Microchip Technology - PIC24FJ256GA705-I/M4 Datasheet

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	16-Bit
Speed	32MHz
Connectivity	I²C, IrDA, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, I²S, LVD, POR, PWM, WDT
Number of I/O	40
Program Memory Size	256KB (85.5K x 24)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	16K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 14x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	48-UFQFN Exposed Pad
Supplier Device Package	48-UQFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic24fj256ga705-i-m4

Peripheral Features

- High-Current Sink/Source 18 mA/18 mA on All I/O Pins
- Independent, Low-Power 32 kHz Timer Oscillator
- Timer1: 16-Bit Timer/Counter with External Crystal Oscillator; Timer1 can Provide an A/D Trigger
- Timer2,3: 16-Bit Timer/Counter, can Create 32-Bit Timer; Timer3 can Provide an A/D Trigger
- Three Input Capture modules, Each with a 16-Bit Timer
- Three Output Compare/PWM modules, Each with a 16-Bit Timer
- Four MCCP modules, Each with a Dedicated 16/32-Bit Timer:
  - One 6-output MCCP module
  - Three 2-output MCCP modules
- Three Variable Width, Synchronous Peripheral Interface (SPI) Ports on All Devices; 3 Operation modes:
  - 3-wire SPI (supports all 4 SPI modes)
  - 8 by 16-bit or 8 by 8-bit FIFO
  - I^2S mode
- Two I^2C Master and Slave w/Address Masking, and IPMI Support
- Two UART modules:
  - LIN/J2602 bus support (auto-wake-up, Auto-Baud Detect (ABD), Break character support)
  - RS-232 and RS-485 support
  - IrDA® mode (hardware encoder/decoder functions)
- Five External Interrupt Pins
- Parallel Master Port/Enhanced Parallel Slave Port (PMP/EPSP), 8-Bit Data with External Programmable Control (polarity and protocol)
- Enhanced CRC module
- Reference Clock Output with Programmable Divider
- Two Configurable Logic Cell (CLC) Blocks:
  - Two inputs and one output, all mappable to peripherals or I/O pins
  - AND/OR/XOR logic and D/JK flip-flop functions
- Peripheral Pin Select (PPS) with Independent I/O Mapping of Many Peripherals

Device	Memory		Pins	GPIO	DMA Channels	Peripherals													JTAG
	Program (bytes)	SRAM (bytes)				10/12-Bit A/D Channels	Comparators	CRC	MCCP 6-Output/2-Output	IC/OC PWM	16-Bit Timers	I^2C	Variable Width SPI	LIN-USART/IrDA®	CTMU Channels	EPMP (Address/Data Line)	CLC	RTCC	
PIC24FJ64GA705	64K	16K	48	40	6	14	3	Yes	1/3	3/3	3	2	3	2	13	10/8	2	Yes	Yes
PIC24FJ128GA705	128K	16K	48	40	6	14	3	Yes	1/3	3/3	3	2	3	2	13	10/8	2	Yes	Yes
PIC24FJ256GA705	256K	16K	48	40	6	14	3	Yes	1/3	3/3	3	2	3	2	13	10/8	2	Yes	Yes
PIC24FJ64GA704	64K	16K	44	36	6	14	3	Yes	1/3	3/3	3	2	3	2	13	10/8	2	Yes	Yes
PIC24FJ128GA704	128K	16K	44	36	6	14	3	Yes	1/3	3/3	3	2	3	2	13	10/8	2	Yes	Yes
PIC24FJ256GA704	256K	16K	44	36	6	14	3	Yes	1/3	3/3	3	2	3	2	13	10/8	2	Yes	Yes
PIC24FJ64GA702	64K	16K	28	22	6	10	3	Yes	1/3	3/3	3	2	3	2	12	No	2	Yes	Yes
PIC24FJ128GA702	128K	16K	28	22	6	10	3	Yes	1/3	3/3	3	2	3	2	12	No	2	Yes	Yes
PIC24FJ256GA702	256K	16K	28	22	6	10	3	Yes	1/3	3/3	3	2	3	2	12	No	2	Yes	Yes

4.2 Data Memory Space

> **Note:** This data sheet summarizes the features of this group of PIC24F devices. It is not intended to be a comprehensive reference source. For more information, refer to the *"dsPIC33/PIC24 Family Reference Manual"*, **"Data Memory with Extended Data Space (EDS)"** (DS39733). The information in this data sheet supersedes the information in the FRM.

The PIC24F core has a 16-bit wide data memory space, addressable as a single linear range. The Data Space is accessed using two Address Generation Units (AGUs), one each for read and write operations. The Data Space memory map is shown in Figure 4-2.

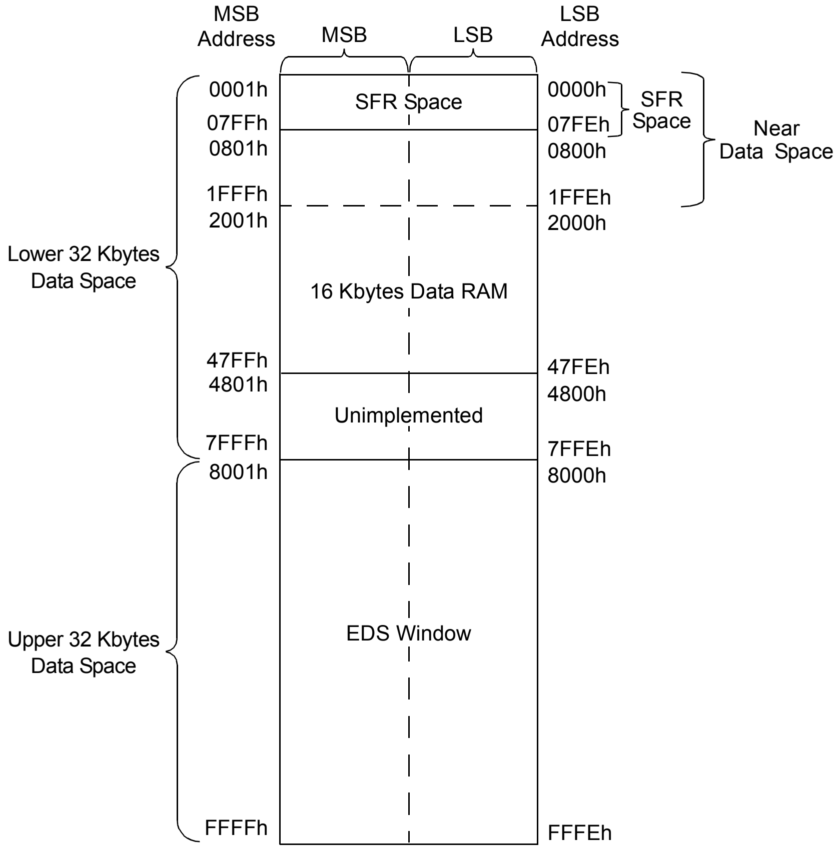
The 16-bit wide data addresses in the data memory space point to bytes within the Data Space (DS). This gives a DS address range of 16 Kbytes or 8K words. The lower half (0000h to 7FFFh) is used for implemented (on-chip) memory addresses.

The upper half of data memory address space (8000h to FFFFh) is used as a window into the Extended Data Space (EDS). This allows the microcontroller to directly access a greater range of data beyond the standard 16-bit address range. EDS is discussed in detail in **Section 4.2.5 "Extended Data Space (EDS)"**.

4.2.1 DATA SPACE WIDTH

The data memory space is organized in byte-addressable, 16-bit wide blocks. Data is aligned in data memory and registers as 16-bit words, but all Data Space EAs resolve to bytes. The Least Significant Bytes (LSBs) of each word have even addresses, while the Most Significant Bytes (MSBs) have odd addresses.

FIGURE 4-2: DATA SPACE MEMORY MAP FOR PIC24FJ256GA705 DEVICES



Note: Memory areas are not shown to scale.

4.2.2 DATA MEMORY ORGANIZATION AND ALIGNMENT

To maintain backward compatibility with PIC® MCUs and improve Data Space memory usage efficiency, the PIC24F instruction set supports both word and byte operations. As a consequence of byte accessibility, all EA calculations are internally scaled to step through word-aligned memory. For example, the core recognizes that Post-Modified Register Indirect Addressing mode, [Ws++], will result in a value of Ws + 1 for byte operations and Ws + 2 for word operations.

Data byte reads will read the complete word, which contains the byte, using the LSB of any EA to determine which byte to select. The selected byte is placed onto the LSB of the data path. That is, data memory and registers are organized as two parallel, byte-wide entities with shared (word) address decode, but separate write lines. Data byte writes only write to the corresponding side of the array or register which matches the byte address.

All word accesses must be aligned to an even address. Misaligned word data fetches are not supported, so care must be taken when mixing byte and word operations or translating from 8-bit MCU code. If a misaligned read or write is attempted, an address error trap will be generated. If the error occurred on a read, the instruction underway is completed; if it occurred on a write, the instruction will be executed but the write will not occur. In either case, a trap is then executed, allowing the system and/or user to examine the machine state prior to execution of the address Fault.

All byte loads into any W register are loaded into the LSB. The Most Significant Byte (MSB) is not modified.

A Sign-Extend (SE) instruction is provided to allow users to translate 8-bit signed data to 16-bit signed values. Alternatively, for 16-bit unsigned data, users can clear the MSB of any W register by executing a Zero-Extend (ZE) instruction on the appropriate address.

Although most instructions are capable of operating on word or byte data sizes, it should be noted that some instructions operate only on words.

4.2.3 NEAR DATA SPACE

The 8-Kbyte area between 0000h and 1FFFh is referred to as the Near Data Space. Locations in this space are directly addressable via a 13-bit absolute address field within all memory direct instructions. The remainder of the Data Space is addressable indirectly. Additionally, the whole Data Space is addressable using MOV instructions, which support Memory Direct Addressing with a 16-bit address field.

4.2.4 SPECIAL FUNCTION REGISTER (SFR) SPACE

The first 2 Kbytes of the Near Data Space, from 0000h to 07FFh, are primarily occupied with Special Function Registers (SFRs). These are used by the PIC24F core and peripheral modules for controlling the operation of the device.

SFRs are distributed among the modules that they control and are generally grouped together by module. Much of the SFR space contains unused addresses; these are read as ‘0’. A diagram of the SFR space, showing where the SFRs are actually implemented, is shown in Table 4-3. Each implemented area indicates a 32-byte region where at least one address is implemented as an SFR. A complete list of implemented SFRs, including their addresses, is shown in Table 4-4 through 4-11.

TABLE 4-3: IMPLEMENTED REGIONS OF SFR DATA SPACE[2]

	SFR Space Address															
	xx00	**xx10**	**xx20**	**xx30**	**xx40**	**xx50**	**xx60**	**xx70**	**xx80**	**xx90**	**xxA0**	**xxB0**	**xxC0**	**xxD0**	**xxE0**	**xxF0**
000h	Core															
100h	OSC	Reset[1]	EPMP			CRC	REFO	PMD		Timers		—	CTMU	RTCC		
200h	Capture			Compare			MCCP								Comp	ANCFG
300h	MCCP				—	—	—	—	—	UART				—	—	— SPI
400h	SPI					—	CLC		—	—	I^2C			DMA		
500h	DMA	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
600h	—	—	—	—	—	I/O										—
700h	—	A/D					NVM	—	—	PPS						

Legend: — = No implemented SFRs in this block

Note 1: Includes HLVD control.

2: Regions shown are approximate. Refer to Table 4-4 through Table 4-11 for exact addresses.

TABLE 4-4: SFR MAP: 0000h BLOCK

File Name	Address	All Resets	File Name	Address	All Resets
CPU CORE			**INTERRUPT CONTROLLER (CONTINUED)**		
WREG0	0000	0000	IEC1	009A	0000
WREG1	0002	0000	IEC2	009C	0000
WREG2	0004	0000	IEC3	009E	0000
WREG3	0006	0000	IEC4	00A0	0000
WREG4	0008	0000	IEC5	00A2	0000
WREG5	000A	0000	IEC6	00A4	0000
WREG6	000C	0000	IEC7	00A6	0000
WREG7	000E	0000	IPC0	00A8	4444
WREG8	0010	0000	IPC1	00AA	4444
WREG9	0012	0000	IPC2	00AC	4444
WREG10	0014	0000	IPC3	00AE	4444
WREG11	0016	0000	IPC4	00B0	4444
WREG12	0018	0000	IPC5	00B2	4404
WREG13	001A	0000	IPC6	00B4	4444
WREG14	001C	0000	IPC7	00B6	4444
WREG15	001E	0800	IPC8	00B8	0044
SPLIM	0020	xxxx	IPC9	00BA	4444
PCL	002E	0000	IPC10	00BC	4444
PCH	0030	0000	IPC11	00BE	4444
DSRPAG	0032	0000	IPC12	00C0	4444
DSWPAG	0034	0000	IPC13	00C2	0440
RCOUNT	0036	xxxx	IPC14	00C4	4400
SR	0042	0000	IPC15	00C6	4444
CORCON	0044	0004	IPC16	00C8	4444
DISICNT	0052	xxxx	IPC17	00CA	4444
TBLPAG	0054	0000	IPC18	00CC	0044
INTERRUPT CONTROLLER			IPC19	00CE	0040
INTCON1	0080	0000	IPC20	00D0	4440
INTCON2	0082	8000	IPC21	00D2	4444
INTCON4	0086	0000	IPC22	00D4	4444
IFS0	0088	0000	IPC23	00D6	4400
IFS1	008A	0000	IPC24	00D8	4444
IFS2	008C	0000	IPC25	00DA	0440
IFS3	008E	0000	IPC26	00DC	0400
IFS4	0090	0000	IPC27	00DE	4440
IFS5	0092	0000	IPC28	00E0	4444
IFS6	0094	0000	IPC29	00E2	0044
IFS7	0096	0000	INTTREG	00E4	0000
IEC0	0098	0000			

Legend: x = undefined. Reset values are shown in hexadecimal.

4.3.3 READING DATA FROM PROGRAM MEMORY USING EDS

The upper 32 Kbytes of Data Space may optionally be mapped into any 16K word page of the program space. This provides transparent access of stored constant data from the Data Space without the need to use special instructions (i.e., `TBLRDL/H`).

Program space access through the Data Space occurs when the MSb of EA is '1' and the DSRPAG<9> bit is also '1'. The lower 8 bits of DSRPAG are concatenated to the Wn<14:0> bits to form a 23-bit EA to access program memory. The DSRPAG<8> decides which word should be addressed; when the bit is '0', the lower word, and when '1', the upper word of the program memory is accessed.

The entire program memory is divided into 512 EDS pages, from 200h to 3FFh, each consisting of 16K words of data. Pages, 200h to 2FFh, correspond to the lower words of the program memory, while 300h to 3FFh correspond to the upper words of the program memory.

Using this EDS technique, the entire program memory can be accessed. Previously, the access to the upper word of the program memory was not supported.

Table 4-15 provides the corresponding 23-bit EDS address for program memory with EDS page and source addresses.

For operations that use PSV and are executed outside a `REPEAT` loop, the `MOV` and `MOV.D` instructions will require one instruction cycle in addition to the specified execution time. All other instructions will require two instruction cycles in addition to the specified execution time.

For operations that use PSV, which are executed inside a `REPEAT` loop, there will be some instances that require two instruction cycles in addition to the specified execution time of the instruction:

- Execution in the first iteration
- Execution in the last iteration
- Execution prior to exiting the loop due to an interrupt
- Execution upon re-entering the loop after an interrupt is serviced

Any other iteration of the `REPEAT` loop will allow the instruction accessing data, using PSV, to execute in a single cycle.

TABLE 4-15: EDS PROGRAM ADDRESS WITH DIFFERENT PAGES AND ADDRESSES

DSRPAG (Data Space Read Register)	Source Address while Indirect Addressing	23-Bit EA Pointing to EDS	Comment
200h • • • 2FFh	8000h to FFFFh	000000h to 007FFEh • • • 7F8000h to 7FFFFEh	Lower words of 4M program instructions (8 Mbytes) for read operations only.
300h • • • 3FFh		000001h to 007FFFh • • • 7F8001h to 7FFFFFh	Upper words of 4M program instructions (4 Mbytes remaining; 4 Mbytes are phantom bytes) for read operations only.
000h		Invalid Address	Address error trap.[1]

Note 1: When the source/destination address is above 8000h and DSRPAG/DSWPAG is '0', an address error trap will occur.

EXAMPLE 4-3: EDS READ CODE FROM PROGRAM MEMORY IN ASSEMBLY

```
; Set the EDS page from where the data to be read
    mov     #0x0202, w0
    mov     w0, DSRPAG                ;page 0x202, consisting lower words, is selected for read
    mov     #0x000A, w1               ;select the location (0x0A) to be read
    bset    w1, #15                   ;set the MSB of the base address, enable EDS mode
;Read a byte from the selected location
    mov.b   [w1++], w2                ;read Low byte
    mov.b   [w1++], w3                ;read High byte
;Read a word from the selected location
    mov     [w1], w2                  ;
;Read Double - word from the selected location
    mov.d   [w1], w2                  ;two word read, stored in w2 and w3
```

9.4 Clock Switching Operation

With few limitations, applications are free to switch between any of the four clock sources (POSC, SOSC, FRC and LPRC) under software control and at any time. To limit the possible side effects that could result from this flexibility, PIC24F devices have a safeguard lock built into the switching process.

Note: The Primary Oscillator mode has three different submodes (XT, HS and EC), which are determined by the POSCMDx Configuration bits. While an application can switch to and from Primary Oscillator mode in software, it cannot switch between the different primary submodes without reprogramming the device.

9.4.1 ENABLING CLOCK SWITCHING

To enable clock switching, the FCKSM1 Configuration bit in FOSC must be programmed to '0'. (Refer to **Section 29.1 "Configuration Bits"** for further details.) If the FCKSM1 Configuration bit is unprogrammed ('1'), the clock switching function and Fail-Safe Clock Monitor function are disabled; this is the default setting.

The NOSCx control bits (OSCCON<10:8>) do not control the clock selection when clock switching is disabled. However, the COSC<2:0> bits (OSCCON<14:12>) will reflect the clock source selected by the FNOSCx Configuration bits.

The OSWEN control bit (OSCCON<0>) has no effect when clock switching is disabled; it is held at '0' at all times.

9.4.2 OSCILLATOR SWITCHING SEQUENCE

At a minimum, performing a clock switch requires this basic sequence:

1. If desired, read the COSCx bits (OSCCON<14:12>) to determine the current oscillator source.
2. Perform the unlock sequence to allow a write to the OSCCON register high byte.
3. Write the appropriate value to the NOSCx bits (OSCCON<10:8>) for the new oscillator source.
4. Perform the unlock sequence to allow a write to the OSCCON register low byte.
5. Set the OSWEN bit to initiate the oscillator switch.

Once the basic sequence is completed, the system clock hardware responds automatically as follows:

1. The clock switching hardware compares the COSCx bits with the new value of the NOSCx bits. If they are the same, then the clock switch is a redundant operation. In this case, the OSWEN bit is cleared automatically and the clock switch is aborted.
2. If a valid clock switch has been initiated, the LOCK (OSCCON<5>) and CF (OSCCON<3>) bits are cleared.
3. The new oscillator is turned on by the hardware if it is not currently running. If a crystal oscillator must be turned on, the hardware will wait until the OST expires. If the new source is using the PLL, then the hardware waits until a PLL lock is detected (LOCK = 1).
4. The hardware waits for 10 clock cycles from the new clock source and then performs the clock switch.
5. The hardware clears the OSWEN bit to indicate a successful clock transition. In addition, the NOSCx bits value is transferred to the COSCx bits.
6. The old clock source is turned off at this time, with the exception of LPRC (if WDT or FSCM is enabled) or SOSC (if SOSCEN remains set).

Note 1: The processor will continue to execute code throughout the clock switching sequence. Timing-sensitive code should not be executed during this time.

2: Direct clock switches between any Primary Oscillator mode with PLL and FRCPLL mode are not permitted. This applies to clock switches in either direction. In these instances, the application must switch to FRC mode as a transitional clock source between the two PLL modes.

REGISTER 10-1: PMD1: PERIPHERAL MODULE DISABLE REGISTER 1

U-0	U-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0
—	—	T3MD	T2MD	T1MD	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0
I2C1MD	U2MD	U1MD	SPI2MD	SPI1MD	—	—	ADC1MD
bit 7							bit 0

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

bit 15-14 **Unimplemented:** Read as ‘0’

bit 13 **T3MD:** Timer3 Module Disable bit
1 = Module is disabled
0 = Module power and clock sources are enabled

bit 12 **T2MD:** Timer2 Module Disable bit
1 = Module is disabled
0 = Module power and clock sources are enabled

bit 11 **T1MD:** Timer1 Module Disable bit
1 = Module is disabled
0 = Module power and clock sources are enabled

bit 10-8 **Unimplemented:** Read as ‘0’

bit 7 **I2C1MD:** I2C1 Module Disable bit
1 = Module is disabled
0 = Module power and clock sources are enabled

bit 6 **U2MD:** UART2 Module Disable bit
1 = Module is disabled
0 = Module power and clock sources are enabled

bit 5 **U1MD:** UART1 Module Disable bit
1 = Module is disabled
0 = Module power and clock sources are enabled

bit 4 **SPI2MD:** SPI2 Module Disable bit
1 = Module is disabled
0 = Module power and clock sources are enabled

bit 3 **SPI1MD:** SPI1 Module Disable bit
1 = Module is disabled
0 = Module power and clock sources are enabled

bit 2-1 **Unimplemented:** Read as ‘0’

bit 0 **ADC1MD:** A/D Converter Module Disable bit
1 = Module is disabled
0 = Module power and clock sources are enabled

REGISTER 10-5: PMD5: PERIPHERAL MODULE DISABLE REGISTER 5

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	CCP4MD	CCP3MD	CCP2MD	CCP1MD
bit 7							bit 0

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

bit 15-4 **Unimplemented:** Read as ‘0’

bit 3 **CCP4MD:** MCCP4 Module Disable bit
1 = Module is disabled
0 = Module power and clock sources are enabled

bit 2 **CCP3MD:** MCCP3 Module Disable bit
1 = Module is disabled
0 = Module power and clock sources are enabled

bit 1 **CCP2MD:** MCCP2 Module Disable bit
1 = Module is disabled
0 = Module power and clock sources are enabled

bit 0 **CCP1MD:** MCCP1 Module Disable bit
1 = Module is disabled
0 = Module power and clock sources are enabled

REGISTER 11-8: IOCPx: INTERRUPT-ON-CHANGE POSITIVE EDGE x REGISTER(1,2,3)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
IOCPx<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IOCPx<7:0>							
bit 7							bit 0

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

bit 15-0 **IOCPx<15:0>:** Interrupt-on-Change Positive Edge x Enable bits

1 = Interrupt-on-Change is enabled on the IOCx pin for a positive going edge; the associated status bit and interrupt flag will be set upon detecting an edge

0 = Interrupt-on-Change is disabled on the IOCx pin for a positive going edge

Note 1: Setting both IOCPx and IOCNx will enable the IOCx pin for both edges, while clearing both registers will disable the functionality.

2: Changing the value of this register while the module is enabled (IOCON = 1) may cause a spurious IOC event. The corresponding interrupt must be ignored, cleared (using IOCFx) or masked (within the interrupt controller), or this module must be enabled (IOCON = 0) when changing this register.

3: See Table 11-3, Table 11-4 and Table 11-5 for individual bit availability in this register.

REGISTER 11-9: IOCNx: INTERRUPT-ON-CHANGE NEGATIVE EDGE x REGISTER(1,2,3)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
IOCNx<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IOCNx<7:0>							
bit 7							bit 0

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

bit 15-0 **IOCNx<15:0>:** Interrupt-on-Change Negative Edge x Enable bits

1 = Interrupt-on-Change is enabled on the IOCx pin for a negative going edge; the associated status bit and interrupt flag will be set upon detecting an edge

0 = Interrupt-on-Change is disabled on the IOCx pin for a negative going edge

Note 1: Setting both IOCPx and IOCNx will enable the IOCx pin for both edges, while clearing both registers will disable the functionality.

2: Changing the value of this register while the module is enabled (IOCON = 1) may cause a spurious IOC event. The corresponding interrupt must be ignored, cleared (using IOCFx) or masked (within the interrupt controller), or this module must be enabled (IOCON = 0) when changing this register.

3: See Table 11-3, Table 11-4 and Table 11-5 for individual bit availability in this register.

TABLE 20-2: ENHANCED PARALLEL MASTER PORT PIN DESCRIPTIONS

Pin Name (Alternate Function)	Type	Description
PMA<22:16>	O	Address Bus bits<22:16>
PMA14	O	Address Bus bit 14
	I/O	Data Bus bit 14 (16-bit port with Multiplexed Addressing)
(PMCS1)	O	Chip Select 1 (alternate location)
PMA<13:8>	O	Address Bus bits<13:8>
	I/O	Data Bus bits<13:8> (16-bit port with Multiplexed Addressing)
PMA<7:3>	O	Address Bus bits<7:3>
PMA2	O	Address Bus bit 2
(PMALU)	O	Address Latch Upper Strobe for Multiplexed Address
PMA1	I/O	Address Bus bit 1
(PMALH)	O	Address Latch High Strobe for Multiplexed Address
PMA0	I/O	Address Bus bit 0
(PMALL)	O	Address Latch Low Strobe for Multiplexed Address
PMD<15:8>	I/O	Data Bus bits<15:8> (Demultiplexed Addressing)
PMD<7:4>	I/O	Data Bus bits<7:4>
	O	Address Bus bits<7:4> (4-bit port with 1-Phase Multiplexed Addressing)
PMD<3:0>	I/O	Data Bus bits<3:0>
PMCS1	O	Chip Select 1
PMCS2	O	Chip Select 2
PMWR	I/O	Write Strobe(1)
(PMENB)	I/O	Enable Signal(1)
PMRD	I/O	Read Strobe(1)
(PMRD/$\overline{\text{PMWR}}$)	I/O	Read/$\overline{\text{Write}}$ Signal(1)
PMBE1	O	Byte Indicator
PMBE0	O	Nibble or Byte Indicator
PMACK1	I	Acknowledgment Signal 1
PMACK2	I	Acknowledgment Signal 2

Note 1: Signal function depends on the setting of the MODE<1:0> and SM bits (PMCON1<9:8> and PMCSxCF<8>).

REGISTER 21-2: RTCCON1H: RTCC CONTROL REGISTER 1 (HIGH)

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ALRMEN	CHIME	—	—	AMASK3	AMASK2	AMASK1	AMASK0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ALMRPT7	ALMRPT6	ALMRPT5	ALMRPT4	ALMRPT3	ALMRPT2	ALMRPT1	ALMRPT0
bit 7							bit 0

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

bit 15 **ALRMEN:** Alarm Enable bit

1 = Alarm is enabled (cleared automatically after an alarm event whenever ALMRPT<7:0> = 00h and CHIME = 0)
0 = Alarm is disabled

bit 14 **CHIME:** Chime Enable bit

1 = Chime is enabled; ALMRPT<7:0> bits roll over from 00h to FFh
0 = Chime is disabled; ALMRPT<7:0> bits stop once they reach 00h

bit 13-12 **Unimplemented:** Read as ‘0’

bit 11-8 **AMASK<3:0>:** Alarm Mask Configuration bits

0000 = Every half second
0000 = Every second
0010 = Every 10 seconds
0011 = Every minute
0100 = Every 10 minutes
0101 = Every hour
0110 = Once a day
0111 = Once a week
1000 = Once a month
1001 = Once a year (except when configured for February 29th, once every 4 years)
101x = Reserved – do not use
11xx = Reserved – do not use

bit 7-0 **ALMRPT<7:0>:** Alarm Repeat Counter Value bits

11111111 = Alarm will repeat 255 more times
•
•
•
00000000 = Alarm will repeat 0 more times
The counter decrements on any alarm event. The counter is prevented from rolling over from ‘00’ to ‘FF’ unless CHIME = 1.

REGISTER 23-4: CLCxGLSL: CLCx GATE LOGIC INPUT SELECT LOW REGISTER (CONTINUED)

bit 8 **G2D1N:** Gate 2 Data Source 1 Negated Enable bit
1 = The Data Source 1 inverted signal is enabled for Gate 2
0 = The Data Source 1 inverted signal is disabled for Gate 2

bit 7 **G1D4T:** Gate 1 Data Source 4 True Enable bit
1 = The Data Source 4 signal is enabled for Gate 1
0 = The Data Source 4 signal is disabled for Gate 1

bit 6 **G1D4N:** Gate 1 Data Source 4 Negated Enable bit
1 = The Data Source 4 inverted signal is enabled for Gate 1
0 = The Data Source 4 inverted signal is disabled for Gate 1

bit 5 **G1D3T:** Gate 1 Data Source 3 True Enable bit
1 = The Data Source 3 signal is enabled for Gate 1
0 = The Data Source 3 signal is disabled for Gate 1

bit 4 **G1D3N:** Gate 1 Data Source 3 Negated Enable bit
1 = The Data Source 3 inverted signal is enabled for Gate 1
0 = The Data Source 3 inverted signal is disabled for Gate 1

bit 3 **G1D2T:** Gate 1 Data Source 2 True Enable bit
1 = The Data Source 2 signal is enabled for Gate 1
0 = The Data Source 2 signal is disabled for Gate 1

bit 2 **G1D2N:** Gate 1 Data Source 2 Negated Enable bit
1 = The Data Source 2 inverted signal is enabled for Gate 1
0 = The Data Source 2 inverted signal is disabled for Gate 1

bit 1 **G1D1T:** Gate 1 Data Source 1 True Enable bit
1 = The Data Source 1 signal is enabled for Gate 1
0 = The Data Source 1 signal is disabled for Gate 1

bit 0 **G1D1N:** Gate 1 Data Source 1 Negated Enable bit
1 = The Data Source 1 inverted signal is enabled for Gate 1
0 = The Data Source 1 inverted signal is disabled for Gate 1

24.4 Achieving Maximum A/D Converter Performance

In order to get the shortest overall conversion time (called the ‘throughput’) while maintaining accuracy, several factors must be considered. These are described in detail below.

- Dependence of AVDD – If the AVDD supply is < 2.7V, the Charge Pump Enable bit (PUMPEN, AD1CON3<13>) should be set to ‘1’. The input channel multiplexer has a varying resistance with AVDD (the lower AVDD, the higher the internal switch resistance). The charge pump provides a higher internal AVDD to keep the switch resistance as low as possible.
- Dependence on TAD – The ADC timing is driven by TAD, not TCYC. Selecting the TAD time correctly is critical to getting the best ADC throughput. It is important to note that the overall ADC throughput is not simply the ‘Conversion Time’ of the SAR. It is the combination of the Conversion Time, the Sample Time and additional TAD delays for internal synchronization logic.
- Relationship between TCYC and TAD – There is not a fixed 1:1 timing relationship between TCYC and TAD. The fastest possible throughput is fundamentally set by TAD (min), not by TCYC. The TAD time is set as a programmable integer multiple of TCYC by the ADCS<7:0> bits. Referring to Table 32-25, the TAD (min) time is greater than the 4 MHz period of the dedicated ADC RC clock generator. Therefore, TAD must be 2 TCYC in order to use the RC clock for fastest throughput. The TAD (min) is a multiple of 3.597 MHz as opposed to 4 MHz. To run as fast as possible, TCYC must be a multiple of TAD (min) because values of ADCSx are integers. For example, if a standard ‘color burst’ crystal of 14.31818 MHz is used, TCYC is 279.4 ns, which is very close to TAD (min) and the ADC throughput is optimal. Running at 16 MHz will actually reduce the throughput, because TAD will have to be 500 ns as the TCYC of 250 ns violates TAD (min).
- Dependence on driving Source Resistance (RS) – Certain transducers have high output impedance (> 2.5 kΩ). Having a high RS will require longer sampling time to charge the S/H cap through the resistance path (see Figure 25-3). The worst case is a full-range voltage step of AVSS to AVDD with the sampling cap at AVSS. The capacitor time constant is (RS + RIC + RSS) (CHOLD) and the sample time needs to be 6 time constants minimum (8 are preferred). Since the ADC logic timing is TAD-based, the sample time (in TAD) must be long enough, over all conditions, to charge/discharge CHOLD. Do not assume one TAD is sufficient sample time; longer times may be required to achieve the accuracy needed by the application. The value of CHOLD is 40 pF.

  A small amount of charge is present at the ADC input pin when the sample switch is closed. If RS is high, this will generate a DC error exceeding 1 LSB. Keeping RS < 50Ω is recommenced for best results. The error can also be reduced by increasing sample time (a 2 kΩ value of RS requires a 3 µS sample time to eliminate the error).
- Calculating Throughput – The throughput of the ADC is based on TAD. The throughput is given by:

  Throughput = 1/(Sample Time + SAR Conversion Time + Clock Sync Time)

  where:

  Sample Time is the calculated TAD periods for the application. SAR Conversion Time is 12 TAD for 10-bit and 14 TAD for 12-bit conversions. Clock Sync Time is 2.5 TAD (worst case).

  Example: For a 12-bit ADC throughput, if using FRC = 8 MHz and the Sample Time is 1 TAD, the use of an 8 MHz FRC means the TCYC = 250 ns and this requires: TAD = 2 TCYC = 500 ns. Therefore, the throughput is:

  *Throughput = 1/(500 ns) + (14 * 500 ns) + (2.5 * 500 ns) = 114.28KS/sec*

  Note that the clock sync delay could be as little as 1.5 TAD, which could produce 121 KS/sec, but that cannot be ensured as the timing relationship is asynchronous and not specified. The worst case timing of 2.5 TAD should be used to calculate throughput.

  Example: A certain transducer has a 20 kΩ output impedance. If AVDD is 3.0, the maximum sample time needed would be determined by the following:

$$\begin{aligned} \textit{Sample Time} &= 6 * (R_S + R_{IC} + R_{SS}) * C_{HOLD} \\ &= 6 * (20K + 250 + 350) * 40\ pF \\ &= 4.95\ \mu S \end{aligned}$$

If TAD = 500 ns, this requires a Sample Time of 4.95 µs/ 500 ns = 10 TAD (for a full-step voltage on the transducer output). RSS is 350Ω because AVDD is above 2.7V.

REGISTER 24-1: AD1CON1: A/D CONTROL REGISTER 1

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADON	—	ADSIDL	DMABM[1]	DMAEN	MODE12	FORM1	FORM0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0, HSC	R/C-0, HSC
SSRC3	SSRC2	SSRC1	SSRC0	—	ASAM	SAMP	DONE
bit 7							bit 0

Legend:	C = Clearable bit	U = Unimplemented bit, read as ‘0’	
R = Readable bit	W = Writable bit	HSC = Hardware Settable/Clearable bit	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15 **ADON:** A/D Operating Mode bit
1 = A/D Converter is operating
0 = A/D Converter is off

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **ADSIDL:** A/D Stop in Idle Mode bit
1 = Discontinues module operation when device enters Idle mode
0 = Continues module operation in Idle mode

bit 12 **DMABM:** Extended DMA Buffer Mode Select bit[1]
1 = Extended Buffer mode: Buffer address is defined by the DMADSTn register
0 = PIA mode: Buffer addresses are defined by the DMA Controller and AD1CON4<2:0>

bit 11 **DMAEN:** Extended DMA/Buffer Enable bit
1 = Extended DMA and buffer features are enabled
0 = Extended features are disabled

bit 10 **MODE12:** A/D 12-Bit Operation Mode bit
1 = 12-bit A/D operation
0 = 10-bit A/D operation

bit 9-8 **FORM<1:0>:** Data Output Format bits (see formats following)
11 = Fractional result, signed, left justified
10 = Absolute fractional result, unsigned, left justified
01 = Decimal result, signed, right justified
00 = Absolute decimal result, unsigned, right justified

bit 7-4 **SSRC<3:0>:** Sample Clock Source Select bits
0000 = SAMP is cleared by software
0001 = INT0
0010 = Timer3
0100 = CTMU trigger
0101 = Timer1 (will not trigger during Sleep mode)
0110 = Timer1 (may trigger during Sleep mode)
0111 = Auto-Convert mode

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **ASAM:** A/D Sample Auto-Start bit
1 = Sampling begins immediately after last conversion; SAMP bit is auto-set
0 = Sampling begins when SAMP bit is manually set

Note 1: This bit is only available when Extended DMA and buffer features are available (DMAEN = 1).

REGISTER 24-4: AD1CON4: A/D CONTROL REGISTER 4

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	DMABL<2:0>[1]		
bit 7							bit 0

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

bit 15-3 **Unimplemented:** Read as ‘0’

bit 2-0 **DMABL<2:0>:** DMA Buffer Size Select bits[1]

111 = Allocates 128 words of buffer to each analog input
110 = Allocates 64 words of buffer to each analog input
101 = Allocates 32 words of buffer to each analog input
100 = Allocates 16 words of buffer to each analog input
011 = Allocates 8 words of buffer to each analog input
010 = Allocates 4 words of buffer to each analog input
001 = Allocates 2 words of buffer to each analog input
000 = Allocates 1 word of buffer to each analog input

Note 1: The DMABL<2:0> bits are only used when AD1CON1<11> = 1 and AD1CON1<12> = 0; otherwise, their value is ignored.

REGISTER 29-3: FSIGN CONFIGURATION REGISTER

U-1	U-1	U-1	U-1	U-1	U-1	U-1	U-1
—	—	—	—	—	—	—	—
bit 23							bit 16

r-0	U-1	U-1	U-1	U-1	U-1	U-1	U-1
—	—	—	—	—	—	—	—
bit 15							bit 8

U-1	U-1	U-1	U-1	U-1	U-1	U-1	U-1
—	—	—	—	—	—	—	—
bit 7							bit 0

Legend:	PO = Program Once bit	r = Reserved bit	
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

bit 23-16 **Unimplemented:** Read as ‘1’

bit 15 **Reserved:** Maintain as ‘0’

bit 14-0 **Unimplemented:** Read as ‘1’

30.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

For easy source level debugging, the compilers provide debug information that is optimized to the MPLAB X IDE.

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. MPLAB XC Compiler uses the assembler to produce its object file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

30.3 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code, and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

30.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

30.5 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

TABLE 31-1: SYMBOLS USED IN OPCODE DESCRIPTIONS

Field	Description
#text	Means literal defined by "text"
(text)	Means "content of text"
[text]	Means "the location addressed by text"
{ }	Optional field or operation
<n:m>	Register bit field
.b	Byte mode selection
.d	Double-Word mode selection
.S	Shadow register select
.w	Word mode selection (default)
bit4	4-bit Bit Selection field (used in word addressed instructions) ∈ {0...15}
C, DC, N, OV, Z	MCU Status bits: Carry, Digit Carry, Negative, Overflow, Sticky Zero
Expr	Absolute address, label or expression (resolved by the linker)
f	File register address ∈ {0000h...1FFFh}
lit1	1-bit unsigned literal ∈ {0,1}
lit4	4-bit unsigned literal ∈ {0...15}
lit5	5-bit unsigned literal ∈ {0...31}
lit8	8-bit unsigned literal ∈ {0...255}
lit10	10-bit unsigned literal ∈ {0...255} for Byte mode, {0:1023} for Word mode
lit14	14-bit unsigned literal ∈ {0...16383}
lit16	16-bit unsigned literal ∈ {0...65535}
lit23	23-bit unsigned literal ∈ {0...8388607}; LSB must be '0'
None	Field does not require an entry, may be blank
PC	Program Counter
Slit10	10-bit signed literal ∈ {-512...511}
Slit16	16-bit signed literal ∈ {-32768...32767}
Slit6	6-bit signed literal ∈ {-16...16}
Wb	Base W register ∈ {W0..W15}
Wd	Destination W register ∈ { Wd, [Wd], [Wd++], [Wd--], [++Wd], [--Wd] }
Wdo	Destination W register ∈ { Wnd, [Wnd], [Wnd++], [Wnd--], [++Wnd], [--Wnd], [Wnd+Wb] }
Wm,Wn	Dividend, Divisor Working register pair (direct addressing)
Wn	One of 16 Working registers ∈ {W0..W15}
Wnd	One of 16 destination Working registers ∈ {W0..W15}
Wns	One of 16 source Working registers ∈ {W0..W15}
WREG	W0 (Working register used in file register instructions)
Ws	Source W register ∈ { Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws] }
Wso	Source W register ∈ { Wns, [Wns], [Wns++], [Wns--], [++Wns], [--Wns], [Wns+Wb] }

NOTES: