Using External Data Memory with PIC24F/24H/dsPIC33F Devices

Author: Vidyadhar Vivekananda
Microchip Technology Inc.

INTRODUCTION

This application note describes the methodology to use the Parallel Master Port (PMP) module to interface with external data memory; either external Flash or external RAM. This application note also lists the APIs and describes how to implement different types of interfaces.

Using the PMP module, the memory devices with 64K locations (Kbytes or K words) can be interfaced with no extra I/Os and software. This application note describes how to interface the memory devices with more than 64K locations using some I/O pins and provides the required APIs.

This application note describes the following topics:

- “External Data Memory Interface Overview”
- “Functional Implementation”
- “Expansion Of External Memory”
- “Reference Code”

EXTERNAL DATA MEMORY INTERFACE OVERVIEW

The PIC24F/24H/dsPIC33F architecture supports up to 64 Kbytes of internal data memory. If internal memory is insufficient, the external memory can be used. But, this external memory cannot be directly accessed by the CPU of the controller. The CPU can access through the PMP module.

This section describes the topics:

- Signals Required for Interfacing Memory Devices
- Signals Generated by the PMP Module
- Registers Associated with the PMP Module

Signals Required for Interfacing Memory Devices

Table 1 provides the signals required to interface different types of memory devices.

TABLE 1: TYPICAL MEMORY DEVICE INTERFACE CONNECTIONS

Pin Name	Function
Address Lines (A0 to An)	‘n’ number of lines are required to address all the memory locations in a 2^n Kbytes/K words memory device.
Data Lines (I/O 0 to 7 or I/O 0 to 15)	8 or 16 Data Lines are required to read/write the data in a byte or word memory device.
Chip Enable ($\overline{CE}$)	One Chip Enable signal for each memory device.
Write Enable ($\overline{WE}$)	One Write Enable signal, which should go active whenever data is to be written into the memory device.
Output Enable ($\overline{OE}$)	One Output Enable signal, which should go active whenever data is to be read from the memory device.
Byte Enable (A-1) and Word/$\overline{Byte}$	One Byte Enable signal and one Word/$\overline{Byte}$ signal, if the memory device is a 16-bit device and supports both Word and Byte modes.

Signals Generated by the PMP Module

The PMP module enables interfacing with many types of parallel devices. The module can be configured as either a master or as a slave.

There are mainly two ways of interfacing read and write signals:

- Read and write signals generated on two different pins (most memory devices use this type of interface).
- Read and write signals generated on the same pin with separate enable signals.

The PMP module in Master mode allows selection of different wait states to suit the electrical characteristics of a particular memory device

The signals used to interface with the memory devices are the address bus, data bus, read signal, write signal, chip select (optional), address latch signal (if required) and byte enable (in case of 16-bit data).

ADDRESS LINES

PMA0 to PMA15 (up to 16 address lines are available):

- PMA14 pin is multiplexed with PMCS1 pin.
- PMA15 pin is multiplexed with PMCS2 pin.
- Up to 64K locations can be accessed when Chip Select mode is not selected.
- Up to 32K locations of memory can be accessed when only one Chip Select mode is selected.
- Up to 32K locations (i.e., 16K locations x 2) of memory can be accessed when two Chip Select modes are selected.

DATA LINES

PMD0 to PMD7 (8 data lines):

- In 8-bit operation, 8-bit data is transmitted/received through these lines.
- In 16-bit operation, 16-bit data is divided into the Least Significant Byte (LSB) and Most Significant Byte (MSB). First the LSB is transmitted/received through these lines, and then the MSB.

CONTROL LINES

- PMCS1 and PMCS2 (up to two chip select lines)

These lines are multiplexed with PMA14 and PMA15. If chip select signals are selected, the address lines are necessarily reduced.

- PMWR can be used as a write line or an enable signal. To interface with a memory device, it should be used as a write line.
- PMRD can be used as a read line or a read/write line. To interface with a memory device, it should be used as a read line.
- PMBE is a byte enable line, used during 16-bit data operation. It goes active for MSB and inactive for LSB.
- PMALL and PMALH address latch lines are required only when the address bus is multiplexed with the data bus. There are two methods of multiplexing:
  - Multiplexing only the lower 8-bit address lines with 8-bit data lines. In this method, PMALL is generated on the PMA0 line. This can be used to latch the lower byte of the address.
  - Multiplexing both the lower 8-bit and the higher 8-bit address lines with 8-bit data lines. In this method, the PMA0 becomes PMALL and PMA1 becomes PMALH. PMALH is used to latch the higher byte of the address.

Figure 1 illustrates signals generated by the PMP module that are useful when interfacing with a memory device.

© 2008 Microchip Technology Inc.

FIGURE 1: MEMORY INTERFACE PMP PINS

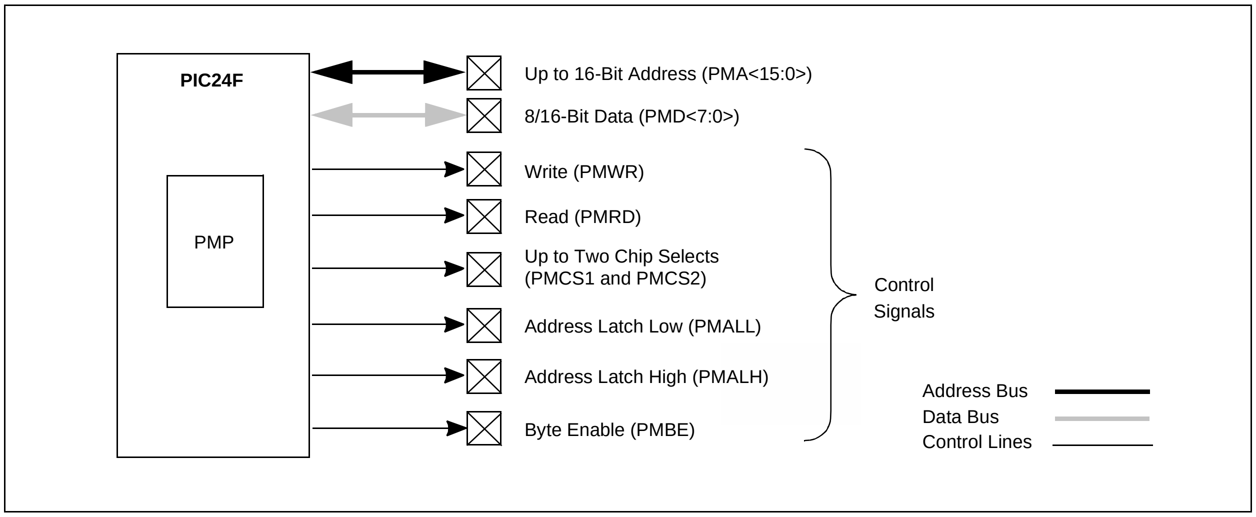

PIC24F
PMP
Up to 16-Bit Address (PMA<15:0>)
8/16-Bit Data (PMD<7:0>)
Write (PMWR)
Read (PMRD)
Up to Two Chip Selects (PMCS1 and PMCS2)
Address Latch Low (PMALL)
Address Latch High (PMALH)
Byte Enable (PMBE)
Control Signals
Address Bus
Data Bus
Control Lines

TABLE 2: MEMORY INTERFACE PMP PINS

Memory Device Pins	Pins Associated with PMP Module
Address Lines (A0 to An)	PMA0 to PMAn (up to PMA15) PMALL and PMALH (in case address is multiplexed with data)
Data Lines (I/O 0 to 1/O 7 or I/O 15)	PMD0 to PMD7
Chip Enable ($\overline{\text{CE}}$)	PMCS2 and PMCS1
Write Enable ($\overline{\text{WE}}$)	PMWR
Output Enable ($\overline{\text{OE}}$)	PMRD
Byte Enable (A-1)	PMBE

Registers Associated with the PMP Module

The following registers are associated with the PMP module in Master mode:

- PMCON – Parallel Master Port Control register
- PMMODE – Parallel Master Port Mode Selection register
- PMAEN – Parallel Master Port Address Enable register
- PMADDR – Parallel Master Port Address register
- PMDIN1 – Parallel Master Port Data register

PMCON REGISTER

The PMCON register controls these PMP functions:

- Enables PMP module
- Selects/deselects PMP module in Idle mode
- Selects different modes for data address multiplexing
- Enables or disables byte enable signal (PMBE) (byte enable signal is used only in 16-Bit Data mode)
- Enables write signal (PMWR)
- Enables read signal (PMRD)
- Selects a chip select signal or higher address lines
- Selects polarity of the address latch signals, PMALL and PMALH, used when address and data lines are multiplexed. (The signal polarity is the state of that signal when it is active; the signal will have the opposite state when it is Idle.)
- Selects polarity of Chip Select 2 signal (PMCS2) when Chip Select 2 is used
- Selects polarity of Chip Select 1 signal (PMCS1) when Chip Select 1 is used
- Selects polarity of byte enable signal (PMBE) when 16-Bit Data mode is opted
- Selects polarity of write signal (PMWR)
- Selects polarity of read signal (PMRD)

PMMODE REGISTER

The PMMODE register controls these PMP functions:

- Determines the status, whether the PMP module is busy or not
- Selects when to set the interrupt flag
- Selects either to auto-increment or decrement address
- Selects 8-Bit or 16-Bit Data mode
- Selects between two Master and two Slave modes. (For a memory interface, select Master mode with separate read and write signals.)
- Selects different Wait periods (For more information, refer to the **"Wait States and Their Usage"** section.)

PMAEN REGISTER

The PMAEN register controls these PMP functions:

- Enables Chip Select 2/Address 15 (PMCS2/PMA15) port
- Enables Chip Select 1/Address 14 (PMCS1/PMA14) port
- Enables Address 13:2 (PMA<13:2>) ports
- Enables Address 1/Address High Latch (PMA1/PMALH) port
- Enables Address 0/Address Low Latch (PMA0/PMALL) port

PMADDR REGISTER

This register holds the address of the memory location to be accessed. This either remains unchanged, increments or decrements on data access as per the PMMODE configuration.

PMDIN1 REGISTER

This register holds the data read while reading, and holds the data to be written while writing. When the PMP is configured in 8-Bit Data mode, only the LSB of the PMDIN1 register is valid.

Note:	For more information on these registers, refer to the specific device data sheet.

Wait States and Their Usage

All memory devices have setup time, hold time and control signal width specifications. To meet these specifications, all three Wait states can be configured in the PMP module.

Setup time can be configured between 1 TCY and 4 TCY, but the setup time is independently configurable only when the address lines and data lines are not multiplexed. When address lines and data lines are multiplexed, setup time and the width of the address phase on the data bus both are configured using a common set of bits.

Hold time can also be configured between 1 TCY and 4 TCY.

The control signals (read and write) pulse width (control signal width) can be configured between 1 TCY and 15 TCY.

When Wait states are disabled, setup time is set to 1/4 TCY, hold time is set to 1/4 TCY, control signal width is set to 1/2 TCY and the address width on data lines (when address and data are multiplexed) is set to 1 TCY.

Figure 14 and Figure 15 depict the effect of using the Wait states.

© 2008 Microchip Technology Inc.

FUNCTIONAL IMPLEMENTATION

This section describes the interfaces implemented in this application note. The following topics are described:

- Interfacing a 64K x 8-bit memory device (with chip select permanently activated)
- Interfacing a 32K x 8-bit memory device
- Interfacing two 16K x 8-bit memory devices
- Interfacing a 32K x 16-bit word memory device

Interfacing a 64K x 8-Bit Memory Device (with Chip Enable Permanently Activated)

To interface a 64K x 8-bit memory device, 16 address lines are required. The PMP module can generate up to a 16-bit address (16-bit address is available only when chip select is not enabled). Figure 2 illustrates how a 64K x 8-bit memory device would be connected.

Figure 4 provides a timing diagram (PMCS2 should be ignored as chip enable is permanently activated). It can be observed that each read and write operation takes one instruction cycle.

Table 3 provides the register configurations for associated registers.

To use the APIs provided with this application note for this configuration, uncomment the following lines in the `MIDefn.h` file:

```
#define Single64KBChipNoCS
#define NoAddressDataMux
```

FIGURE 2: BLOCK DIAGRAM OF 64K x 8-BIT MEMORY DEVICE INTERFACE

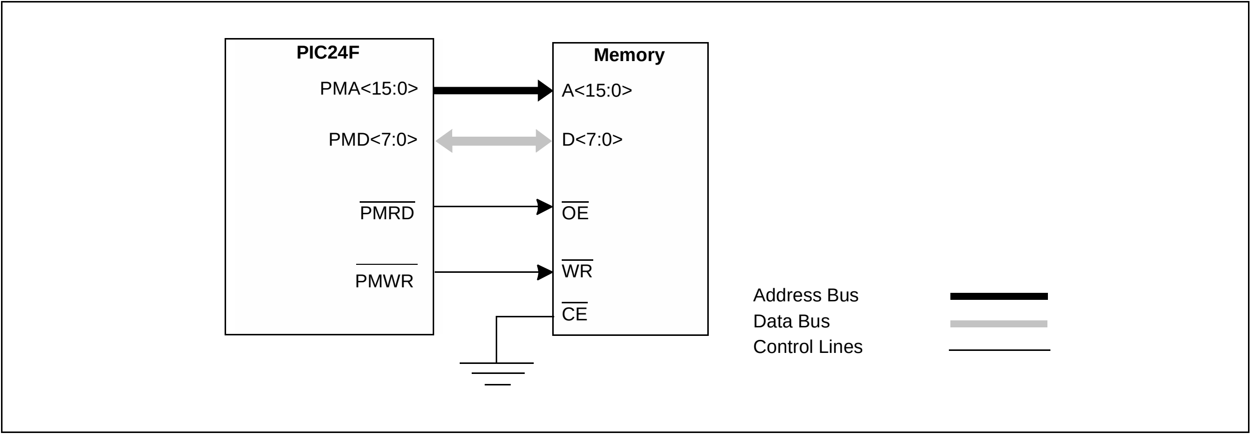


TABLE 3: CONFIGURATION OF PMP REGISTERS FOR INTERFACING 64K x 8-BIT MEMORY DEVICE USING 16 ADDRESS LINES AND CHIP ENABLE PERMANENTLY ACTIVATED

Register	Value	Description
PMCON	`10x0001100xxxx00`	• PMP module enabled • Select to run/stop in Idle mode • Address and data on separate pins • PMBE port disabled • PMWR port enabled • PMRD port enabled • PMCS1 and PMCS2 functioning as PMA15 and PMA14 • Address latch signal polarity is irrelevant (no address latch signals used) • PMCS2 polarity is irrelevant (no PMCS2 used) • PMCS1 polarity is irrelevant (no PMCS1 used) • Byte enable is irrelevant (no byte enable used) • Write strobe polarity, active-low • Read strobe polarity, active-low
PMMOD	`00xxx010xxxxxxxx`	• Busy status bit • Whether to get interrupted on read/write or not • Auto-increment/decrement or no auto-change of address • 8-Bit Data mode • Master mode with separate read and write strobes • Required data setup time • Required read/write strobe width • Required data hold time after strobe
PMAEN	`1111111111111111`	Enable as many address line ports as required
PMADDR	`xxxxxxxxxxxxxxxx`	Address register
PMDIN1	N/A	Data register

© 2008 Microchip Technology Inc.

Interfacing a 32K x 8-Bit Memory Device

While interfacing a 64K x 8-bit memory device, the chip enable pin of the memory device was connected to ground. If the chip select generated by the PMP is used to connect to the chip enable of the memory device, then only 15 address lines will be left, and hence, only 32K x 8-bit memory device can be interfaced.

In this interface all the three multiplexing modes are described. These three modes can also be used during any of the other interfaces described in this application note. The three multiplexing modes are:

- No Multiplex mode (address data demultiplexed)
- Partially Multiplexed mode (lower address multiplexed with data)
- Fully Multiplexed mode (both lower and higher bytes of address multiplexed with data)

DEMULTIPLEXED MODE

In this mode, all address and data lines have separate pins. Figure 3 illustrates the interface between a 32K x 8-bit memory device and a PIC24F device. Figure 4 provides a timing diagram. In Demultiplex mode, each read and write operation takes one instruction cycle.

Table 4 provides the register configurations for associated registers.

To use the APIs provided with this application note for this configuration, uncomment the following lines in the `MIDefn.h` file:

```
#define Single32KBChip
#define NoAddressDataMux
```

FIGURE 3: 32K x 8-BIT MEMORY DEVICE INTERFACE (DEMULTIPLEXED MODE)

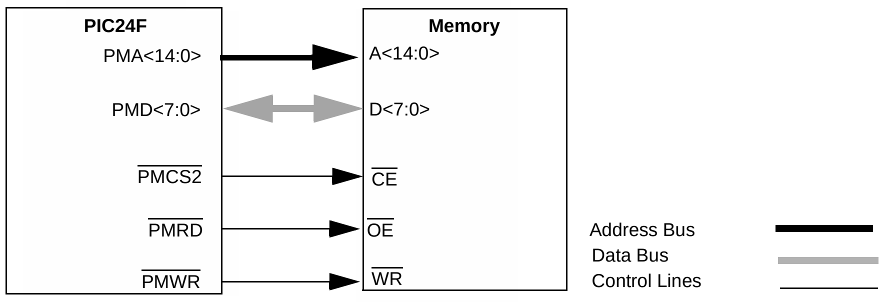
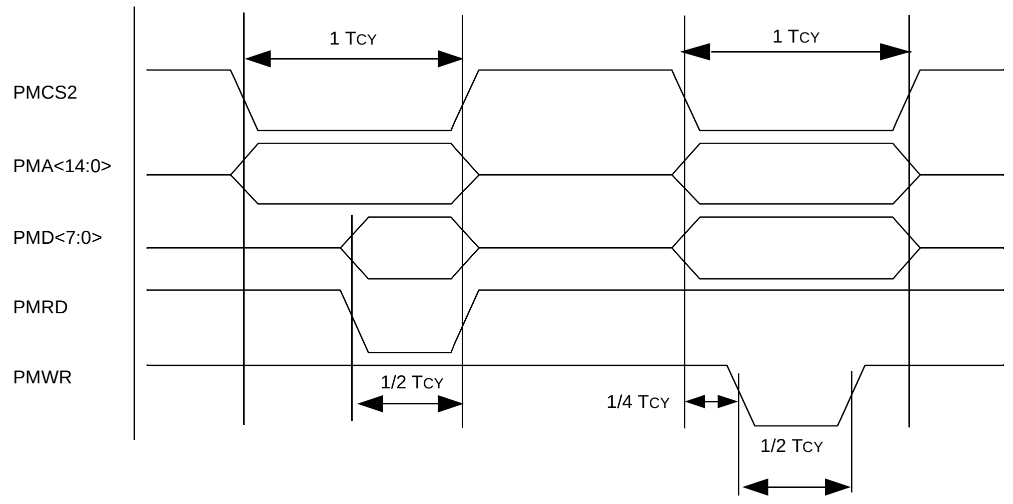


TABLE 4: CONFIGURATION OF PMP REGISTERS FOR INTERFACING A 32K x 8-BIT MEMORY DEVICE (DEMULTIPLEXED MODE)

Register	Value	Description
PMCON	`10x0001101x0xx00`	• PMP module enabled • Select to run/stop in Idle mode • Address and data on separate pins • PMBE port disabled • PMWR port enabled • PMRD port enabled • PMCS1 functioning as PMA14 and PMCS2 as chip select • Address latch signal polarity is irrelevant (no address latch signal used) • PMCS2 polarity low • PMCS1 polarity is irrelevant (no PMCS1 used) • Byte enable polarity is irrelevant (no byte enable used) • Write strobe polarity, active-low • Read strobe polarity, active-low
PMMODE	`00xxx010xxxxxxxx`	• Busy status bit • Whether to get interrupted on read/write or not • Auto-increment/decrement or no auto-change of address • 8-Bit Data mode • Master mode with separate read and write strobes • Required data setup time • Required read/write strobe width • Required data hold time after strobe
PMAEN	`1111111111111111`	• Enable PMCS2 port • Enable as many address line ports as required
PMADDR	`1xxxxxxxxxxxxxxx`	Address register (bit 15 enables PMCS2 and bits<14:0> are address bits)
PMDIN1	N/A	Data register

FIGURE 4: READ AND WRITE TIMING WHEN ADDRESS AND DATA ARE DEMULTIPLEXED



© 2008 Microchip Technology Inc.

PARTIALLY MULTIPLEXED MODE

In Partially Multiplexed mode, the lower address byte lines are multiplexed with the PMD<7:0> pins. The higher address byte lines are on the PMA<14:8> pins. The PMA0 pin becomes the PMALL pin; this latches the lower address byte. Therefore, seven pins (PMA<7:1>) are available (free from the PMP module) for other purposes. Figure 5 illustrates the interface of a 32K x 8-bit memory device with lower address byte lines multiplexed with data lines. Figure 6 provides the timing diagram. In the Partially Multiplexed mode, each read and write operation takes two instruction cycles.

Table 5 provides the register configurations for the associated registers.

To use the APIs provided with this application note for this configuration, uncomment the following lines in the `MIDefn.h` file:

```
#define Single32KBChip
#define LowAddressDataMux
```

FIGURE 5: 32K x 8-BIT MEMORY DEVICE INTERFACE USING PARTIALLY MULTIPLEXED MODE

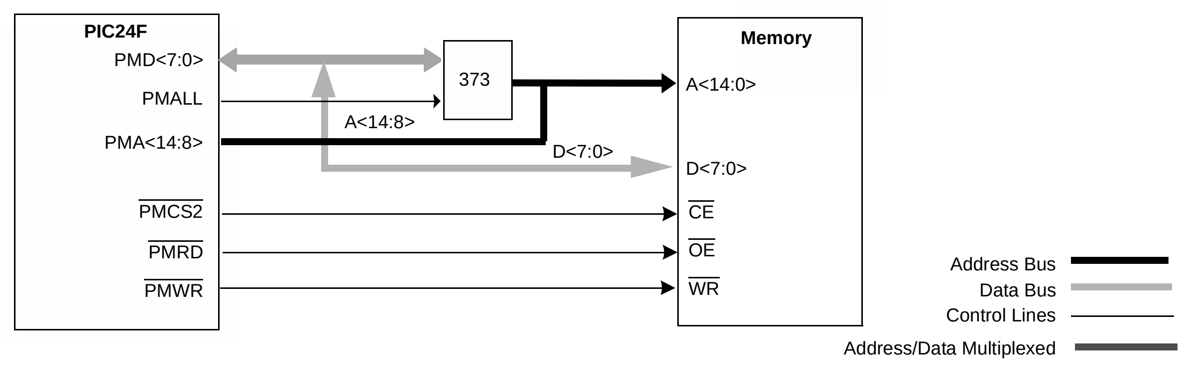
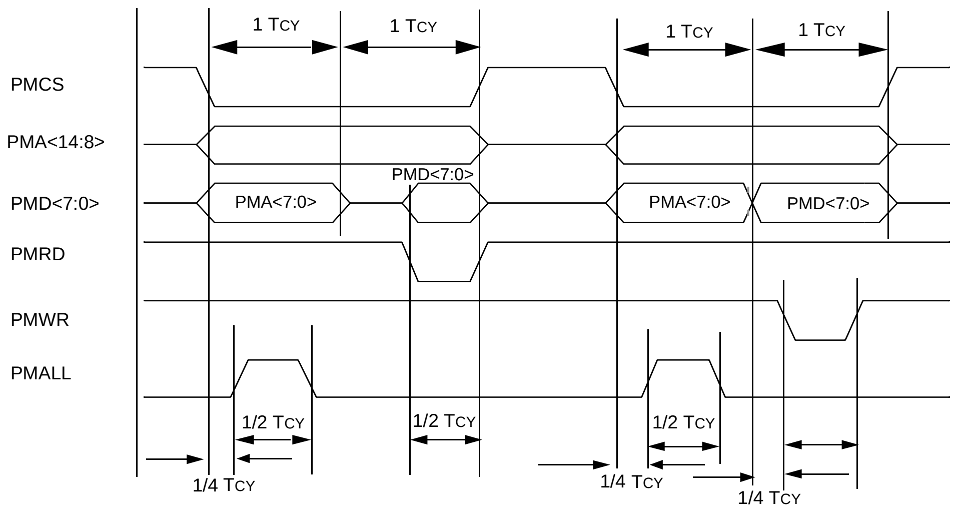


TABLE 5: CONFIGURATION OF PMP REGISTERS FOR INTERFACING A 32K x 8-BIT MEMORY DEVICE USING PARTIAL MULTIPLEXED MODE

Register	Value	Description
PMCON	`10x010110110xx00`(1)	• PMP module enabled • Select to run/stop in Idle mode • Higher address byte on separate pins and lower address byte multiplexed with data pins • PMBE port disabled • PMWR port enabled • PMRD port enabled • PMCS1 functioning as PMA14 and PMCS2 as chip select(1) • Address latch signal high (for 373 latch) • PMCS2 polarity low • PMCS1 polarity is irrelevant (no PMCS1 used) • Byte enable polarity is irrelevant (no byte enable used) • Write strobe polarity, active-low • Read strobe polarity, active-low
PMMODE	`00xxx010xxxxxxxx`	• Busy status bit • Whether to get interrupted on read/write or not • Auto-increment/decrement or no auto-change of address • 8-Bit Data mode • Master mode with separate read and write strobes • Required width of the address bus on data lines • Required read/write strobe width • Required data hold time after strobe
PMAEN	`1111111100000001`	• Enable PMCS2 port • Enable as many higher address line ports as required • Enable PMALL port
PMADDR	`1xxxxxxxxxxxxxxx`(1)	Address register (bit 15 enables PMCS2 and bits<14:0> are address bits)
PMDIN1	N/A	Data register

Note 1: If chip select is not used, PMCON = `10x01011001xxx00` and PMADDR = `xxxxxxxxxxxxxxxx`.

FIGURE 6: READ AND WRITE TIMING WHEN ADDRESS AND DATA LINES ARE PARTIALLY MULTIPLEXED



© 2008 Microchip Technology Inc.

FULLY MULTIPLEXED MODE

In fully multiplexed mode, the lower and the higher address byte lines are multiplexed with PMD<7:0>. The PMA0 pin becomes the PMALL pin; this latches the lower address byte. The PMA1 pin becomes the PMALH pin; this latches the higher address byte. Therefore, 13 pins (PMA<14:2>) are available (free from the PMP module) for other purposes. Figure 7 illustrates the interface of a 32K x 8-bit memory device, with the lower address byte lines and the higher address byte lines, multiplexed with data lines. Figure 8 provides the timing diagram. In this mode, each read and write takes three instruction cycles.

Table 6 provides the register configurations for associated registers.

To use the APIs provided with this application note for this configuration, uncomment the following lines in the `MIDefn.h` file:

```
#define Single32KBChip
#define FullAddressDataMux
```

FIGURE 7: 32K x 8-BIT MEMORY DEVICE INTERFACE USING FULLY MULTIPLEXED MODE

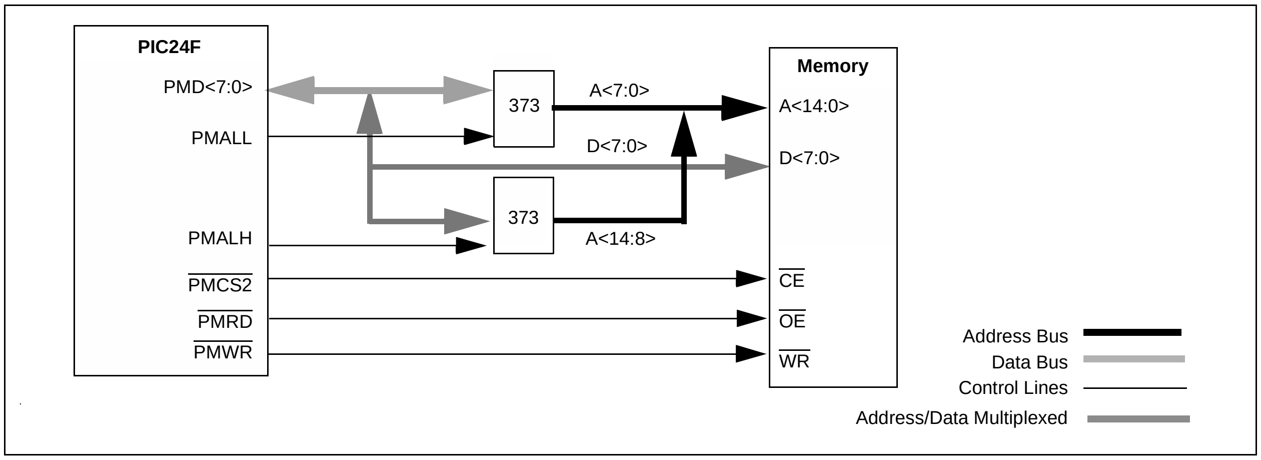


FIGURE 8: READ AND WRITE TIMING WHEN ADDRESSES ARE FULLY MULTIPLEXED WITH DATA

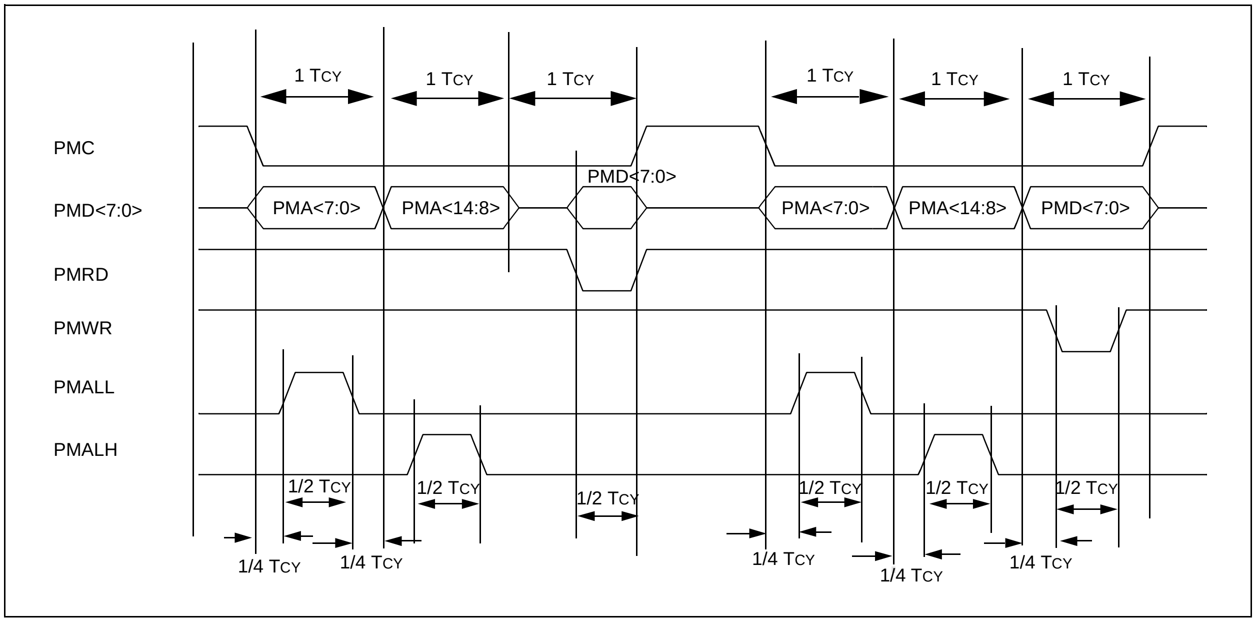


TABLE 6: CONFIGURATION OF PMP REGISTERS FOR INTERFACING A 32K x 8-BIT MEMORY DEVICE USING FULLY MULTIPLEXED MODE

Register	Value	Description
PMCON	`10x100110110xx00`[(1)]	• PMP module enabled • Select to run/stop in Idle mode • Lower address and higher address multiplexed with data pins • PMBE port disabled • PMWR port enabled • PMRD port enabled • PMCS1 functioning as PMA14 and PMCS2 as chip select[(1)] • Address latch signal polarity high (for 373 latch) • PMCS2 polarity low • PMCS1 polarity is irrelevant (no PMCS1 used) • Byte enable polarity is irrelevant (no byte enable used) • Write strobe polarity, active-low • Read strobe polarity, active-low
PMMODE	`00xxx010xxxxxxxx`	• Busy status bit • Whether to get interrupted on read/write or not • Auto-increment/decrement or no auto-change of address • 8-Bit Data mode • Master mode with separate read and write strobes • Required width of the address bus on data lines • Required read/write strobe width • Required data hold time after strobe
PMAEN	`1000000000000011`[(1)]	• Enable PMCS2 port • Enable PMALH port • Enable PMALL port
PMADDR	`1xxxxxxxxxxxxxxx`[(1)]	Address register (bit 15 enables PMCS2 and bits<14:0> are address bits)
PMDIN1	N/A	Data register

Note 1: If chip select is not used, PMCON = `10x10011001xxx00`, PMAEN = `0000000000000011` and PMADDR = `xxxxxxxxxxxxxxxx`.

© 2008 Microchip Technology Inc.

Interfacing Two 16K x 8-Bit Memory Devices

To interface two memory devices, two chip selects are required; therefore, only 14 address bits can be generated by the PMP module. In this configuration, only two memory devices (up to 16K x 8-bit) can be connected.

Figure 9 illustrates the interface of two 16-Kbyte memory devices. Figure 8 provides the timing diagram. The timing diagram illustrates only PMCS2. Similarly, when the first chip is accessed, PMCS1 becomes active instead of PMCS2.

Table 7 provides the register configurations for the associated registers.

To use the APIs provided with this application note for this configuration, uncomment the following lines in the `MIDefn.h` file:

```
#define Two16KBChips
#define FullAddressDataMux
```

FIGURE 9: INTERFACING TWO 16K x 8-BIT MEMORY DEVICES

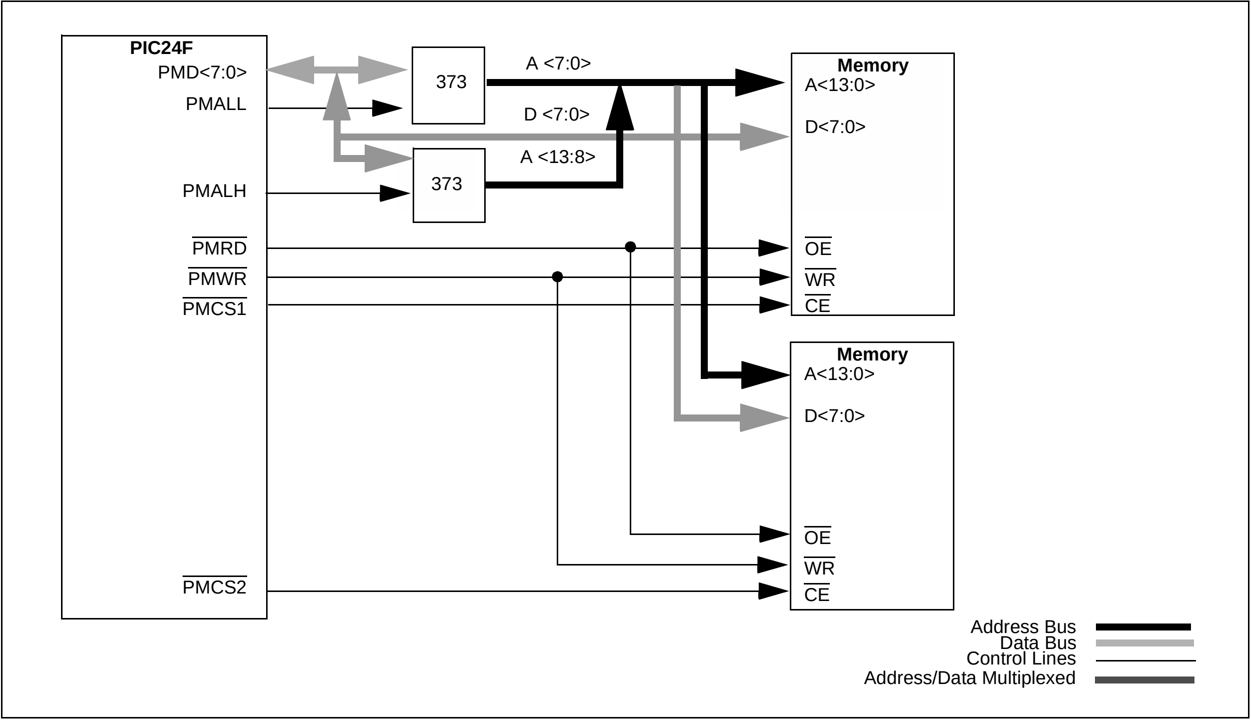


TABLE 7: CONFIGURATION OF PMP REGISTERS FOR INTERFACING TWO 16K x 8-BIT MEMORY DEVICES USING FULLY MULTIPLEXED MODE

Register	Value	Description
PMCON	`10x1001110100x00`(1,2)	• PMP module enabled • Select to run/stop in Idle mode • Address and data fully multiplexed(1,2) • PMBE port disabled • PMWR port enabled • PMRD port enabled • PMCS1 and PMCS2 as chip selects • Address latch signal polarity high(2) • PMCS2 polarity low • PMCS1 polarity low • Byte enable polarity is irrelevant (no byte enable used) • Write strobe polarity, active-low • Read strobe polarity, active-low
PMMODE	`00xxx010xxxxxxxx`	• Busy status bit • Whether to get interrupted on read/write or not • Auto-increment/decrement or no auto-change of address • 8-Bit Data mode • Master mode with separate read and write strobes • Required data setup time • Required read/write strobe width • Required data hold time after strobe
PMAEN	`1100000000000011`(1,2)	• Enable PMCS2 port • Enable PMCS1 port • Enable PMALL port • Enable PMALH port
PMADDR	`xxxxxxxxxxxxxxxx`	Address register (bit 15 enables PMCS2, bit 14 enables PMCS1 and bits<13:0> are address bits)
PMDIN1	N/A	Data register

Note 1: If partial address is multiplexed with data lines, PMCON = `10x0101110100x00` and PMAEN = `1111111100000001`.

2: If the address and data are on separate lines, PMCON = `10x0001110000x00` and PMAEN = `1111111111111111`.

© 2008 Microchip Technology Inc.

Interfacing a 32K x 16-Bit Word Memory Device

To interface a 16-bit memory device, 16 data lines are required. The PMP module has only 8 data lines. The 16-bit data is split into two 8-bit data phases, first the LSB phase and then the MSB phase. Figure 10 and Figure 11 illustrate how to interface a 32K x 16-bit memory device.

Some 16-bit memory devices support both word and byte access. These devices will have the A-1 pin, which decides the byte accessed while in Byte mode. It should be noted that we are using Byte Access mode. The PMBE pin should be connected to this pin, as illustrated in Figure 10.

If the memory device supports only Word Access mode, the connections are to be made as illustrated in Figure 11.

Figure 12 provides the timing diagram. In 16-bit mode, each read and write takes one extra instruction cycle for the same operation in 8-bit mode. Hence, in Fully Multiplexed mode with 16-bit data, each read and write takes four instruction cycles.

Table 8 provides the register configurations for the associated registers.

To use the APIs provided with this application note for this configuration, uncomment the following lines in the `MIDefn.h` file:

`#define Data16bit`

`#define HighByteEnb`, if polarity of byte enable signal should be high

`#define FullAddressDataMux`

FIGURE 10: 32K x 16-BIT MEMORY DEVICE (EXAMPLE 1)

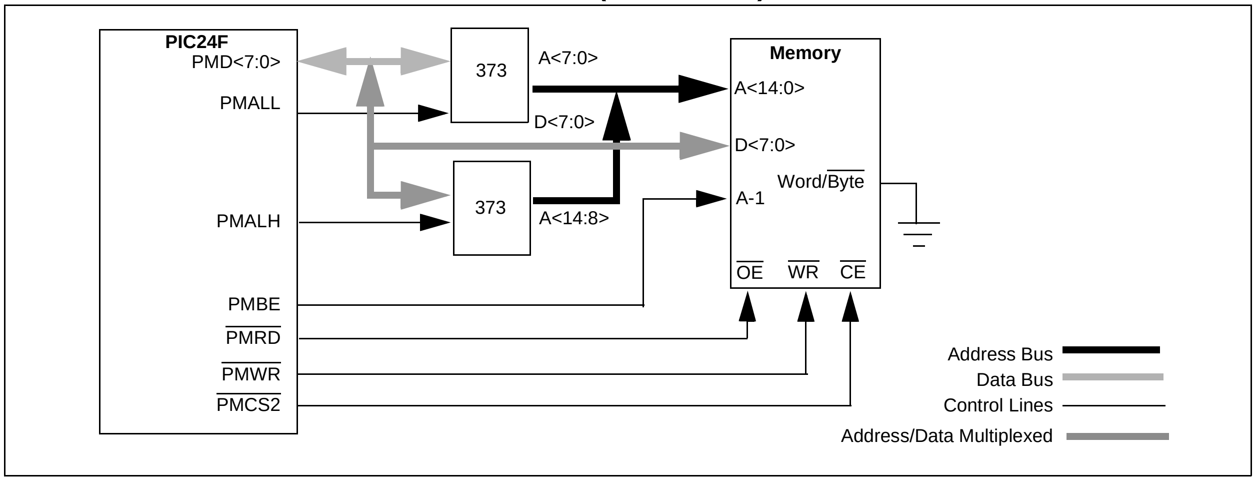


FIGURE 11: 32K x 16-BIT MEMORY DEVICE, ADDRESS AND DATA MULTIPLEXED (EXAMPLE 2)

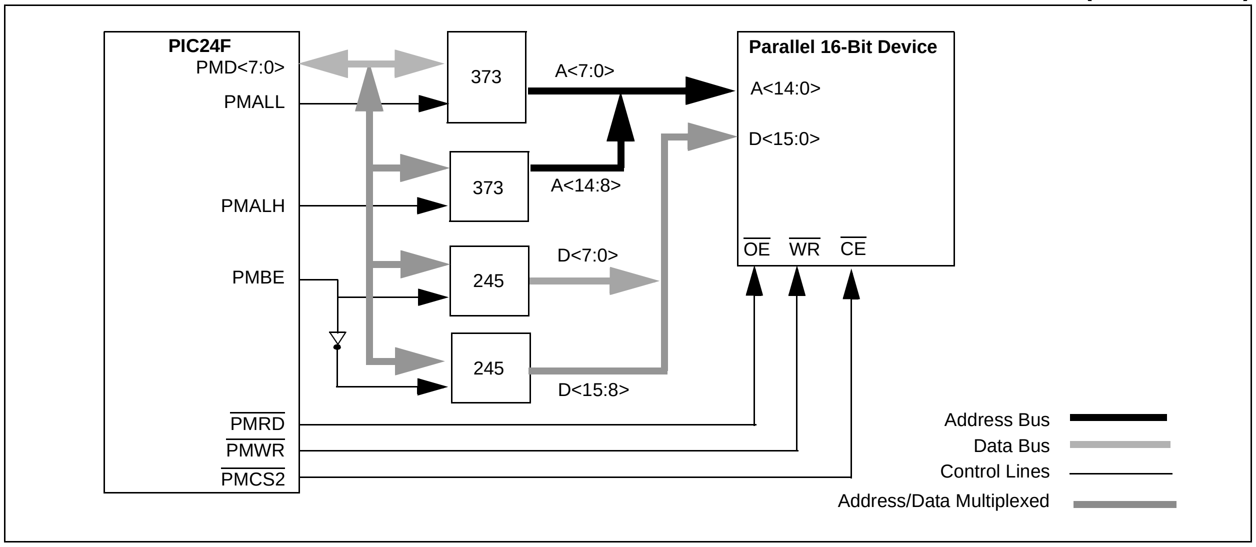


TABLE 8: CONFIGURATION OF PMP REGISTERS FOR INTERFACING A 32K x 16-BIT MEMORY DEVICE USING FULLY MULTIPLEXED MODE

Register	Value	Description
PMCON	`10x101110110x100`(1,2,3,4,5)	• PMP module enabled • Select to run/stop in Idle mode • Address and data fully multiplexed(1,2,4,5) • PMBE port enabled • PMWR port enabled • PMRD port enabled • PMCS1 functioning as PMA14 and PMCS2 as chip select(3,4,5) • Address latch signal polarity high(2,5) • PMCS2 polarity low • PMCS1 polarity is irrelevant(3,4,5) • Byte enable polarity active-high • Write strobe polarity active-low and read strobe polarity active-low
PMMODE	`00xxx110xxxxxxxx`	• Busy status bit • Get interrupted on read/write or not • Auto-increment/decrement or no auto-change of address • 16-Bit Data mode • Master mode with separate read and write strobes • Required width of the address bus on data lines • Required read/write strobe width • Required data hold time after strobe
PMAEN	`1000000000000011`(1,2,3,4,5)	• Enable PMCS2 port • Enable PMALH port • Enable PMALL port
PMADDR	`1xxxxxxxxxxxxxxx`(3,4,5)	Address register (bit 15 enables PMCS2 and bits<14:0> are address bits)
PMDIN1	N/A	Data register

Note 1: If partial address is multiplexed with data lines, PMCON = `10x011110110x100` and PMAEN = `1111111100000001` (this is for full 15-bit address).
2: If the address and data are on separate lines, PMCON = `10x001110100x100` and PMAEN = `1111111111111111` (this is for full 15-bit address).
3: If full address is multiplexed with data lines with two chip selects, PMCON = `10x1011110100100`, PMAEN = `1100000000000011` (this is for full 14-bit address) and PMADDR = `11xxxxxxxxxxxxxx`.
4: If partial address is multiplexed with data lines with two chip selects, PMCON = `10x0111110100100`, PMAEN = `1111111100000011` (this is for full 14-bit address) and PMADDR = `11xxxxxxxxxxxxxx`.
5: If the address and data are on separate lines with two chip selects, PMCON = `10x0011110000100`, PMAEN = `1111111111111111` (this is for full 14-bit address) and PMADDR = `11xxxxxxxxxxxxxx`.

© 2008 Microchip Technology Inc.

FIGURE 12: READ TIMING WHEN THE ADDRESSES ARE FULLY MULTIPLEXED WITH DATA IN 16-BIT DATA MODE

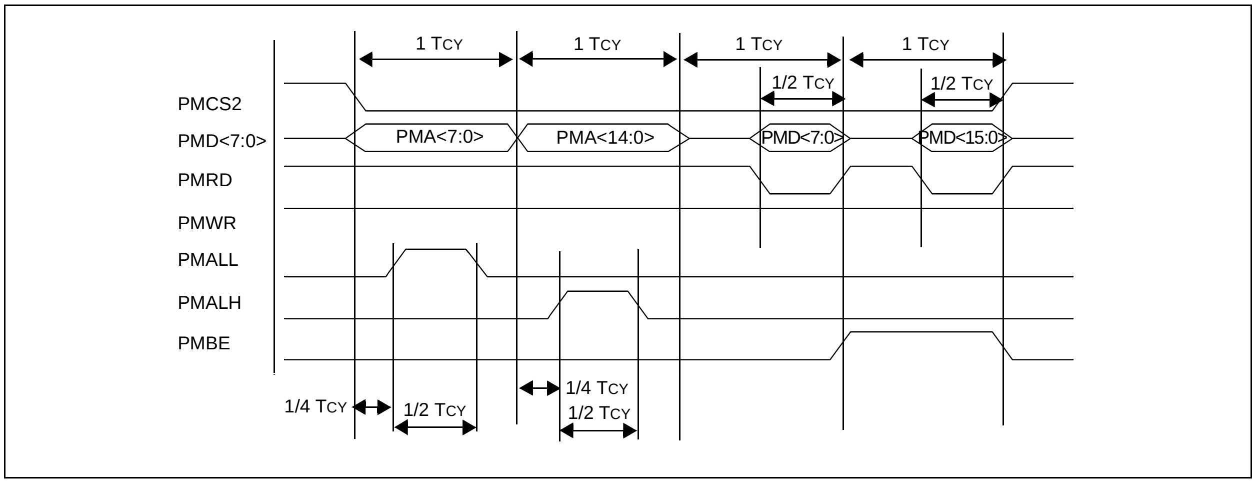


FIGURE 13: WRITE TIMING WHEN THE ADDRESSES ARE FULLY MULTIPLEXED WITH DATA IN 16-BIT DATA MODE

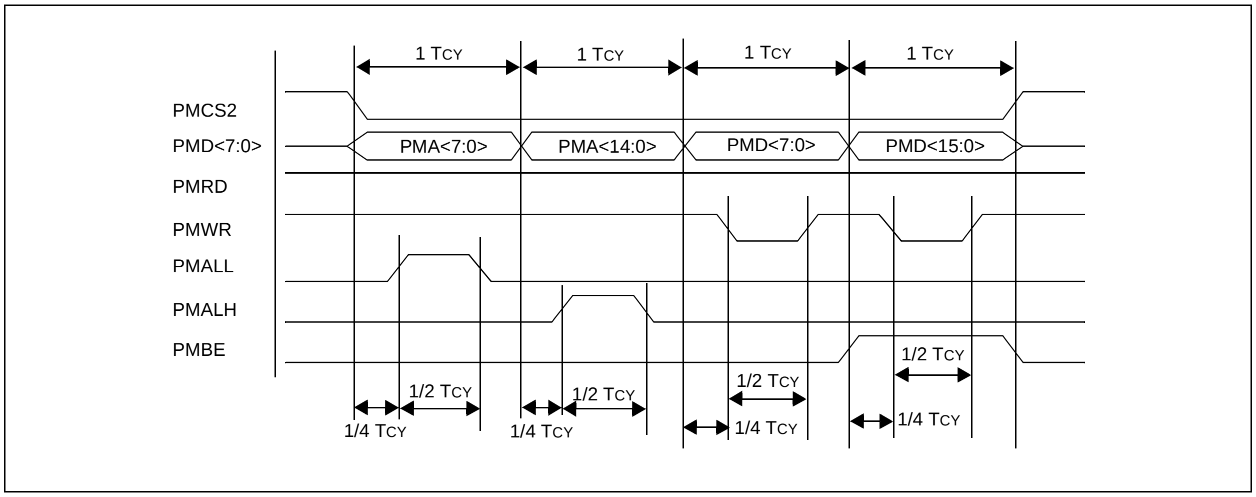
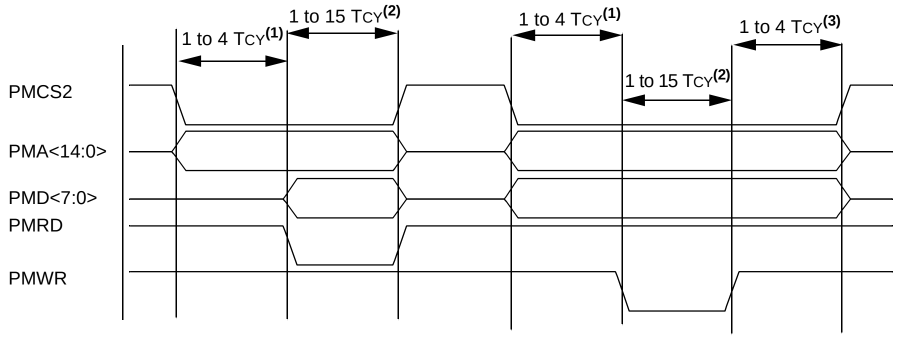
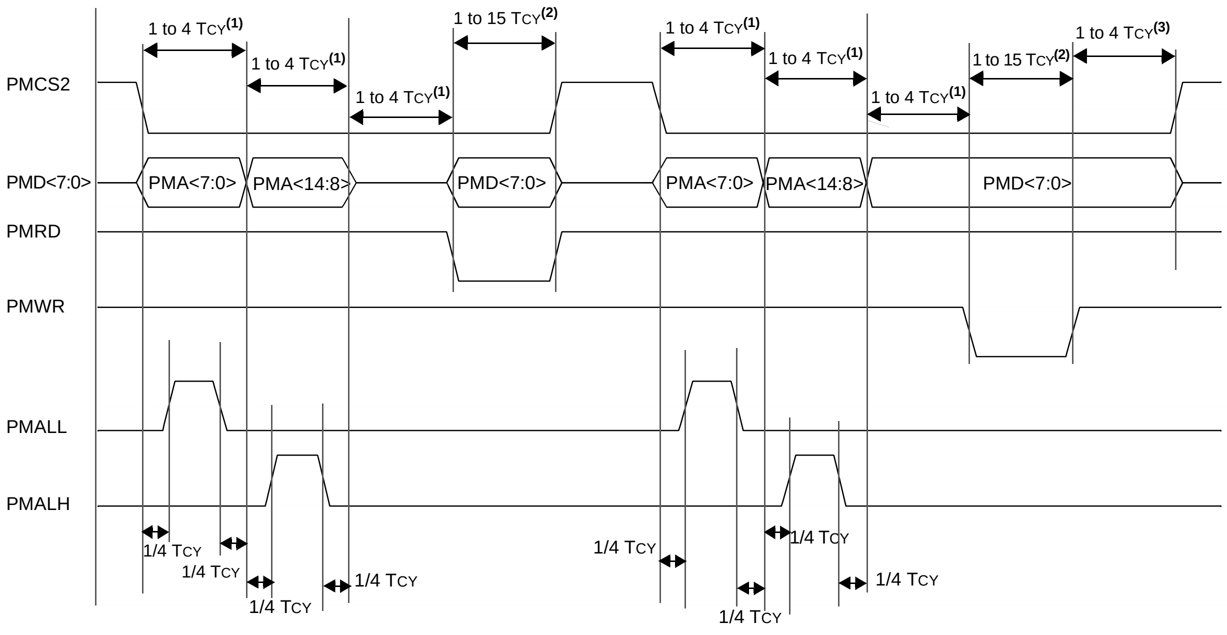


FIGURE 14: READ AND WRITE TIMING WHEN ADDRESS AND DATA ARE NOT MULTIPLEXED AND WAIT STATES ARE ENABLED



Note 1: Depends on setup time settings.
2: Depends on control signal width settings.
3: Depends on hold time settings.

FIGURE 15: READ AND WRITE TIMING WHEN ADDRESS AND DATA ARE FULLY MULTIPLEXED AND WAIT STATES ARE ENABLED



Note 1: Depends on setup time settings.
2: Depends on control signal width settings.
3: Depends on hold time settings.

© 2008 Microchip Technology Inc.

EXPANSION OF EXTERNAL MEMORY

External data memory can be expanded in two ways:

- Interfacing single memory device of sizes more than 32 Kbytes (APIs support up to 8 Mbytes)
- Interfacing multiple memory devices of 32 Kbytes each (APIs support up to 256 devices)

Interfacing Single Memory Device of More than 32 Kbytes (Up to 8 Mbytes)

In this interface, 15 address lines are generated by the PMP module. The higher address byte lines should be generated in the software using general purpose I/O pins.

To implement this:

1. Define a variable `Address_High`, which would have A15 to An.
2. Select a port to output the higher address byte and the content of the `Address_High` variable.
3. On a sequential read or write, increment the variable, `Address_High`, on every overflow of the address bus generated by the PMP module.

Figure 16 illustrates the interface of a single chip with a memory size of more than 32 Kbytes.

Table 9 provides the register configurations for the associated registers.

To use the APIs provided with this application note for this configuration, uncomment the following lines in the `MIDefn.h` file:

```
#define SingleMorethan32KBChip
```

```
#define AddressHighPort LATx
```
(where LATx can be one of LATA, LATB, LATC, LATD or LATE)

```
#define NumberofAddedAdrsLine x
```
(where x can be anything between 1 and 8)

Note: The APIs support the expansion of the interface up to 8 Mbytes of memory (generating the address lines A22:A15).

FIGURE 16: INTERFACING A SINGLE CHIP OF MORE THAN 32 Kbytes MEMORY

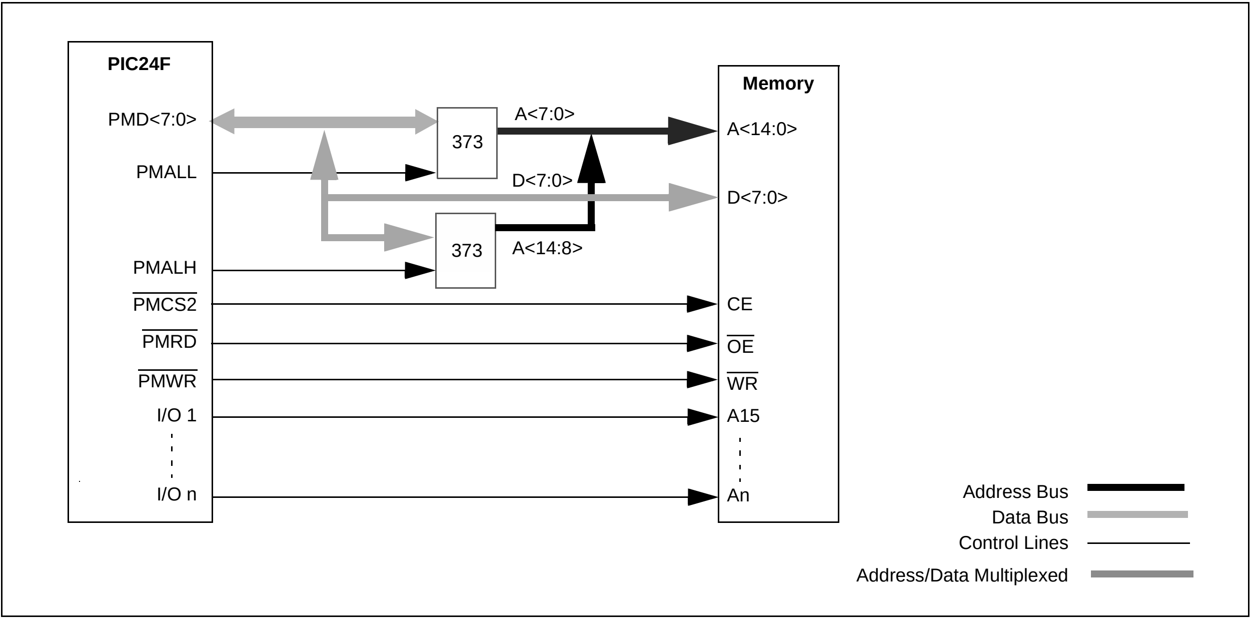


TABLE 9: CONFIGURATION OF PMP FOR INTERFACING SINGLE CHIP OF MORE THAN 32 Kbytes MEMORY

Register	Value	Description
PMCON	10x100110110xx00(1,2,3)	• PMP module enabled • Select to stop/run in Idle mode • Address and data on fully multiplexed(1,2) • PMBE port disabled • PMWR port enabled • PMRD port enabled • PMCS1 functioning as PMA14 and PMCS2 as chip select(3) • Address latch signal polarity, active-high(2) • PMCS2 polarity active-low(3) • PMCS1 polarity is irrelevant (no PMCS1 used) • Byte enable polarity is irrelevant (no byte enable used) • Write strobe polarity, active-low • Read strobe polarity, active-low
PMMODE	00xxx110xxxxxxxx	• Busy status bit • Whether to get interrupted on read/write or not • Auto-increment/decrement or no auto-change of address • 16-Bit Data mode • Master mode with separate read and write strobes • Required width of the address bus on data lines • Required read/write strobe width • Required data hold time after strobe
PMAEN	1000000000000011(1,2)	• Enable PMCS2 port • Enable PMALH port • Enable PMALL port
Address_High Higher	N/A	Address register
PMADDR	1xxxxxxxxxxxxxxx	Address register (bit 15 enables PMCS2 and bits<14:0> are address bits)
PMDIN1	N/A	Data Register

Note 1: If partial address is multiplexed with data lines, PMCON = 10x01011001xxx00 and PMAEN = 1111111100000001.
2: If the address and data are on separate lines, PMCON = 10x00011000xxx00 and PMAEN = 1111111111111111.
3: If no chip select is used, then PMCON = 10x10011001xxx00.

© 2008 Microchip Technology Inc.

Interfacing Multiple Memory Devices of 32 Kbytes Each (Up to 256 devices)

A technique to address multiple memory devices is to use a discrete demultiplexer on the chip select signal. Port I/O provide the binary encoded value to select the desired memory device.

This can be implemented by defining a variable to hold the desired demultiplexer channel. Moving this value to the Port Latch register will activate the selected memory chip.

For sequential read or write operations, the variable can be incremented on every PMP address overflow.

To implement this, perform the following steps:

1. Define a variable, `Chip_Select`, to selectively enable or disable different chips.
2. Select a port to output the contents of the variable, `Chip_Select`.
3. On a sequential read or write, increment the variable, `Chip_Select`, on every overflow of the address bus generated by the PMP module.

Figure 17 illustrates the interface of multiple chips of 32 Kbytes memory size. By using this method, the number of I/O pins required to generate the chip selects can be reduced.

Table 10 provides the register configurations for the associated registers.

To use the APIs provided with this application note for this configuration, uncomment the following lines in the `MIDefn.h` file:

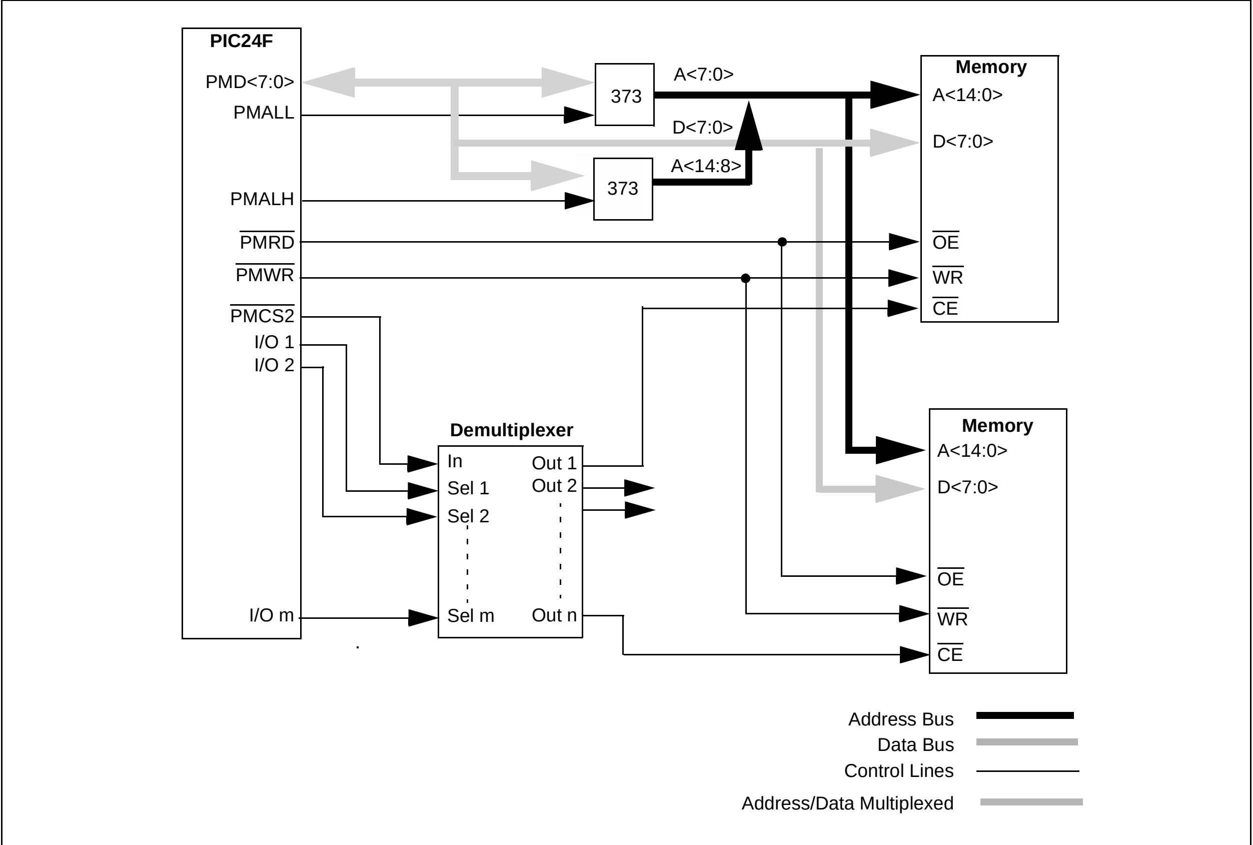
```
#define More32KBChips
```

```
#define ChipSelectPort LATx
```
(where LATx can be one of LATA, LATB, LATC, LATD or LATE)

```
#define NumberofAddedCSLine x
```
(where x can be anything between 1 and 3)

Note: The APIs provide support expansion up to 8 Mbytes of memory (generating the select signals, Sel<3:1>, for the demultiplexer).

FIGURE 17: INTERFACING MULTIPLE 32 Kbytes MEMORY DEVICES



Note: The 'm' number of select signals required to enable 'n' number of memory devices where m 'n'=2m.

TABLE 10: CONFIGURATION OF PMP REGISTERS FOR INTERFACING MULTIPLE 32 Kbytes MEMORY DEVICES

Register	Value	Description
PMCON	`10x10011010xxx00`(1,2)	• PMP module enabled • Select to stop/run in Idle mode • Address and data are fully multiplexed(1,2) • PMBE port disabled • PMWR port enabled • PMRD port enabled • PMCS1 functioning as PMA14 and PMCS2 as chip select • Address latch signal polarity active-high(2) • PMCS2 polarity is active-low • PMCS1 polarity is irrelevant (no PMCS1 used) • Byte enable polarity is irrelevant (no byte enable used) • Write strobe polarity, active-low • Read strobe polarity, active-low
PMMODE	`00xxx110xxxxxxxx`	• Busy status bit • Whether to get interrupted on read/write or not • Auto-increment/decrement or no auto-change of address • 16-Bit Data mode • Master mode with separate read and write strobes • Required data setup time • Required read/write strobe width • Required data hold time after strobe
PMAEN	`1000000000000011`(1,2)	• Enable PMCS2 port • Enable PMALH port • Enable PMALL port
PMADDR	`1xxxxxxxxxxxxxx`	Address register (bit 15 enables PMCS2 and bits<14:0> are address bits)
PMDIN1	N/A	Data register
`Chip_Select`	N/A	Current `Chip_Select` information

Note 1: If partial address is multiplexed with data lines, PMCON = `10x01011001xxx00` and PMAEN = `1111111100000001`.

2: If the address and data are on separate lines, PMCON = `10x00011000xxx00` and PMAEN = `1111111111111111`.

© 2008 Microchip Technology Inc.

REFERENCE CODE

This section describes the APIs provided as reference code to interface different types of memory devices.

To implement the interfaces described in this application note, the following APIs are provided:

- `PMPInit()`

  Initializes the PMP module per the macros defined in the `MIDefn.h` file.
- `MemByteRead()`

  Returns a byte read from the specified memory address location.
- `MemBulkRead()`

  Reads the specified length of sequential data from the specified memory location and stores it at a specified array.
- `MemByteWrite()`

  Writes the byte passed into the specified memory location.
- `MemBulkWrite()`

  Writes the specified length of data stored in an array into the memory locations from the specified address.

To use these APIs, add the `MemInterface.c`, `MemInterface.h` and `MIDefn.h` files to your project. Edit the `MIDefn.h` file as required.

The selections are:

Memory size and number of chips:

```
#define Single64KBChipNoCS
#define Single32KBChip
#define Two16KBChips
#define SingleMorethan32KBChip
#define More32KBChips
```

Select the required macros by uncommenting the macro name and comment the remaining four.

For example:

- To interface a 64-Kbyte memory chip, uncomment `#define Single64KBChipNoCS` and comment the rest.
- To interface a 32-Kbyte memory chip, uncomment `#define Single32KBChip` and comment the rest.
- To interface two 16-Kbyte memory chips, uncomment `#define Two16KBChips`, and comment the rest.
- To interface a memory chip of more than 32 Kbytes, uncomment `#define SingleMorethan32KBChip` and comment the rest.
- To interface more 32-Kbyte memory, uncomment `#define More32KBChips` and comment the rest.

Memory chip width 8-bit/16-bit:

```
#define Data16bit
```

Comment if 8-bit chip is used; uncomment if 16-bit chip is used.

`#define HighByteEnb`
(applicable only when in 16-bit mode)

Comment this if active-low byte enable signal is required.

Address/Data Multiplex mode:

```
#define AddressDataNoMux
#define LowAddressDataMux
#define FullAddressDataMux
```

Keep the required macro uncommented and comment the other two.

For example:

- To generate address and data on separate pins, uncomment `#define AddressDataNoMux` and comment the rest.
- To multiplex only the lower address byte with the data pins, uncomment `#define LowAddressDataMux` and comment the rest.
- To multiplex both the lower and higher address bytes with the data pins, uncomment `#define FullAddressDataMux` and comment the rest.

Setup time, hold time and control signal width:

`#define SetDataSetupWait_TCY x`
(where x = 1/2/3/4)

`#define SetControlWidthWait_TCY x`
(where x = 0/1/…/15)

`#define SetDataHoldWait_TCY x`

(where x = 1/2/3/4)

Select the setup time from one of four options (1 TCY to 4 TCY).
Select the hold time from one of the four options (1 TCY to 4 TCY).

To select the required setup time, write the count (number of TCY) against `#define SetDataSetupWait_TCY`.

To select the required hold time, write the count (number of TCY) against `#define SetDataHoldWait_TCY`.

Select the control signal width from one of 16 options (1 TCY to 16 TCY).

To select the required control signal width, write the count (number of TCY) against `#define SetControlWidthWait_TCY`.

Port selection for memory extension:

To use single chip higher memory device and to select the port for generating the higher address byte,
`#define AddressHighPort LATx`
(where LATx = LATA/LATB/LATC/LATD/LATE)

To specify the number of additional address lines required,
`#define NumberofAdded AdrsLine x`
(where x = anything between 1 and 8)

To use multiple 32-Kbyte devices and to select the port for generating the select signal for the demultiplexer,
`#define ChipSelectPort LATx`
(where LATx = LATA/LATB/LATC/LATD/LATE)

To specify the number of select lines required,
`#define NumberofAddedCSLine x`
(where x = anything between 1 and 3)

Table 11 provides and describes the APIs.

Note: While using a single memory device, higher than 32 Kbytes, the additional address lines (above A14) should be generated by the software on general purpose I/O pins; while using multiple memory devices of 32 Kbytes, the select signals for the demultiplexer should be generated by the software on general purpose I/O pins.

TABLE 11: APIs PROVIDED in `MemInterface.c` FILE

Function	Description	Inputs	Outputs
`PMPInit()`	Initializes the PMP module and also the port directions if required, as defined in the `MIDefn.h` file.	None	None
`MemByteRead()`	Reads a byte from a specified location.	Memory location address (unsigned long)	Read data (char)
`MemBulkRead()`	Reads a specified number of bytes starting from a specified location and saves them from a specified pointer location.	Memory location address (unsigned long) Number of bytes to be read (unsigned int) Destination pointer (unsigned char *)	None
`MemByteWrite()`	Writes a byte at a specified location.	Memory location address (unsigned long) Data (unsigned char)	None
`MemBulkWrite()`	Writes a specified number of bytes starting from a specified location.	Memory location address (unsigned long) Number of bytes to be written (unsigned int) Source Pointer (unsigned char *)	None

Note: For the flowcharts of these APIs, refer to Figure 18 through Figure 23.

© 2008 Microchip Technology Inc.

Memory Interface Application File Example

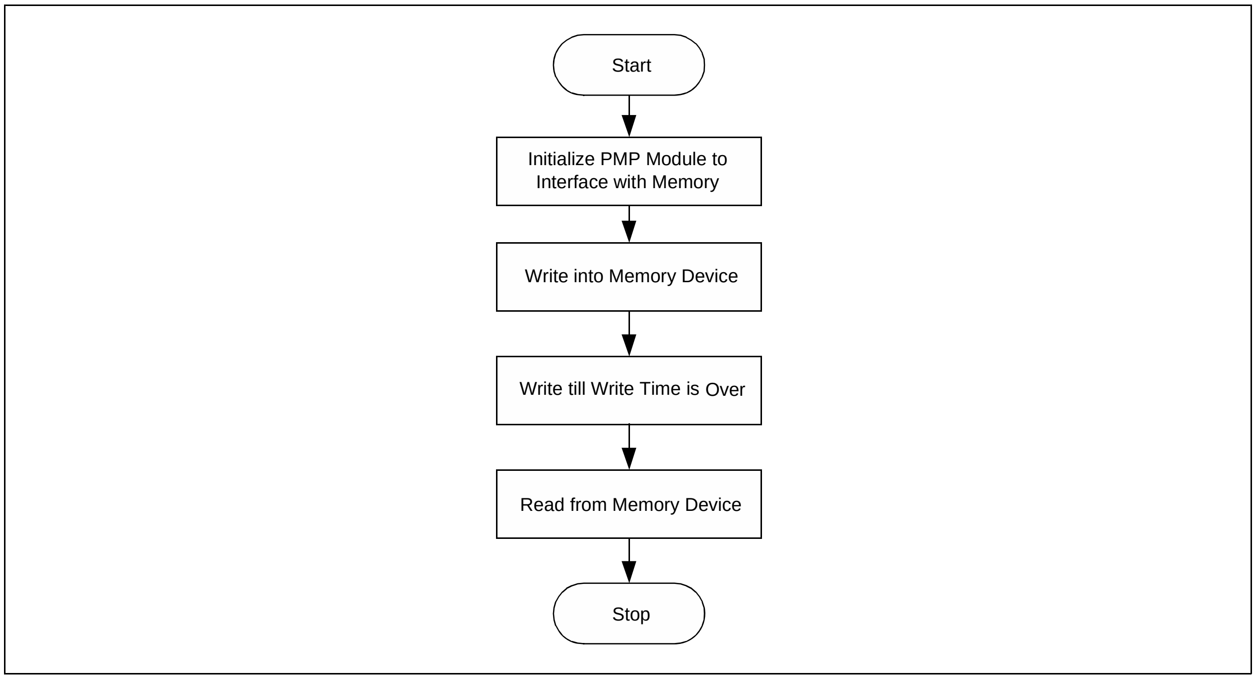
To use these APIs:

1. Write an application file.
2. Call it API.

   The APIs are in the `MemInterface.c` file.
3. Add this file to the project folder with the application file.
4. Uncomment the required definitions in the `MIDefn.h` file.
5. Include the header file, `MemInterface.h`, in the application file.

An example file, `MemIntfExample.c`, is provided along with this application note. This file describes how to use the APIs to access (read and write) the external memory device. Figure 18 illustrates the flow of this example file.

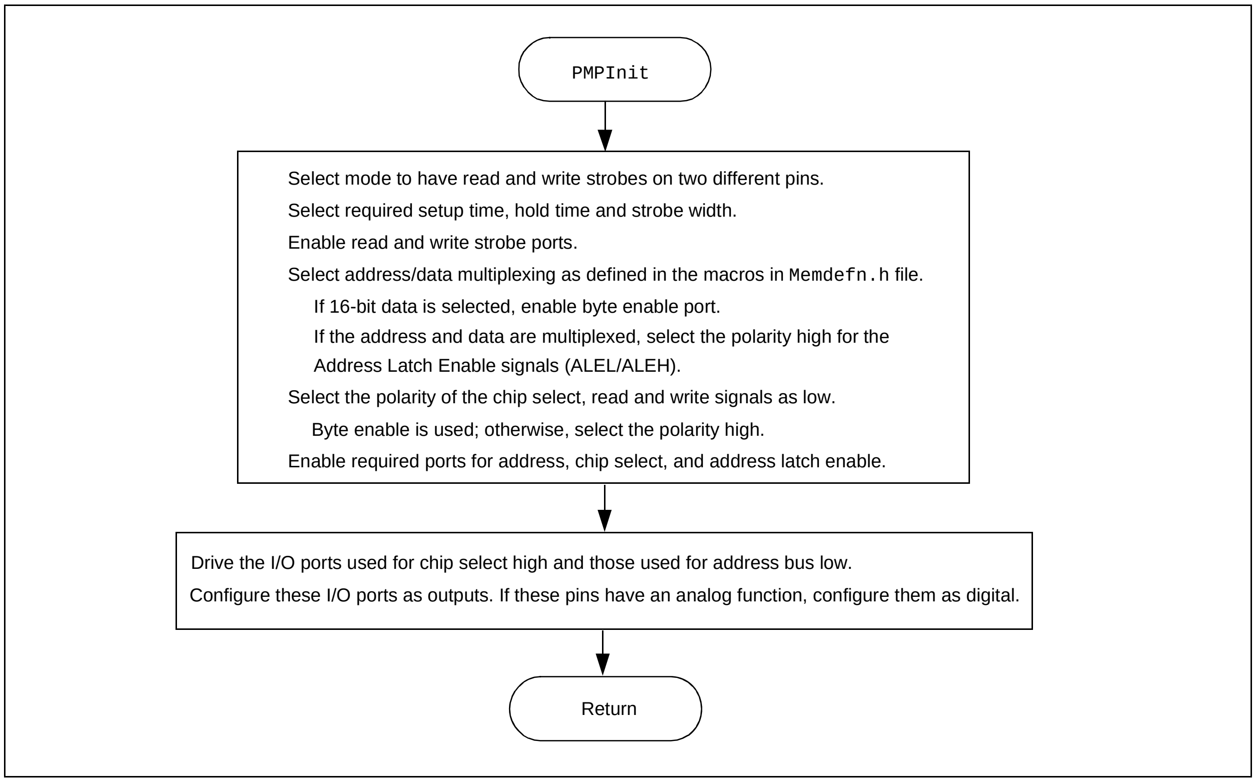
FIGURE 18: PMP MEMORY INTERFACE EXAMPLE FLOWCHART


Start
Initialize PMP Module to Interface with Memory
Write into Memory Device
Write till Write Time is Over
Read from Memory Device
Stop

`PMPInit` API

The API, `PMPInit`, initializes the PMP module as per the definitions in the file, `MIDefn.h`. There are no inputs for this API; this API returns nothing. This API takes all inputs from the `MIDefn.h` file. Figure 19 illustrates the flow of the `PMPInit` API.

FIGURE 19: `PMPInit` API FLOWCHART

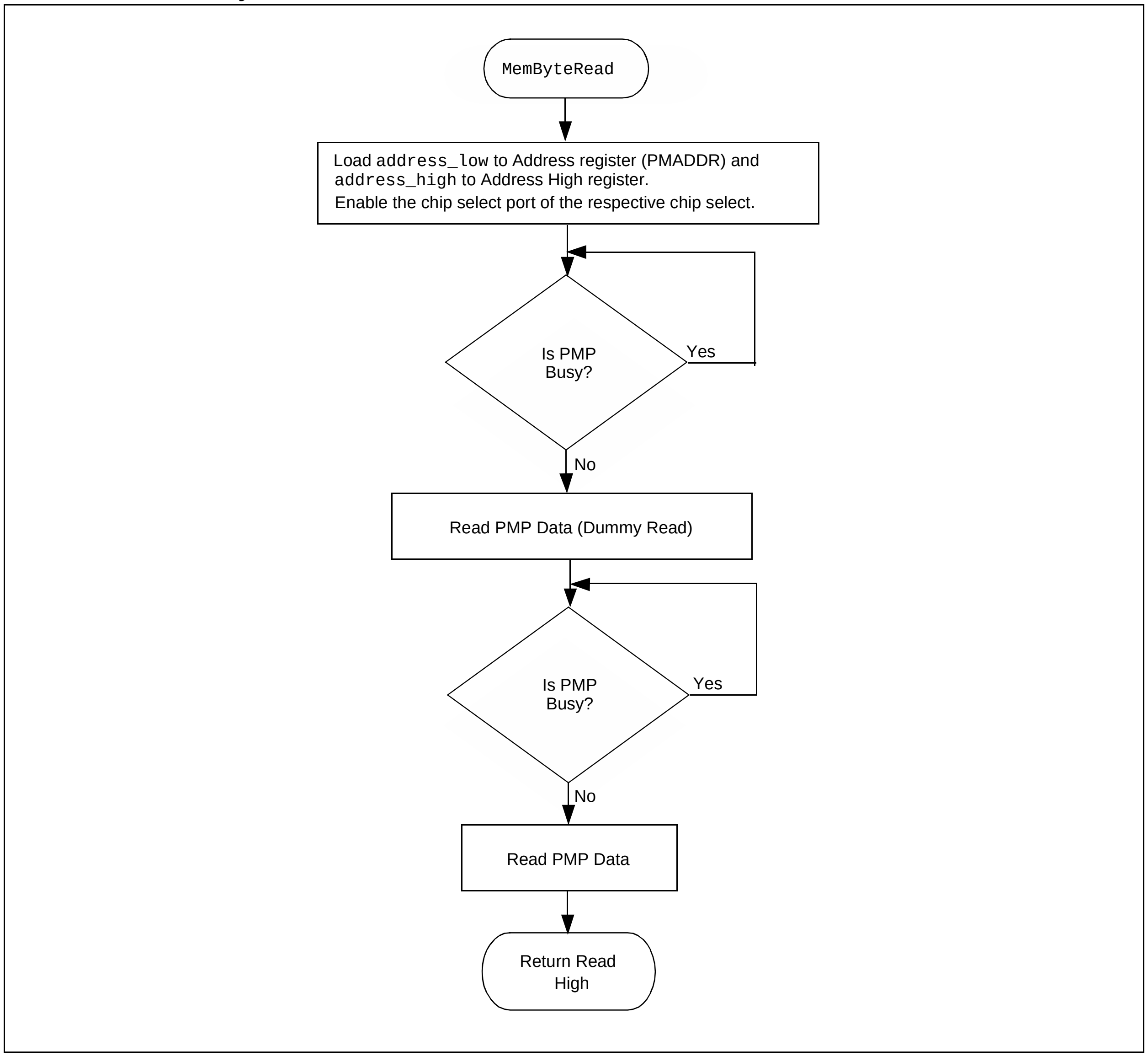

PMPInit
Select mode to have read and write strobes on two different pins.
Select required setup time, hold time and strobe width.
Enable read and write strobe ports.
Select address/data multiplexing as defined in the macros in Memdefn.h file.
If 16-bit data is selected, enable byte enable port.
If the address and data are multiplexed, select the polarity high for the Address Latch Enable signals (ALEL/ALEH).
Select the polarity of the chip select, read and write signals as low.
Byte enable is used; otherwise, select the polarity high.
Enable required ports for address, chip select, and address latch enable.
Drive the I/O ports used for chip select high and those used for address bus low.
Configure these I/O ports as outputs. If these pins have an analog function, configure them as digital.
Return

© 2008 Microchip Technology Inc.

`MemByteRead` API

The API, `MemByteRead`, reads a byte from the specified address. The inputs for this API are the memory address (24 bits) from where the data has to be read. This API returns the data read (8 bits). Figure 20 illustrates the flow of the `MemByteRead` API.

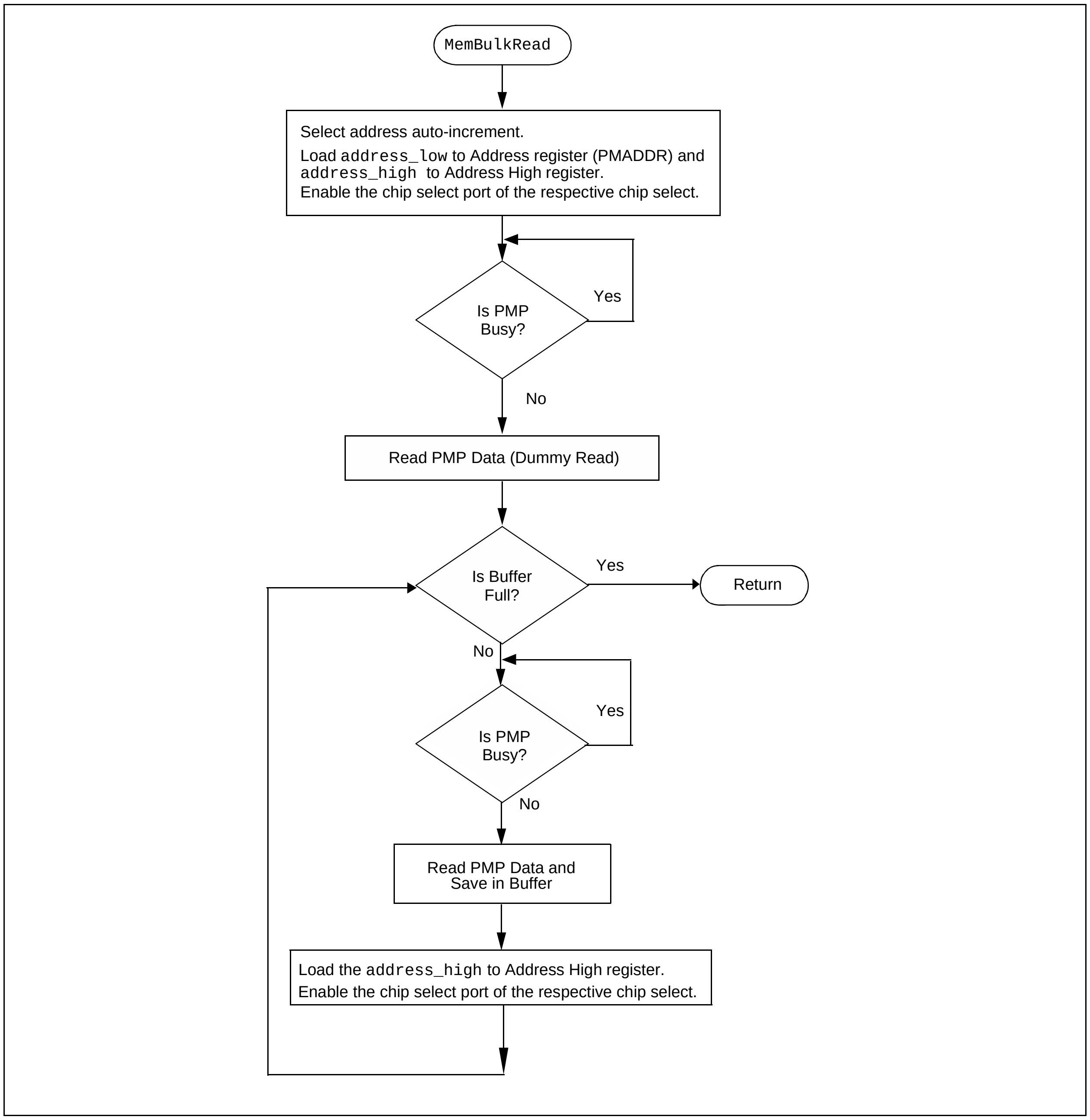
FIGURE 20: `MemByteRead` API FLOWCHART


MemByteRead
Load address_low to Address register (PMADDR) and address_high to Address High register.
Enable the chip select port of the respective chip select.
Is PMP Busy?
Yes
No
Read PMP Data (Dummy Read)
Is PMP Busy?
Yes
No
Read PMP Data
Return Read High

MemBulkRead API

The API MemBulkRead reads a sequence of a specified length of data from the specified address, and stores it in a specified array. The inputs for this API are the starting memory address (24 bits) from where the data has to be read, the number of data bytes (maximum 64 Kbytes) to be read and the address of the array (of char) where the read data needs to be stored. This API returns nothing; it stores the read data in the passed array. Figure 20 illustrates the flow of the MemBulkRead API.

FIGURE 21: MemBulkRead API FLOWCHART

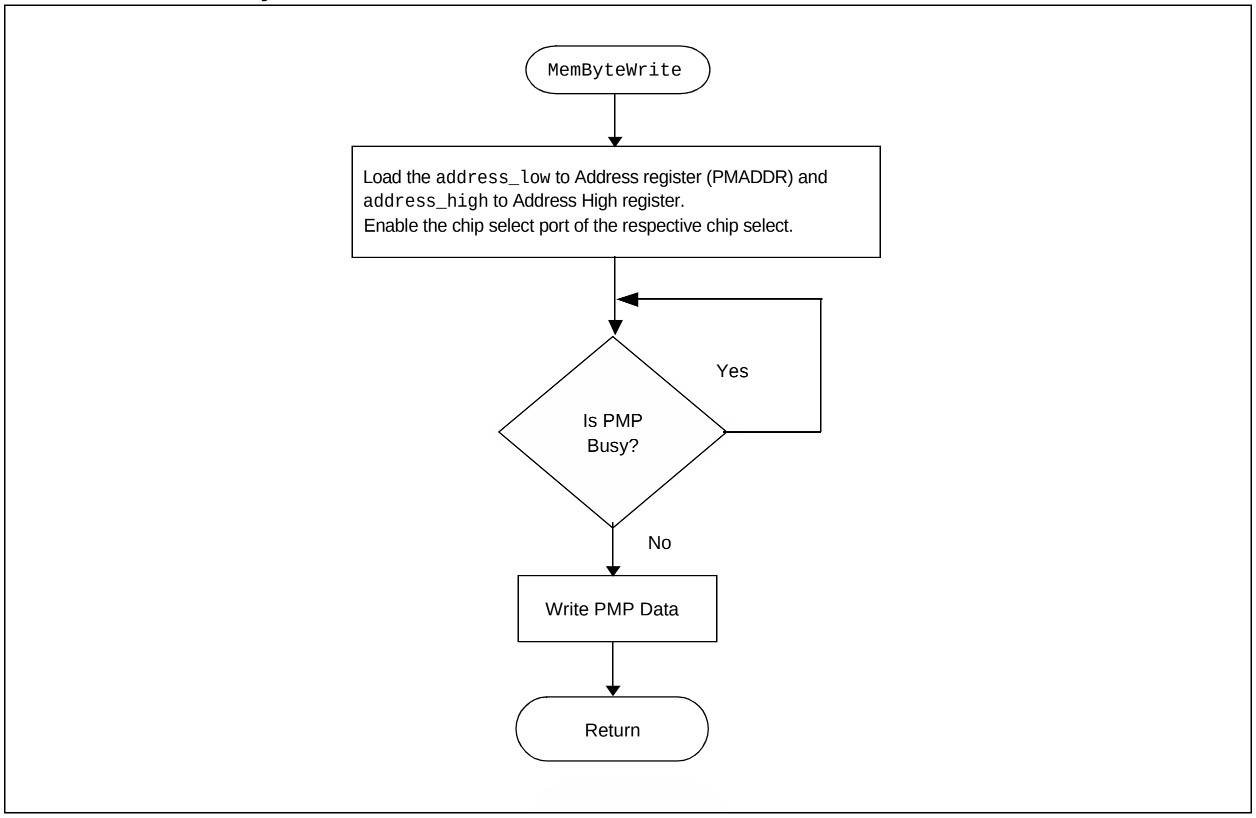

MemBulkRead
Select address auto-increment.
Load address_low to Address register (PMADDR) and address_high to Address High register.
Enable the chip select port of the respective chip select.
Is PMP Busy?
Yes
No
Read PMP Data (Dummy Read)
Is Buffer Full?
Yes
Return
No
Is PMP Busy?
Yes
No
Read PMP Data and Save in Buffer
Load the address_high to Address High register.
Enable the chip select port of the respective chip select.

© 2008 Microchip Technology Inc.

`MemByteWrite` API

The API, `MemByteWrite`, writes a byte to the specified address. The inputs for this API are the memory address (24 bits) where the data has to be written and the data (8 bits) that needs to be written. This API returns nothing. Figure 22 illustrates the flow of the `MemByteWrite` API.

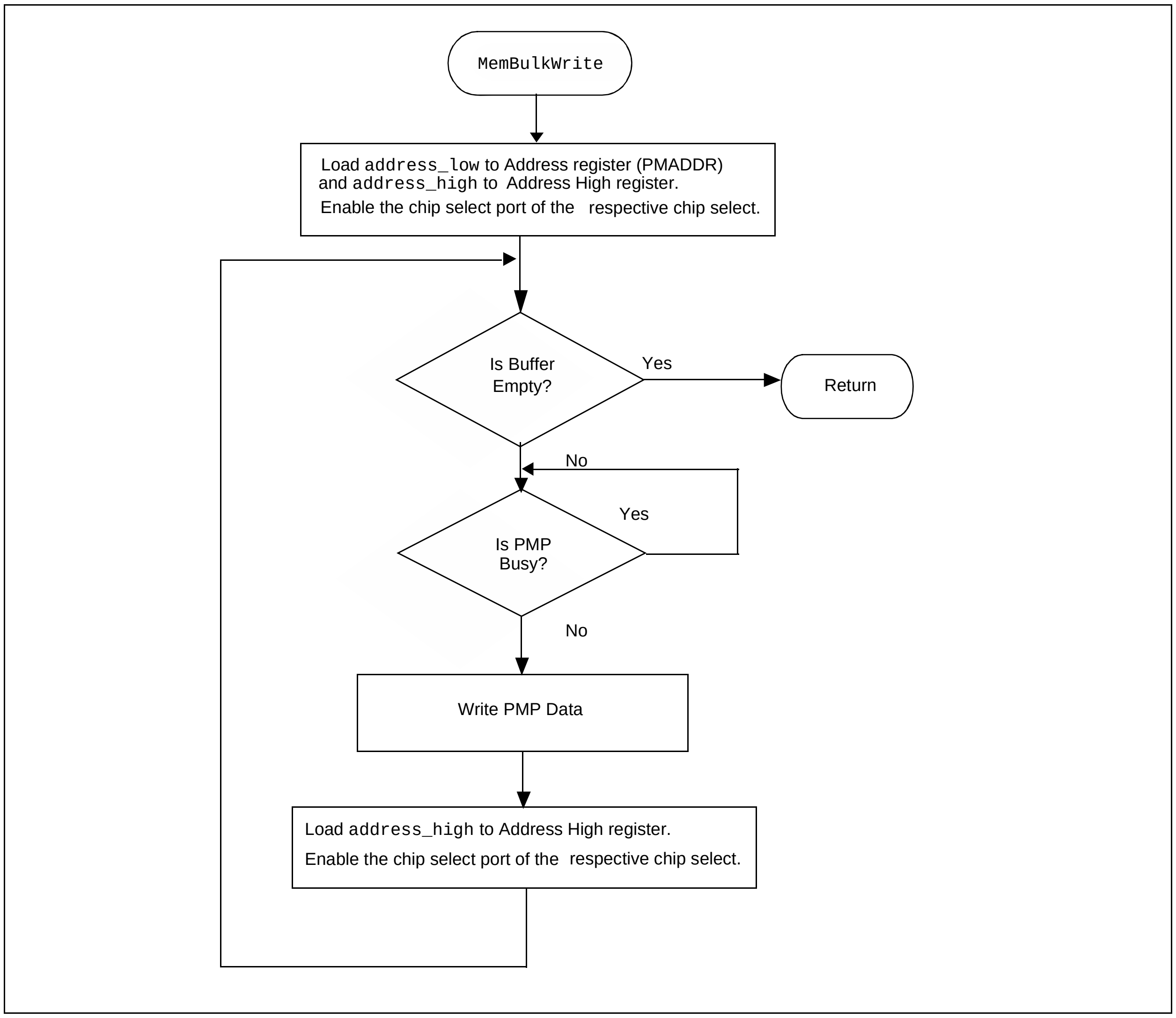
FIGURE 22: `MemByteWrite` API FLOWCHART


MemByteWrite
Load the address_low to Address register (PMADDR) and address_high to Address High register.
Enable the chip select port of the respective chip select.
Is PMP Busy?
Yes
No
Write PMP Data
Return

MemBulkWrite API

The API, MemBulkWrite, writes a sequence of a specified length of bytes which is stored in an array from the specified address. The inputs for this API are, the starting memory address (24 bits) from where the data has to be written, the number of data bytes to be written (maximum 64 Kbytes) and the address of the array (of char) where the data to be written is stored. This API returns nothing. Figure 23 illustrates the flow of the MemBulkWrite API.

FIGURE 23: MemBulkWrite API FLOWCHART


MemBulkWrite
Load address_low to Address register (PMADDR) and address_high to Address High register.
Enable the chip select port of the respective chip select.
Is Buffer Empty?
Yes
Return
No
Is PMP Busy?
Yes
No
Write PMP Data
Load address_high to Address High register.
Enable the chip select port of the respective chip select.

© 2008 Microchip Technology Inc.

CONCLUSION

This application note discusses different ways of interfacing the memory device with the PMP module. There are merits and demerits of each interface type. One should select the appropriate interface type that suits the application most. The APIs provided can be easily used to interface memory with PMP. One has to set the definitions as per the requirements in the `MIDefn.h` file. The `MemIntfExample.c` file is an example file that describes how to use the APIs. All the APIs are provided in the `MemInterface.c` file.

Refer to **Section 13. "Parallel Master Port (PMP)"** in the *"PIC24F Family Reference Manual"* for more information.

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 provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE**.** Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, rfPIC and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, PICkit, PICDEM, PICDEM.net, PICtail, PIC^{32} logo, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, Select Mode, Total Endurance, UNI/O, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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.

© 2008, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 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:
http://support.microchip.com
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-572-9526
Fax: 886-3-572-6459

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/02/08

© 2008 Microchip Technology Inc.