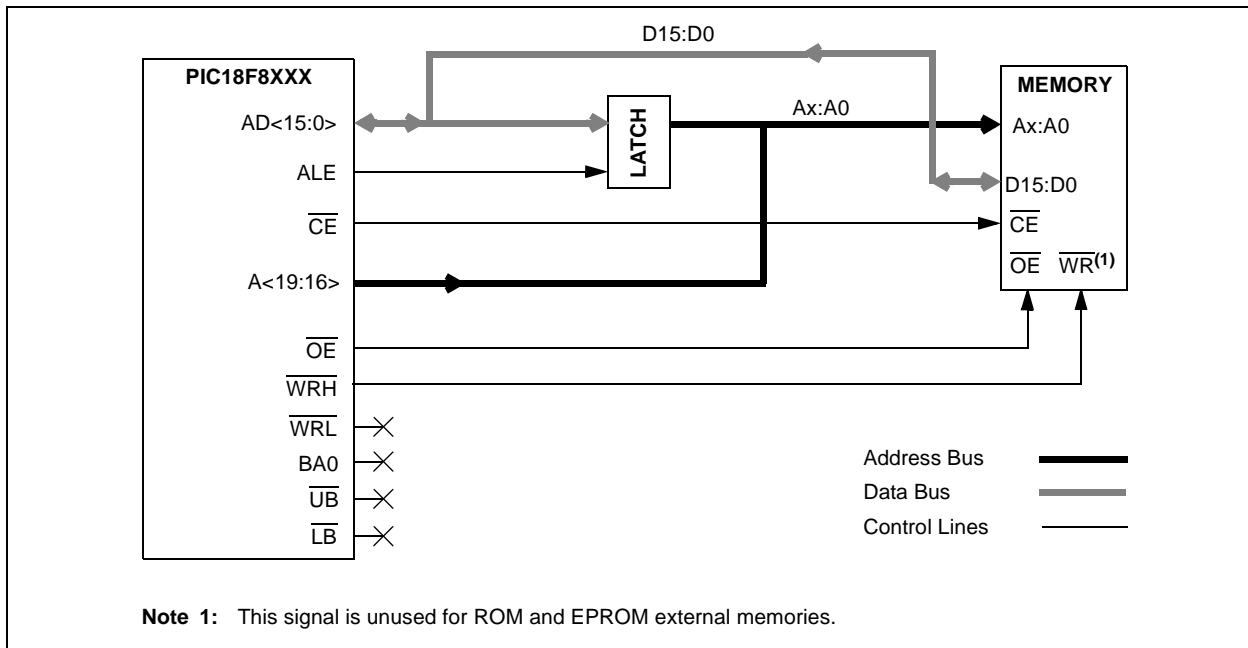
There are seven control lines that are used in the EMI: \overline{OE} , \overline{WRH} , \overline{WRL} , \overline{CE} , \overline{UB} , \overline{LB} and ALE. All of these lines except \overline{OE} may be used during data writes. All of these lines except \overline{WRH} and \overline{WRL} may be used during fetches and reads. The application will determine which control lines are necessary. The timings of these control lines are detailed in the pages that follow.

If EMI is enabled but execution is occurring internally, the address and data lines are tri-stated and the control lines are set in the following manner:

- \overline{OE} , \overline{WRH} , \overline{WRL} , \overline{CE} , \overline{UB} , and $\overline{LB} = 1$
- ALE and BA0 = 0

Figure 4 shows a basic connection diagram for the PIC18F8XXX. Complete connection diagrams are provided under each of the EMI modes in the "Program Fetches" section.

FIGURE 4: BASIC EXTERNAL MEMORY CONNECTION DIAGRAM



Program Fetches

Generally speaking, during one instruction cycle, a 2-byte instruction is executed while the external memory interface fetches the next 2-byte instruction. When an external memory is loaded with code and the interface circuitry is connected properly, program fetching is essentially transparent to the user code. The CPU responds as if it were fetching instructions from internal memory. The only limitation is bus loading characteristics and speed of external memory. At the time this application note was published, the maximum bus speed of the EMI is limited to 25 MHz. The following paragraph describes the timing of external program fetches.

The PIC18 family runs from a clock that is four times faster than its instruction cycle. The four clock pulses are a quarter of the instruction cycle in length and are referred to as Q1, Q2, Q3, and Q4. During Q1, ALE is enabled while address information A<15:0> are placed on pins AD<15:0>. At the same time, the upper address information A<19:16> are available on the upper address bus. On the negative edge of ALE, the address is latched in the external latch. At the beginning of Q3, the \overline{OE} output enable (active low) signal is generated. Also, at the beginning of Q3, BA0 is generated. This signal will be active high only during Q3, indicating the state of the program counter Least Significant bit. At the end of Q4, \overline{OE} goes high and data (16-bit word) is fetched from memory at the low-to-high transition edge of \overline{OE} . The timing diagram for all signals during external memory code execution and table reads is shown in Figure 5. Table reads are discussed in the next section.

FIGURE 5: EMI TIMING FOR PROGRAM FETCH AND TABLE READ (MP MODE)

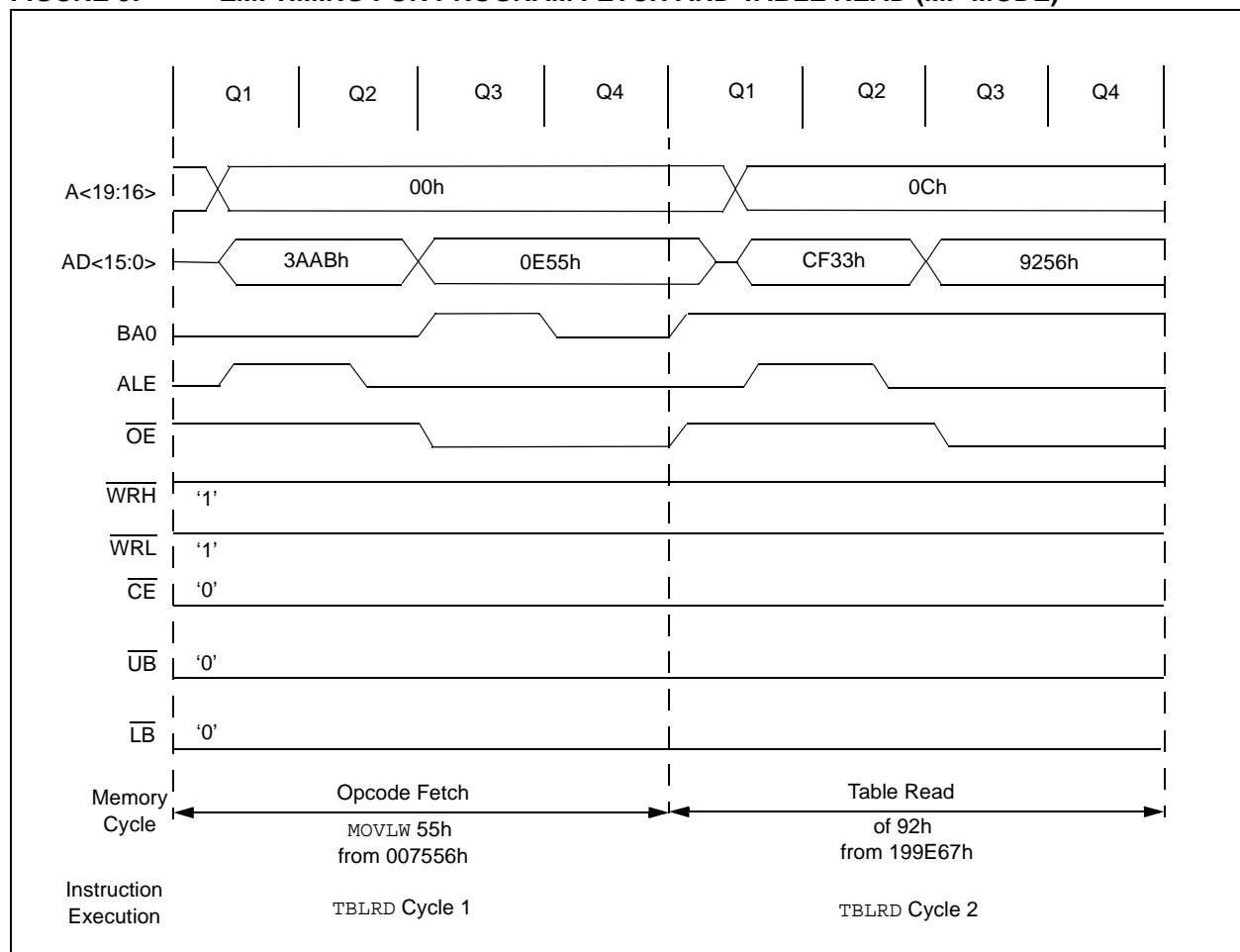


Table Reads

The user code controls data reads through the use of table reads which are very similar to program fetching. The timings are essentially the same (see the previous section) but unlike program fetching, reads are executed on a single byte basis. Therefore, the control signal BA0 is the only signal that behaves differently (see Figure 5). The mechanics of table reads can be found in the following sections.

TABLE REGISTERS

The following two control registers are used in conjunction with the table read instructions:

- TABLAT register
- TBLPTR registers

The table latch (TABLAT) is an 8-bit Special Function Register (SFR). The table latch is used to hold 8-bit data obtained from the read of program memory (internal or external).

The table pointer (TBLPTR) addresses a byte of program memory (internal or external). The TBLPTR is made up of three Special Function Registers:

- Table Pointer Upper byte (TBLPTRU)
- Table Pointer High byte (TBLPTRH)
- Table Pointer Low byte (TBLPTRL)

These three registers join to form a 21-bit wide pointer which allows the device to address up to 2 Mbytes of program memory space. These registers are similarly used in data write operations.

TABLE READ INSTRUCTION (TBLRD*)

The TBLRD* instruction is used to retrieve data from internal or external program memory and places it into data memory. TBLPTR points to a byte address in program memory space. Executing TBLRD* places the byte into TABLAT. In addition, TBLPTR can be modified automatically for the next table read operation:

- TBLRD*+ (post-increment)
- TBLRD*- (post-decrement)
- TBLRD*+* (pre-increment)

During table read operations, the Least Significant bit of TBLPTR is copied to BA0. The values of TBLPTR<20:1> appear on address pins A<19:0>. Next, 16-bits of data are read on to the data bus. Circuitry in TABLAT will select either the high or the low byte of the data from the 16-bit bus, based on the Least Significant bit of the address. That is, when LSb is '0', the lower byte (D<7:0>) is selected; when LSb is '1', the upper byte (D<15:8>) is selected.

The code in Example 1 describes the use of the table read.

EXAMPLE 1: USING THE TBLRD* INSTRUCTION

```
MOVLW    UPPER (SampleTable)    ;Initialize Table Pointer
MOVWF    TBLPTRU                ;with the starting address
MOVLW    HIGH  (SampleTable)    ;of the Table
MOVWF    TBLPTRH                ;
MOVLW    LOW   (SampleTable)    ;
MOVWF    TBLPTRL                ;
TBLRD*+  ;Read Program memory and increment Table Pointer
MOVFF    TABLAT, Mydata         ;Store table latch to FSR Mydata
```


Table Writes

The user code controls data writes through the use of table writes. Table write timing is dependent on the EMI mode (detailed in the “16-Bit EMI Operating Modes” section).

TABLE REGISTERS

In a manner similar to reads, TABLAT and TBLPTR are also used during writes. TABLAT holds the data byte that will be used in the write operation. The address of the program memory (internal or external) location is specified by TBLPTR.

TABLE WRITE INSTRUCTION (TBLWT*)

The TBLWT* instruction is used in the process that writes to program memory. TBLPTR can be modified automatically for the next table write operation:

- TBLWT*+ (post-increment)
- TBLWT*- (post-decrement)
- TBLWT*+* (pre-increment)

When a TBLWT* is executed, the Least Significant bit of TBLPTR is copied to BA0 and the values of TBLPTR<20:1> appear on address pins A<19:0>. Then, depending on the EMI mode and the TBLPTR address, data may be presented on the data bus. This is explained below.

When a TBLWT* is executed that causes data to be physically placed on the bus, the data is always in the form of two bytes. These 16 bits may contain two individual bytes or may contain 1 byte copied. This depends on the EMI. Then, based on the state of the control lines, either one or both bytes will be written to the external memory device during a single instruction cycle.

Word Write mode (detailed in “16-Bit EMI Operating Modes”) is a special case where a one-byte holding register is used in conjunction with TABLAT. During TBLWT* instructions to even addresses, the holding register is loaded but no data is presented externally. During TBLWT* instructions to odd addresses, the holding register and the TABLAT are presented on the data bus and written at the same time during one instruction cycle.

The code in Example 2 describes the use of the table write.

EXAMPLE 2: USING THE TBLWT* INSTRUCTION

```

MOVLW    UPPER (SampleTable)      ;Initialize Table Pointer
MOVWF    TBLPTRU                  ;with the starting address
MOVLW    HIGH  (SampleTable)      ;of the Table
MOVWF    TBLPTRH                  ;
MOVLW    LOW   (SampleTable)      ;
MOVWF    TBLPTRL                  ;
MOVLW    LOW   (DataWord)         ;Load table latch with low byte
MOVWF    TABLAT                   ;of value to write
TBLWT*+  ;Write to Program memory and increment Table Pointer
MOVLW    HIGH  (DataWord)         ;Load W register with high byte of value to write
MOVWF    TABLAT                   ;Transfer high byte of value to table latch
TBLWT*  ;Write to next location/Word

```

16-BIT EMI OPERATING MODES

This section details the operation of the EMI Operating modes that are determined by the two LSBs of the MEMCON register. The EMI Operating mode chosen dictates the appropriate types of external memory available, and the method for connection.

MEMCON<1:0>

- 1x = Word Write Mode
- 01 = Byte Select Mode
- 00 = Byte Write Mode

Word Write Mode

Figure 6 shows an example of 16-bit Word Write mode for PIC18F8XXX devices. This mode is used for word-wide memories which includes some of the EPROM and FLASH type memories. This mode allows program fetches, table reads, and table writes from all forms of 16-bit memory.

This method makes a distinction between TBLWT* cycles for even or odd addresses. During a TBLWT* cycle to an even address (TBLPTR<0> = 0), the TABLAT data is transferred to a holding latch and the external address data bus is tri-stated for the data portion of the bus cycle. No write signals are activated.

During a TBLWT* cycle to an odd address (TBLPTR<0> = 1), the TABLAT data is presented on the upper byte of the AD<15:0> bus. At the same time, the contents of the holding latch are presented on the lower byte of the AD<15:0> bus. The WRH signal is strobed for one write cycle; the WRL pin is unused. The signal on the BA0 pin indicates the LSb of TBLPTR but it is left unconnected. The UB and LB signals are both active low to select the two bytes. The obvious limitation to this method is that the table write must be done in pairs on a specific word boundary to correctly write a word location.

FIGURE 6: WORD WRITE MODE EXAMPLE

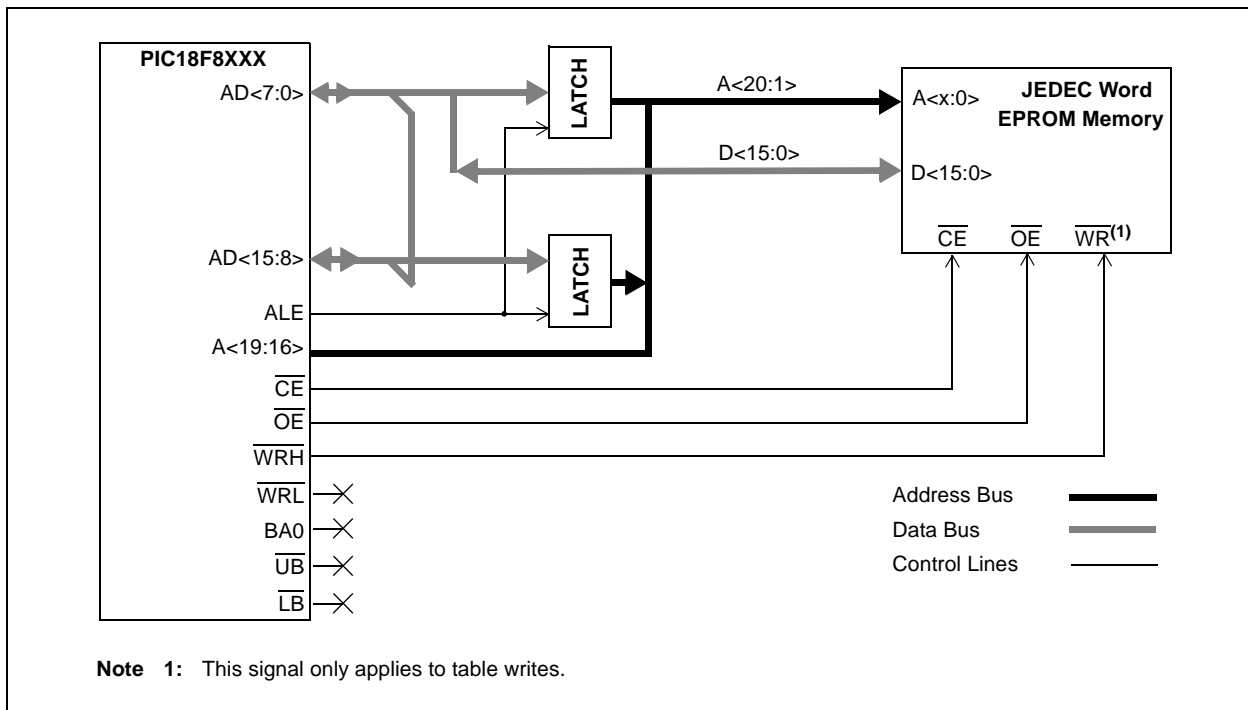
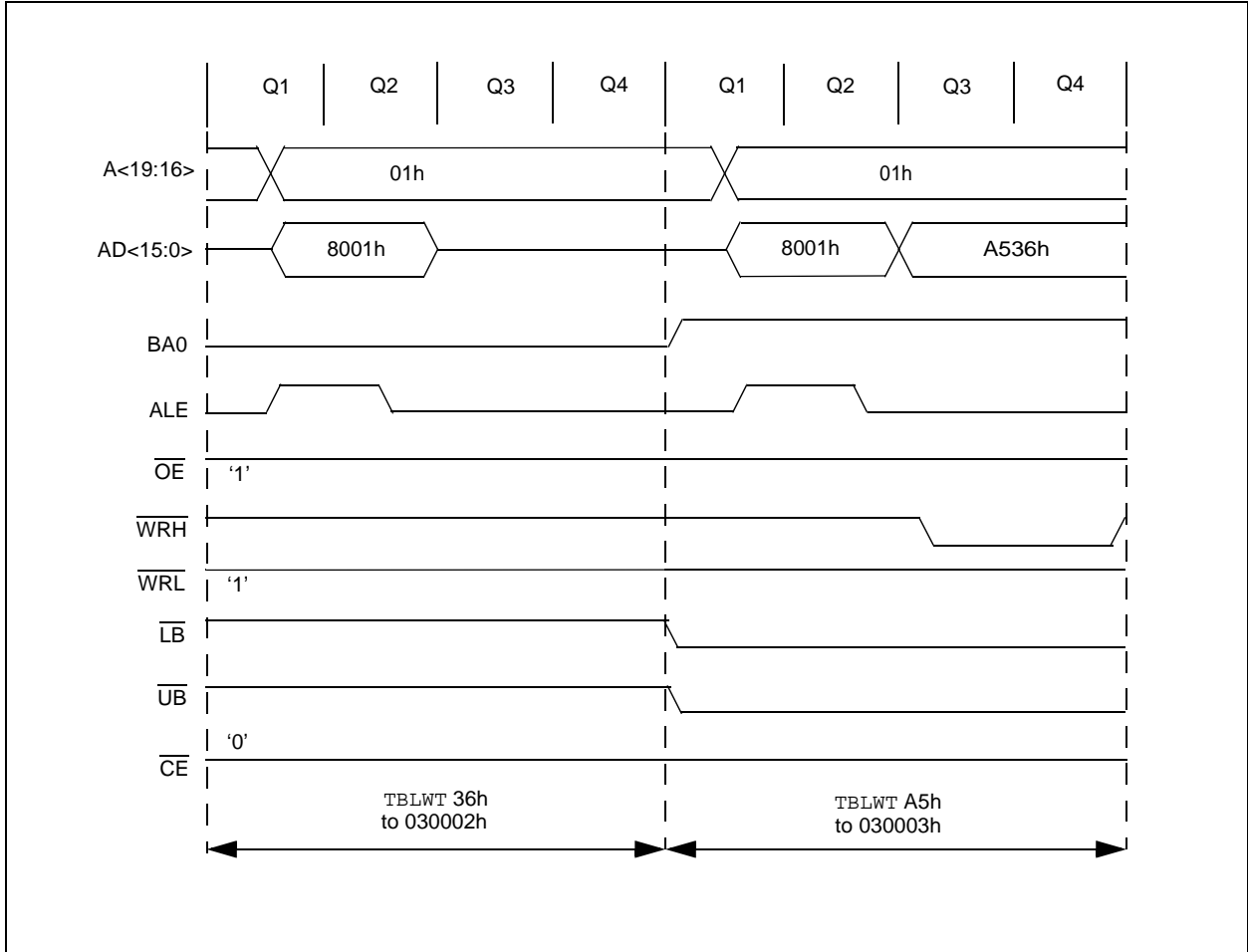


FIGURE 7: WORD WRITE MODE TIMING



AN869

Byte Select Mode

Figure 8 shows an example of 16-bit Byte Select mode for PIC18F8XXX devices. This mode allows table write operations to word-wide external memories with byte selection capability. This generally includes both word-wide FLASH and SRAM devices. During a TBLWT* cycle, the TABLAT data is presented copied on both the upper and lower byte of the AD<15:0> bus. The WRH signal is strobed for each write cycle; the WRL pin is not used. The BA0 or UB/LB signals are used to select the byte to be written based on the Least Significant bit of the TBLPTR register.

FLASH and SRAM devices use different control signal combinations to implement Byte Select mode. JEDEC standard FLASH memories use the BA0 signal from the controller as a byte address. Conversely, JEDEC standard static RAM memories use the UB or LB signals to select the byte.

FIGURE 8: BYTE SELECT MODE EXAMPLE

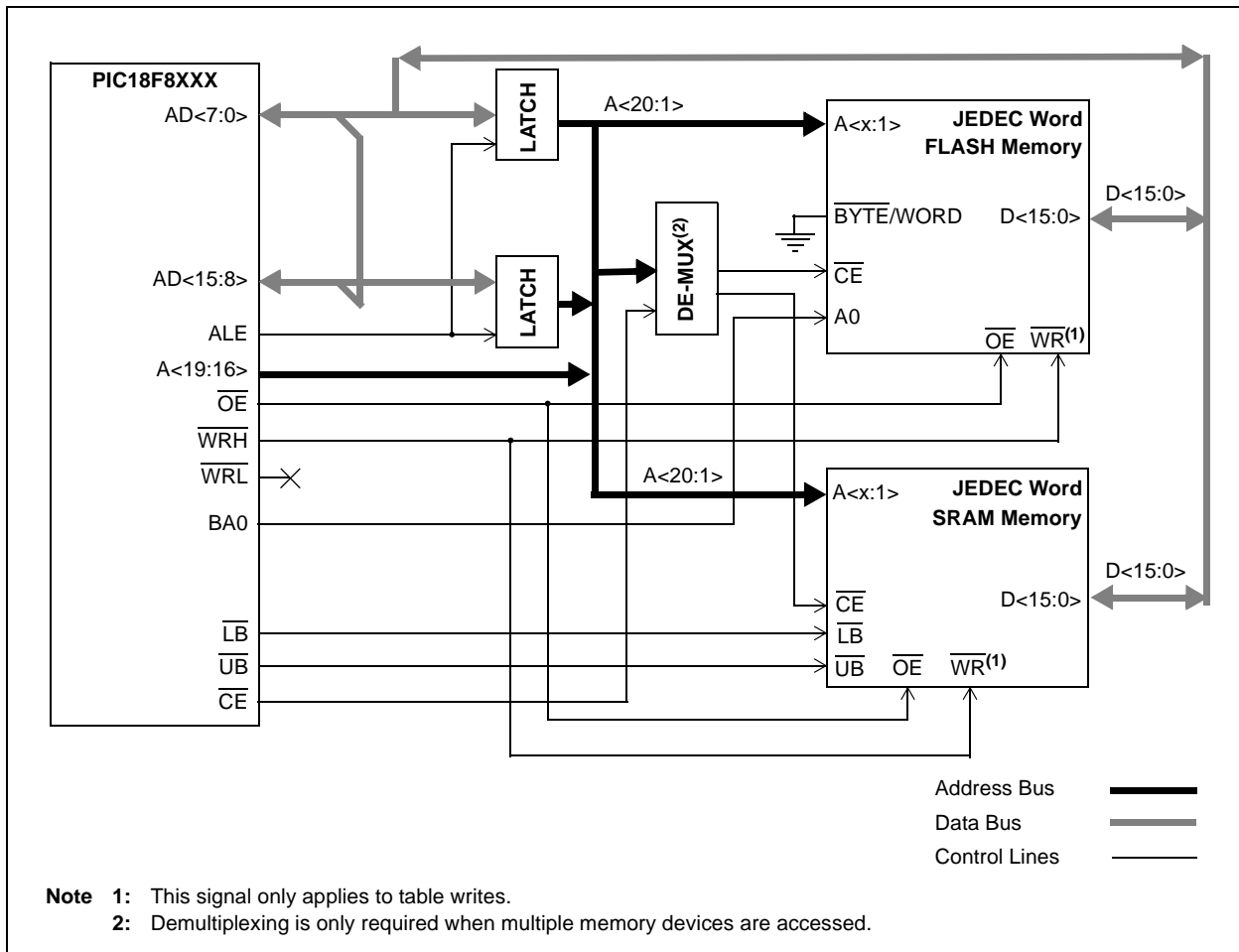
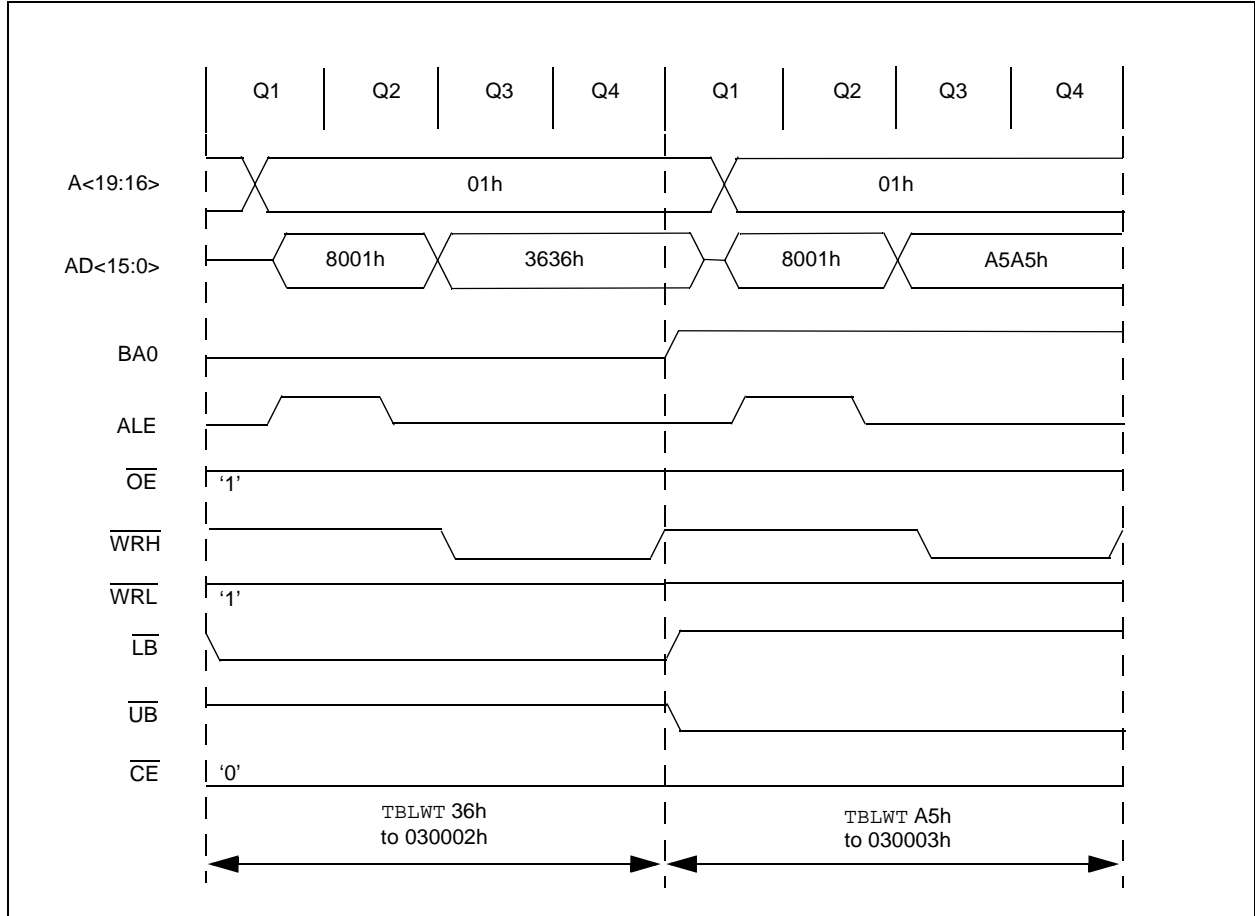


FIGURE 9: BYTE SELECT MODE TIMING



AN869

Byte Write Mode

Figure 10 shows an example of 16-bit Byte Write mode for PIC18F8XXX devices. This mode is used for two separate 8-bit memories connected for 16-bit operation. This generally includes basic EPROM and FLASH devices. It allows table writes to byte-wide external memories. During a TBLWT instruction cycle, the TABLAT data is presented on the upper and lower bytes of the AD<15:0> bus. The appropriate WRH or WRL control line is strobed on the LSb of the TBLPTR.

FIGURE 10: BYTE WRITE MODE EXAMPLE

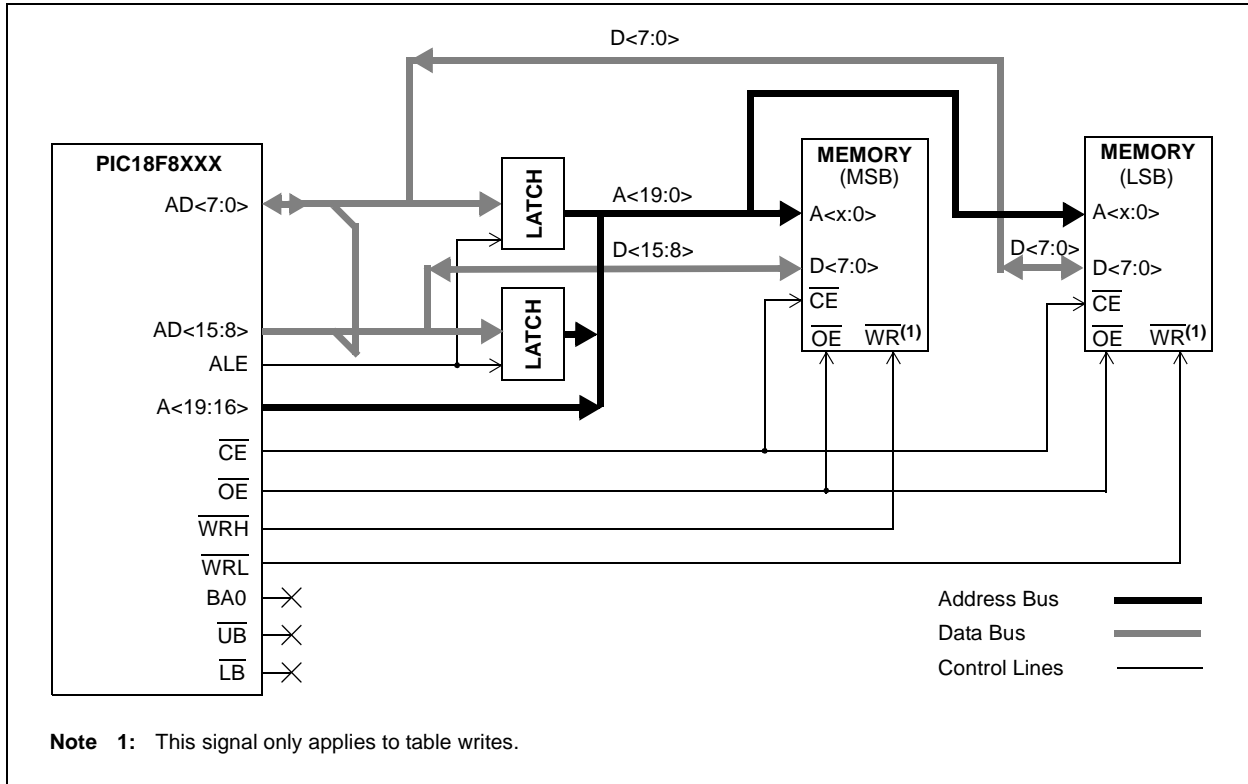
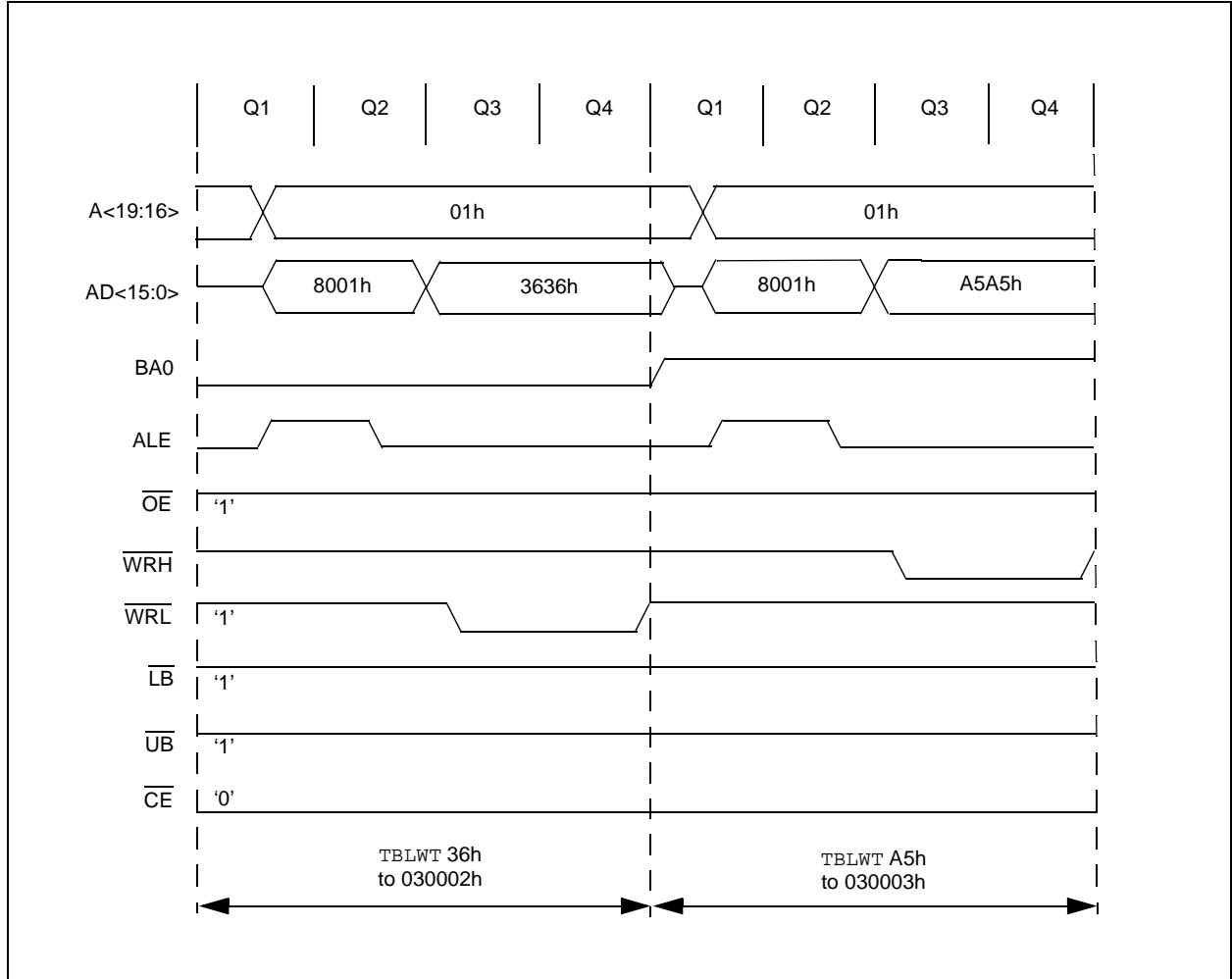


FIGURE 11: BYTE WRITE MODE TIMING



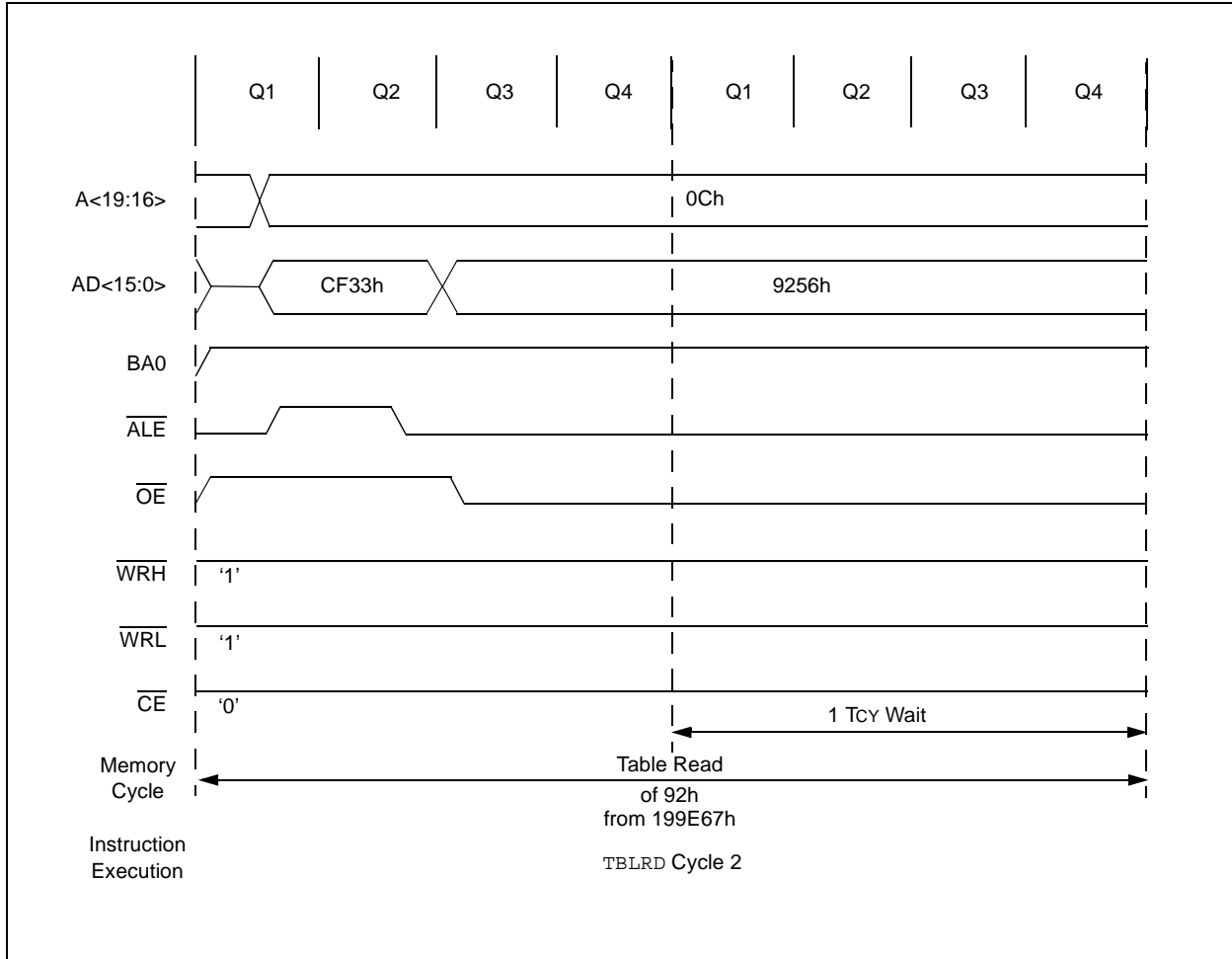
AN869

WAIT States

Depending on the processor speed and the application, it may be necessary or advantageous to use slower memories on the EMI. In this situation, wait states may be enabled. For wait states to be enabled, the WAIT bit located in the CONFIG3L configuration byte must be cleared. The length of one wait state is the equivalent

of one instruction cycle. The number of wait states is determined by WAIT<1:0> bits in MEMCON<5:4>. When wait states are in use, the length of time that the output enable, \overline{OE} , line is active (low) is extended to allow the external memory device to access its data. Figure 12 describes the timing of one wait state during a program memory read.

FIGURE 12: WAIT STATE TIMING FOR TBLRD (MP MODE)



8-BIT EMI SOLUTIONS

Economical 8-Bit Memory Data Storage

This section presents an economical solution for a single 8-bit external SRAM for data storage only. Beyond the SRAM itself, the only additional component(s) are the required 16 bits of latch functionality. This could be two 8-bit latches or a single 16-bit latch. The physical limit of external memory for the PIC18F8XXX in Extended Microcontroller mode is 1,920 Kbytes. The solution described here will limit SRAM storage to half of that: 960 Kbytes. This solution only limits the address capability of the microprocessor. The physical SRAM usage is 100%. In other words, a 128-Kbyte SRAM is needed for 128K of usage.

HARDWARE ANALYSIS

This method uses Byte Write mode. BA0 is not connected. AD<15:0> from the PIC® microcontroller is mapped through the latch to A<15:0> of SRAM. A<19:16> are mapped directly from the PIC microcontroller to A<19:16> of SRAM for larger memories. AD<7:0> are mapped directly to D<7:0> of the SRAM. WRL of the PIC microcontroller maps to WR of the SRAM. WRH is left unconnected. CE is connected to the SRAM CE line. OE is connected to the SRAM OE line. OE is connected to the SRAM OE line.

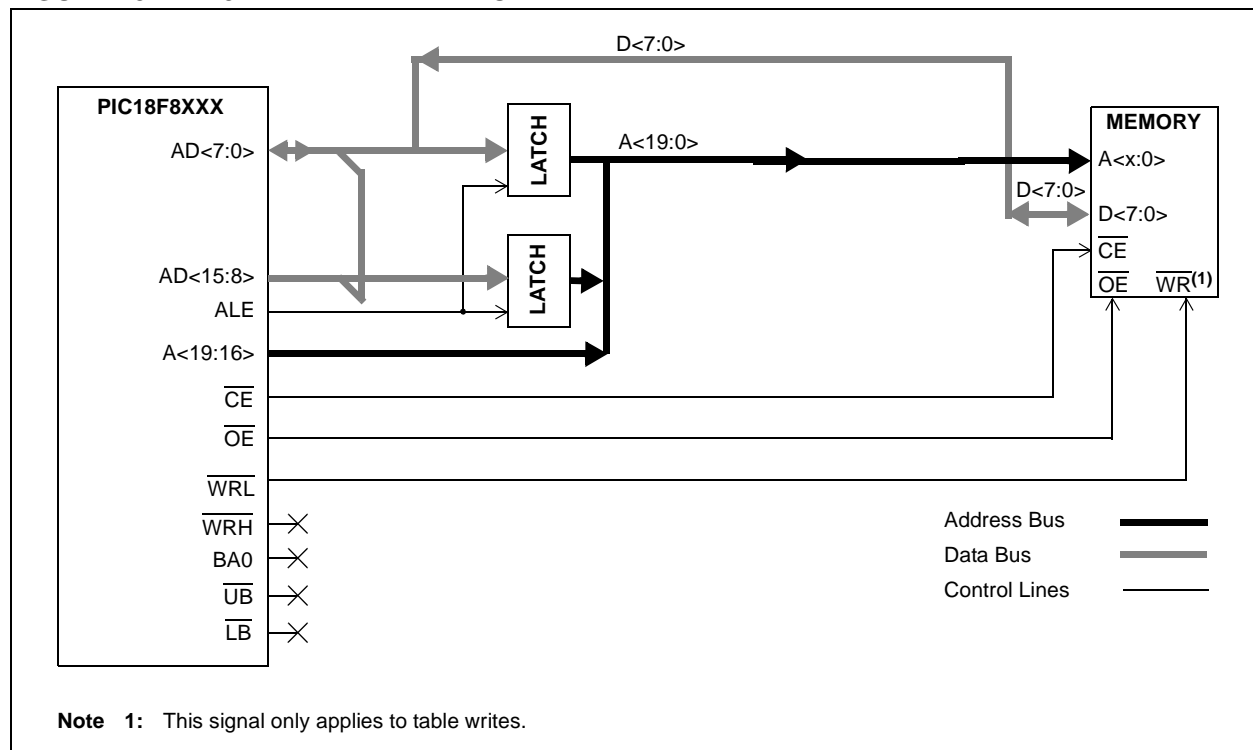
SOFTWARE ANALYSIS

This method will permit only table operations to even addresses. A table write to an even address will copy the contents of TABLAT to both bytes of the AD<15:0> bus and activate the WR signal. Only the lower byte will be written. A table write to an odd address will copy the contents of TABLAT to both bytes of the AD<15:0> bus and activate the WRH signal. In this case, nothing will be written. A valid table read must be initiated for the even address only. Using this strategy, any table operation must align to even boundaries, and any table pointer adjustments must compensate. Where possible, it is recommended to use the built-in increment and decrement operators for the table pointer. Otherwise, manual adjustment between the Table Pointer registers becomes necessary during rollover/rollunder.

ASSEMBLY CODE

Example 3 shows an example of assembly code to write and read 128 bytes of data.

FIGURE 13: 8-BIT BYTE WRITE MODE EXAMPLE



AN869

EXAMPLE 3: 8-BIT DATA STORAGE ASSEMBLY CODE

```
; *** Enable External Memory Bus
      CLRF    MEMCON                ; BYTE WRITE mode

; *** Set up Table Pointer
      MOVLW  b'00000010'           ; Set Table Pointer to an
      MOVWF  TBLPTRU                ; external memory region
      MOVLW  b'00000000'
      MOVWF  TBLPTRH
      MOVLW  b'00000000'
      MOVWF  TBLPTRL

; *** SFR initializations
      MOVLW  h'AA'                  ; Load TABLAT with test data
      MOVWF  TABLAT
      CLRF  count

; *** Write data
writeloop
      TBLWT*+                        ; Write low byte (real write)
      TBLWT*+                        ; Write high byte (dummy write)
      INCF  count
      BNZ   writeloop

      CLRF  TBLPTRH                  ; Reset Table Pointer
      CLRF  TBLPTRL

; *** Read data
readloop
      TBLRD*+                        ; Read Low byte
      MOVF  TABLAT,W                 ; Store in W temporarily
      TBLRD*+                        ; Read next byte (garbage)
      MOVWF TABLAT
      MOVLW h'AA'
      SUBWF TABLAT,W
      BNZ   fail
      INCF  count
      BNZ   readloop

success
      BRA   success

fail
      BRA   fail
```

MPLAB® C18 CODE

In order to implement this in C, it is necessary that for each byte of storage, two table writes are executed. This can be done for an array by a 2-byte assignment such as this:

```
rom unsigned int *DataPtr; //2 bytes
```

Then, when assigning a value in the following manner, the low byte gets written to RAM and the high byte gets ignored since WRH is unconnected:

```
DataPtr[i] = value;
```

When reading a value, the destination assignment will need to be:

```
unsigned char v; //1 byte
```

Then, when retrieving the data in the following manner, only the low byte is retrieved:

```
v = DataPtr[i];
```

Note: If the destination is of the type, unsigned int, both bytes would contain the same value because BA0 is unconnected.

Example 4 shows an example of C18 code that stores to a single 8-bit memory. This code runs a simple loop that stores 256 values to a buffer, then reads back and verifies.

EXAMPLE 4: 8-BIT DATA STORAGE C CODE

```

void main(void)
{
    rom unsigned int *DataPtr;
    unsigned char i;
    unsigned char v;

    // Setup your data memory address.
    DataPtr = (rom unsigned int*)0x20000;

    // Write 0 - 255 at address
    for ( i = 0; i < 255; i++ )
        DataPtr[i] = i;

    // Read back values
    for ( i = 0; i < 255; i++ )
    {
        v = DataPtr[i];
        if ( v != i )           // And compare it with expected value.
            while(1);         // On failure, it will loop here forever.
    }

    // When done stop here.
    while(1);
}

```

This code example, however, does not describe how to convert all variants of C data types to this interface. To accommodate all data types, functions would need to be defined to pack and unpack the data to the unsigned integer format. The following is an example on how this can be done for the integer data type:

EXAMPLE 5: PACK/UNPACK FUNCTIONS FOR INTEGER DATA TYPE

```

rom int *intVal = 0x20000;           // Assume that external RAM is at 128K
int tempInt;                         // Temporary internal RAM

// For every access to external RAM variable, use a special
// function. For instance, there could be
// ReadInt(), WriteInt(), ReadLong(), WriteLong(), ReadMem(), WriteMem()

// Read intVal that is located in external RAM.
TempInt = ReadInt(intVal) ;

// Write to tempInt
WriteInt(intVal, tempInt);

```

AN869

ReadInt() and WriteInt() functions would be implemented as shown in Example 6:

EXAMPLE 6: ReadInt() AND WriteInt() FUNCTIONS

```
int ReadInt(rom int* ptr)
{
    union
    {
        int i;
        char v[2];
    } t;
    unsigned short long sl;

    sl = (unsigned short long)ptr;
    sl <<= 1;
    ptr = (rom int*)sl;           // 16-bit to 8-bit address mapping.
    t.v[0] = *ptr++;             // Read int in little-endian format.
    t.v[1] = *ptr;               // Depending on where memory is wired,
    return t.i;                  // actual value is available at odd or
                                // even addresses only.
}

void WriteInt(rom int* ptr, int val)
{
    union
    {
        int I;
        char v[2];
    } t;
    unsigned short long sl;

    sl = (unsigned short long)ptr;
    sl <<= 1;
    ptr = (rom int*)sl;         // 16-bit to 8-bit address mapping.

    t.i = val;

    *ptr++ = t.v[0];
    *ptr = t.v[1];
}
```

The same concept can be applied to other data types. It is apparent that this conversion will create significant amounts of additional code. With Microchip C18 C Compiler using the static overlay model, each of the above functions could take up to 80 bytes. In addition, each reference to an external variable would require up to 12 bytes of program memory. These numbers are largely dependent on the compiler and the application.

8-Bit Memory Full Capacity Data Storage

Presented here is another solution to use 8-bit RAM with the PIC18F8XXX 16-bit interface for data storage. This system allows full access to the external memory data storage capabilities with no additional software overhead. This solution takes advantage of certain features in Byte Select mode. Details can be found in the "Software and Timing Analysis" section. Other than the required 16 bits of latch, this method uses a bidirectional buffer and one logic gate.

HARDWARE ANALYSIS

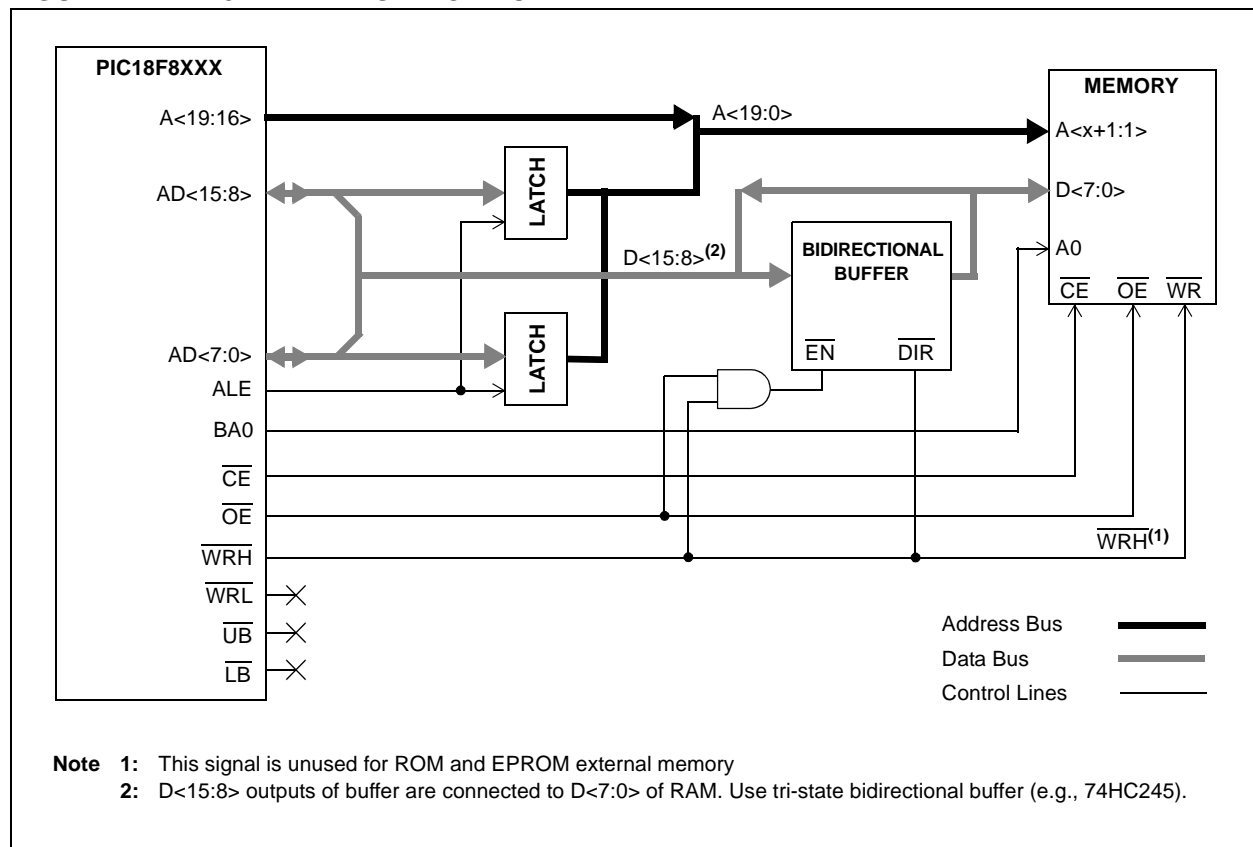
The address lines are connected as follows: connect the BA0 pin directly to the A0 pin of the 8-bit RAM; connect the demultiplexed address bus lines (output of

latches) A<15:0> to A<16:1> of the RAM. For larger memory requirements, connect A<19:16> of the PIC microcontroller to A<20:17> of the RAM.

The data lines are connected as follows: connect AD<7:0> of the PIC microcontroller data bus to D<7:0> of RAM; connect AD<15:8> of the PIC microcontroller to D<7:0> of the RAM through the bidirectional buffer.

The \overline{CE} , \overline{OE} , and \overline{WRH} control lines are connected as in the standard Byte Select mode. However, care must be taken with the enable signal for the bidirectional buffer. The buffer should be enabled only during actual write or read access to RAM. Otherwise, higher and lower bytes of AD<15:0> could be shorted. A solution to this situation is to use the \overline{OE} and \overline{WRH} signals through an AND gate to enable the buffer.

FIGURE 14: 8-BIT BYTE SELECT MODE EXAMPLE



SOFTWARE AND TIMING ANALYSIS

Data storage is accessed with `TBLRD*` and `TBLWT*` instructions in standard fashion. The function of table writes in Byte Select mode is critical for this solution. The following is a discussion of the signal timings in this solution for both operations. Please refer to Figure 5 for the table read timings and Figure 9 for the table write timings.

`TBLRD` cycle 2 shows when the actual read occurs. During Q1 and Q2, the address is placed on the AD bus and the use of the ALE signal latches the address. The BA0 is active during Q1 to Q4, providing demultiplexed A0 address bit for the RAM. The use of the \overline{OE} signal disables the buffer during this access, allowing proper 16-bit address to latch. During Q3 and Q4, the \overline{OE} signal is activated; this enables the buffer which copies the RAM data on D<7:0> to D<15:8>. Now, depending on the status of BA0 (odd or even address), the PIC microcontroller will copy the data of LSB or MSB to TABLAT. As LSB and MSB contain the same data, TABLAT will contain the proper value for any address.

`TBLWT` cycle 2 shows when the actual write occurs. During Q1 and Q2, the address is placed on the AD bus and the use of the ALE signal latches the address. The BA0 is active during Q1 to Q4, providing demultiplexed A0 address bit for RAM. The use of the \overline{WRH} signal disables the buffer during this access, allowing the proper 16-bit address to latch. During Q3 and Q4, the \overline{WRH} signal is activated; this enables the buffer which copies the RAM data on D<15:8> to D<7:0>. This may appear to be a problem as D<15:8> and D<7:0> are shorted through buffer. However, it is not because of Byte Select mode. In this mode, the PIC microcontroller writes the same data to LSB and MSB. Therefore, D<15:8> and D<7:0> contain the same data irrespective to the address being written. Therefore, input data of RAM remains the same even after this indirect short.

THE CHIP ENABLE LINE AND EMI MEMORY MAPPED PERIPHERALS

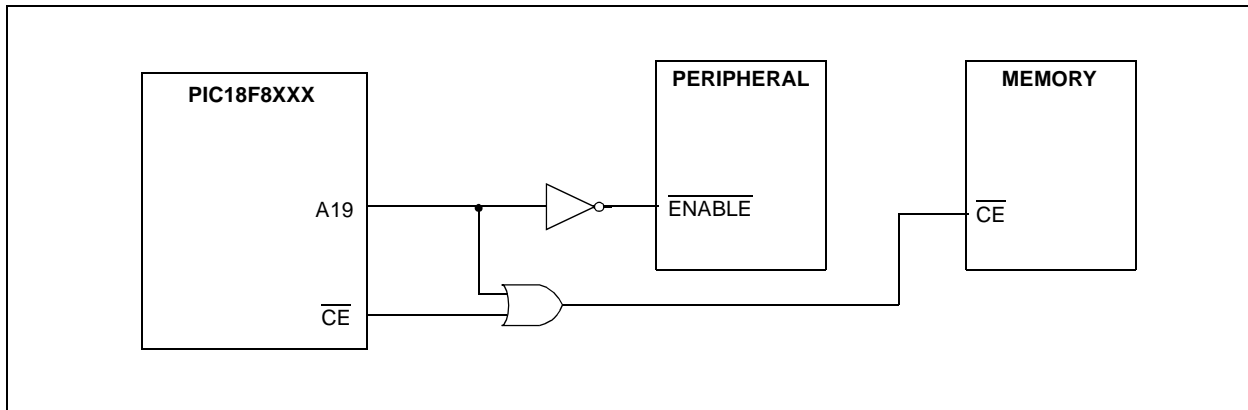
The PIC18F8XXX implements one chip enable (\overline{CE}) signal. This line is enabled when access occurs beyond the internal limit of the PIC microcontroller Operating mode. This ensures that external memories are disabled while the PIC microcontroller is executing internally. A power sensitive application can benefit from this if the memory device uses the inactive enable line to enter a power-down state.

There may be instances where an application requires multiple memory mapped I/O. Because of the availability of only one chip enable line, additional hardware may be needed. In the case of one external RAM space and only one memory mapped I/O, a minimal hardware solution is suggested by the following implementation.

A Memory Mapped Peripheral Solution

If external RAM space encompasses only address space below the 1-Mbyte boundary, a single address line (A19) may be used to enable the memory mapped I/O. An inverter is needed between A19 and the memory mapped I/O enable line. An OR gate will be needed to decode \overline{CE} for addresses below A19. The remaining address, data, and control lines can be connected in the standard fashion. Any table reads and table writes above 1 Mbyte will only access the memory mapped I/O.

FIGURE 15: MEMORY MAPPED PERIPHERAL SCHEMATIC



SUMMARY

The PIC18F8XXX family greatly enhances the capability of any application through access to external devices. Large amounts of code and data storage become available using the External Memory Interface. With minimal amounts of effort and cost, most any 8-bit or 16-bit memory device can complement your PIC microcontroller based application.

AN869

NOTES:

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, dsPIC, KEELoQ, MPLAB, PIC, PICmicro, PICSTART, PRO MATE and PowerSmart are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


FilterLab, microID, MXDEV, MXLAB, PICMASTER, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

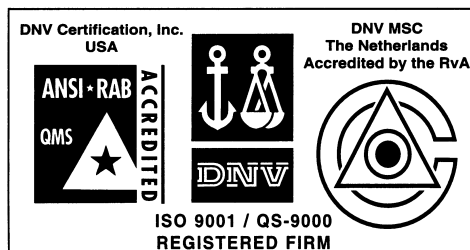
Accuron, Application Maestro, dsPICDEM, dsPICDEM.net, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, PICC, PICkit, PICDEM, PICDEM.net, PowerCal, PowerInfo, PowerMate, PowerTool, rLAB, rPIC, Select Mode, SmartSensor, SmartShunt, SmartTel and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2003, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELoQ® code hopping devices, Serial EEPROMs, microperipherals, non-volatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.



WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: <http://www.microchip.com>

Atlanta

3780 Mansell Road, Suite 130
Alpharetta, GA 30022
Tel: 770-640-0034 Fax: 770-640-0307

Boston

2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago

333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas

4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit

Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo

2767 S. Albright Road
Kokomo, IN 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles

18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

Phoenix

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-4338

San Jose

Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto

6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia

Microchip Technology Australia Pty Ltd
Marketing Support Division
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing

Microchip Technology Consulting (Shanghai)
Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu

Microchip Technology Consulting (Shanghai)
Co., Ltd., Chengdu Liaison Office
Rm. 2401-2402, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-86766200 Fax: 86-28-86766599

China - Fuzhou

Microchip Technology Consulting (Shanghai)
Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Hong Kong SAR

Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

China - Shanghai

Microchip Technology Consulting (Shanghai)
Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen

Microchip Technology Consulting (Shanghai)
Co., Ltd., Shenzhen Liaison Office
Rm. 1812, 18/F, Building A, United Plaza
No. 5022 Binhe Road, Futian District
Shenzhen 518033, China
Tel: 86-755-82901380 Fax: 86-755-8295-1393

China - Qingdao

Rm. B505A, Fullhope Plaza,
No. 12 Hong Kong Central Rd.
Qingdao 266071, China
Tel: 86-532-5027355 Fax: 86-532-5027205

India

Microchip Technology Inc.
India Liaison Office
Marketing Support Division
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaughnessy Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan

Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471-6166 Fax: 81-45-471-6122

Korea

Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5932 or
82-2-558-5934

Singapore

Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-6334-8870 Fax: 65-6334-8850

Taiwan

Microchip Technology (Barbados) Inc.,
Taiwan Branch
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Austria

Microchip Technology Austria GmbH
Durisolstrasse 2
A-4600 Wels
Austria
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

Denmark

Microchip Technology Nordic ApS
Regus Business Centre
Lautrup høj 1-3
Ballerup DK-2750 Denmark
Tel: 45-4420-9895 Fax: 45-4420-9910

France

Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - 1er Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany

Microchip Technology GmbH
Steinheilstrasse 10
D-85737 Ismaning, Germany
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy

Microchip Technology SRL
Via Quasimodo, 12
20025 Legnano (MI)
Milan, Italy
Tel: 39-0331-742611 Fax: 39-0331-466781

Netherlands

Microchip Technology Netherlands
P. A. De Biesbosch 14
NL-5152 SC Drunen, Netherlands
Tel: 31-416-690399 Fax: 31-416-690340

United Kingdom

Microchip Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44-118-921-5869 Fax: 44-118-921-5820

07/10/03