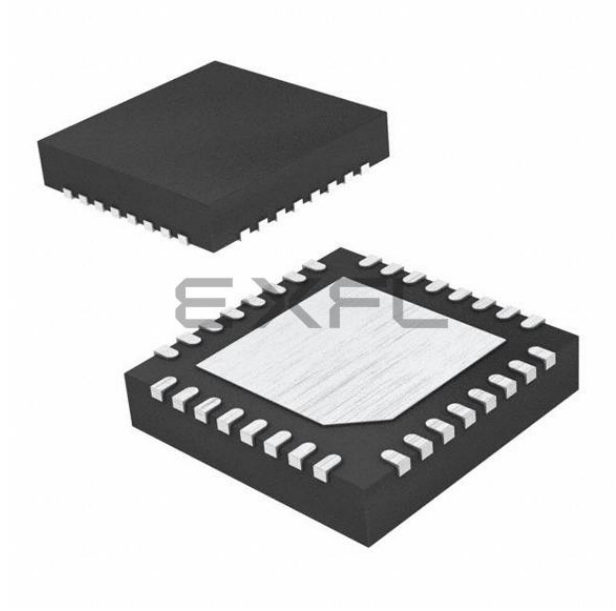
Microchip Technology - PIC24FJ128GA202T-I/MM Datasheet



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, SmartCard, SPI, UART/USART
Peripherals	AES, Brown-out Detect/Reset, DMA, I²S, HLVD, POR, PWM, WDT
Number of I/O	21
Program Memory Size	128KB (43K x 24)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 10x10b/12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-VQFN Exposed Pad
Supplier Device Package	28-QFN-S (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic24fj128ga202t-i-mm

3.0 CPU

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 on the CPU, refer to the *"dsPIC33/PIC24 Family Reference Manual"*, **"CPU with Extended Data Space (EDS)"** (DS39732). The information in this data sheet supersedes the information in the FRM.

The PIC24F CPU has a 16-bit (data) modified Harvard architecture with an enhanced instruction set and a 24-bit instruction word with a variable length opcode field. The Program Counter (PC) is 23 bits wide and addresses up to 4M instructions of user program memory space. A single-cycle instruction prefetch mechanism is used to help maintain throughput and provides predictable execution. All instructions execute in a single cycle, with the exception of instructions that change the program flow, the double-word move (`MOV.D`) instruction and the table instructions. Overhead-free program loop constructs are supported using the `REPEAT` instructions, which are interruptible at any point.

PIC24F devices have sixteen, 16-bit Working registers in the programmer's model. Each of the Working registers can act as a Data, Address or Address Offset register. The 16th Working register (W15) operates as a Software Stack Pointer (SSP) for interrupts and calls.

The lower 32 Kbytes of the Data Space (DS) can be accessed linearly. The upper 32 Kbytes of the Data Space are referred to as Extended Data Space to which the extended data RAM, EPMP memory space or program memory can be mapped.

The Instruction Set Architecture (ISA) has been significantly enhanced beyond that of the PIC18, but maintains an acceptable level of backward compatibility. All PIC18 instructions and addressing modes are supported, either directly, or through simple macros. Many of the ISA enhancements have been driven by compiler efficiency needs.

The core supports Inherent (no operand), Relative, Literal and Memory Direct Addressing modes along with three groups of addressing modes. All modes support Register Direct and various Register Indirect modes. Each group offers up to seven addressing modes. Instructions are associated with predefined addressing modes depending upon their functional requirements.

For most instructions, the core is capable of executing a data (or program data) memory read, a Working register (data) read, a data memory write and a program (instruction) memory read per instruction cycle. As a result, three parameter instructions can be supported, allowing trinary operations (that is, A + B = C) to be executed in a single cycle.

A high-speed, 17-bit x 17-bit multiplier has been included to significantly enhance the core arithmetic capability and throughput. The multiplier supports Signed, Unsigned and Mixed mode, 16-bit x 16-bit or 8-bit x 8-bit, integer multiplication. All multiply instructions execute in a single cycle.

The 16-bit ALU has been enhanced with integer divide assist hardware that supports an iterative non-restoring divide algorithm. It operates in conjunction with the `REPEAT` instruction looping mechanism and a selection of iterative divide instructions to support 32-bit (or 16-bit), divided by 16-bit, integer signed and unsigned division. All divide operations require 19 cycles to complete but are interruptible at any cycle boundary.

The PIC24F has a vectored exception scheme with up to 8 sources of non-maskable traps and up to 118 interrupt sources. Each interrupt source can be assigned to one of seven priority levels.

A block diagram of the CPU is shown in Figure 3-1.

3.1 Programmer's Model

The programmer's model for the PIC24F is shown in Figure 3-2. All registers in the programmer's model are memory-mapped and can be manipulated directly by instructions.

A description of each register is provided in Table 3-1. All registers associated with the programmer's model are memory-mapped.

TABLE 4-4: ICN REGISTER MAP

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
CNPD1	0056	CN15PDE	CN14PDE	CN13PDE	CN12PDE	CN11PDE	CN10PDE(1)	CN9PDE(1)	CN8PDE(1)	CN7PDE	CN6PDE	CN5PDE	CN4PDE	CN3PDE	CN2PDE	CN1PDE	CN0PDE	0000
CNPD2	0058	—	CN30PDE	CN29PDE	CN28PDE(1)	CN27PDE	CN26PDE(1)	CN25PDE(1)	CN24PDE	CN23PDE	CN22PDE	CN21PDE	CN20PDE(1)	CN19PDE(1)	CN18PDE(1)	CN17PDE(1)	CN16PDE	0000
CNPD3	005A	—	—	—	—	—	—	—	—	—	—	—	CN36PDE(1)	CN35PDE(1)	CN34PDE(1)	CN33PDE(1)	—	0000
CNEN1	0062	CN15IE	CN14IE	CN13IE	CN12IE	CN11IE	CN10IE(1)	CN9IE(1)	CN8IE(1)	CN7IE	CN6IE	CN5IE	CN4IE	CN3IE	CN2IE	CN1IE	CN0IE	0000
CNEN2	0064	—	CN30IE	CN29IE	CN28IE(1)	CN27IE	CN26IE(1)	CN25IE(1)	CN24IE	CN23IE	CN22IE	CN21IE	CN20IE(1)	CN19IE(1)	CN18IE(1)	CN17IE(1)	CN16IE	0000
CNEN3	0066	—	—	—	—	—	—	—	—	—	—	—	CN36IE(1)	CN35IE(1)	CN34IE(1)	CN33IE(1)	—	0000
CNPU1	006E	CN15PUE	CN14PUE	CN13PUE	CN12PUE	CN11PUE	CN10PUE(1)	CN9PUE(1)	CN8PUE(1)	CN7PUE	CN6PUE	CN5PUE	CN4PUE	CN3PUE	CN2PUE	CN1PUE	CN0PUE	0000
CNPU2	0070	—	CN30PUE	CN29PUE	CN28PUE(1)	CN27PUE	CN26PUE(1)	CN25PUE(1)	CN24PUE	CN23PUE	CN22PUE	CN21PUE	CN20PUE(1)	CN19PUE(1)	CN18PUE(1)	CN17PUE(1)	CN16PUE	0000
CNPU3	0072	—	—	—	—	—	—	—	—	—	—	—	CN36PUE(1)	CN35PUE(1)	CN34PUE(1)	CN33PUE(1)	—	0000

Legend: — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

Note 1: These bits are unimplemented in 28-pin devices, read as '0'.

4.2.5 EXTENDED DATA SPACE (EDS)

The Extended Data Space (EDS) allows PIC24F devices to address a much larger range of data than would otherwise be possible with a 16-bit address range. EDS includes any additional internal data memory not directly accessible by the lower 32-Kbyte data address space and any external memory through the Enhanced Parallel Master Port (EPMP).

In addition, EDS also allows read access to the program memory space. This feature is called Program Space Visibility (PSV) and is discussed in detail in **Section 4.3.3 "Reading Data from Program Memory Using EDS"**.

Figure 4-4 displays the entire EDS space. The EDS is organized as pages, called EDS pages, with one page equal to the size of the EDS window (32 Kbytes). A particular EDS page is selected through the Data Space Read register (DSRPAG) or Data Space Write register (DSWPAG). For PSV, only the DSRPAG register is used. The combination of the DSRPAG register value and the 16-bit wide data address forms a 24-bit Effective Address (EA).

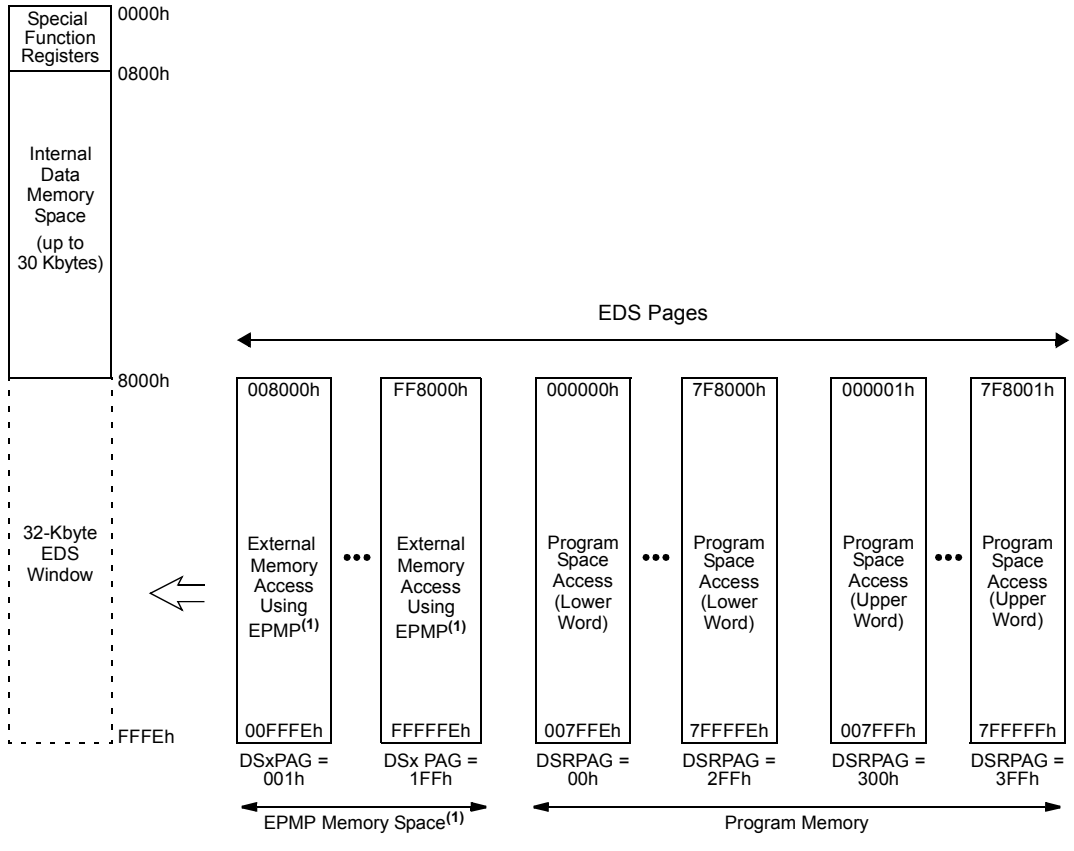
The data addressing range of PIC24FJ128GA204 family devices depends on the version of the Enhanced Parallel Master Port implemented on a particular device; this is, in turn, a function of device pin count. Table 4-33 lists the total memory accessible by each of the devices in this family. For more details on accessing external memory using EPMP, refer to the *"dsPIC33/PIC24 Family Reference Manual"*, **"Enhanced Parallel Master Port (EPMP)"** (DS39730).

TABLE 4-33: TOTAL ACCESSIBLE DATA MEMORY

Family	Internal RAM	External RAM Access Using EPMP
PIC24FJXXXGA204	8K	Up to 16 Mbytes
PIC24FJXXXGA202	8K	Up to 64K

Note: Accessing Page 0 in the EDS window will generate an address error trap as Page 0 is the base data memory (data locations, 0800h to 7FFFh, in the lower Data Space).

FIGURE 4-4: EXTENDED DATA SPACE (EDS)



Note 1: The range of addressable memory available is dependent on the device pin count and EPMP implementation.

TABLE 4-34: EDS MEMORY ADDRESS WITH DIFFERENT PAGES AND ADDRESSES

DSRPAG (Data Space Read Register)	DSWPAG (Data Space Write Register)	Source/Destination Address While Indirect Addressing	24-Bit EA Pointing to EDS	Comment
x(1)	x(1)	0000h to 1FFFh	000000h to 001FFFh	Near Data Space(2)
		2000h to 7FFFh	002000h to 007FFFh	
001h	001h	8000h to FFFFh	008000h to 00FFFEh	EPMP Memory Space
002h	002h		010000h to 017FFEh	
003h • • • • • 1FFh	003h • • • • • 1FFh		018000h to 0187FEh • • • • FF8000h to FFFFFEh	
000h	000h		Invalid Address	Address Error Trap(3)

Note 1: If the source/destination address is below 8000h, the DSRPAG and DSWPAG registers are not considered.

2: This Data Space can also be accessed by Direct Addressing.

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

4.2.6 SOFTWARE STACK

Apart from its use as a Working register, the W15 register in PIC24F devices is also used as a Software Stack Pointer (SSP). The pointer always points to the first available free word and grows from lower to higher addresses. It predecrements for stack pops and post-increments for stack pushes, as shown in Figure 4-7. Note that for a PC push during any CALL instruction, the MSB of the PC is zero-extended before the push, ensuring that the MSB is always clear.

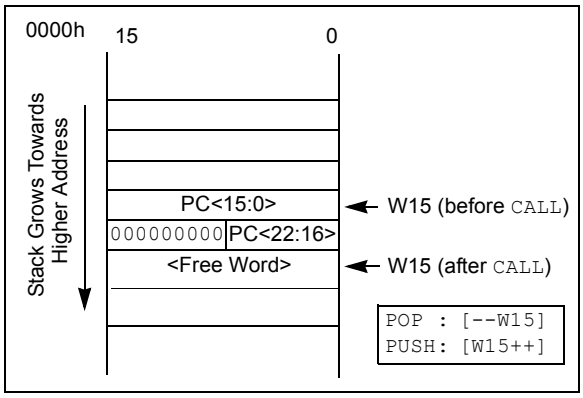
Note:	A PC push during exception processing will concatenate the SRL Register to the MSB of the PC prior to the push.

The Stack Pointer Limit Value (SPLIM) register, associated with the Stack Pointer, sets an upper address boundary for the stack. SPLIM is uninitialized at Reset. As is the case for the Stack Pointer, SPLIM<0> is forced to '0' as all stack operations must be word-aligned. Whenever an EA is generated using W15 as a Source or Destination Pointer, the resulting address is compared with the value in SPLIM. If the contents of the Stack Pointer (W15) and the SPLIM register are equal, and a push operation is performed, a stack error trap will not occur. The stack error trap will occur on a subsequent push operation. Thus, for example, if it is desirable to cause a stack error trap when the stack grows beyond address, 2000h in RAM, initialize the SPLIM with the value, 1FFEh.

Similarly, a Stack Pointer underflow (stack error) trap is generated when the Stack Pointer address is found to be less than 0800h. This prevents the stack from interfering with the SFR space.

A write to the SPLIM register should not be immediately followed by an indirect read operation using W15.

FIGURE 4-7: CALL STACK FRAME



6.6.2 PROGRAMMING A SINGLE WORD OF FLASH PROGRAM MEMORY

If a Flash location has been erased, it can be programmed using Table Write instructions to write an instruction word (24-bit) into the write latch. The TBLPAG register is loaded with the 8 Most Significant Bytes (MSBs) of the Flash address. The TBLWTL and TBLWTH instructions write the desired data into the write latches and specify the lower 16 bits of the program memory address to write to. To configure the NVMCON register for a word write, set the NVMOPx bits (NVMCON<3:0>) to '0011'. The write is performed by executing the unlock sequence and setting the WR bit (see Example 6-5). An equivalent procedure in 'C' compiler language, using the MPLAB® C30 compiler and built-in hardware functions, is shown in Example 6-6.

EXAMPLE 6-5: PROGRAMMING A SINGLE WORD OF FLASH PROGRAM MEMORY

```
; Setup a pointer to data Program Memory
    MOV     #tblpage(PROG_ADDR), W0       ;
    MOV     W0, TBLPAG                    ;Initialize PM Page Boundary SFR
    MOV     #tbloffset(PROG_ADDR), W0     ;Initialize a register with program memory address

    MOV     #LOW_WORD_N, W2               ;
    MOV     #HIGH_BYTE_N, W3              ;
    TBLWTL  W2, [W0]                      ; Write PM low word into program latch
    TBLWTH  W3, [W0++]                    ; Write PM high byte into program latch

; Setup NVMCON for programming one word to data Program Memory
    MOV     #0x4003, W0                   ;
    MOV     W0, NVMCON                    ; Set NVMOP bits to 0011

    DISI    #5                            ; Disable interrupts while the KEY sequence is written
    MOV.B   #0x55, W0                     ; Write the key sequence
    MOV     W0, NVMKEY
    MOV.B   #0xAA, W0
    MOV     W0, NVMKEY
    BSET    NVMCON, #WR                   ; Start the write cycle
    NOP                                   ; Required delays
    NOP
```

EXAMPLE 6-6: PROGRAMMING A SINGLE WORD OF FLASH PROGRAM MEMORY ('C' LANGUAGE CODE)

```
// C example using MPLAB C30
unsigned int offset;
unsigned long progAddr = 0xXXXXXX;            // Address of word to program
unsigned int progDataL = 0xXXXX;              // Data to program lower word
unsigned char progDataH = 0xXX;               // Data to program upper byte

//Set up NVMCON for word programming
NVMCON = 0x4003;                              // Initialize NVMCON

//Set up pointer to the first memory location to be written
TBLPAG = progAddr>>16;                        // Initialize PM Page Boundary SFR
offset = progAddr & 0xFFFF;                   // Initialize lower word of address

//Perform TBLWT instructions to write latches
__builtin_tblwtl(offset, progDataL);          // Write to address low word
__builtin_tblwth(offset, progDataH);          // Write to upper byte
asm("DISI #5");                               // Block interrupts with priority <7
                                              // for next 5 instructions
__builtin_write_NVM();                        // C30 function to perform unlock
                                              // sequence and set WR
```

REGISTER 7-1: RCON: RESET CONTROL REGISTER (CONTINUED)

bit 5 **SWDTEN:** Software Enable/Disable of WDT bit(4)
1 = WDT is enabled
0 = WDT is disabled

bit 4 **WDTO:** Watchdog Timer Time-out Flag bit(1)
1 = WDT time-out has occurred
0 = WDT time-out has not occurred

bit 3 **SLEEP:** Wake from Sleep Flag bit(1)
1 = Device has been in Sleep mode
0 = Device has not been in Sleep mode

bit 2 **IDLE:** Wake from Idle Flag bit(1)
1 = Device has been in Idle mode
0 = Device has not been in Idle mode

bit 1 **BOR:** Brown-out Reset Flag bit(1)
1 = A Brown-out Reset has occurred (also set after a Power-on Reset)
0 = A Brown-out Reset has not occurred

bit 0 **POR:** Power-on Reset Flag bit(1)
1 = A Power-on Reset has occurred
0 = A Power-on Reset has not occurred

Note 1: All of the Reset status bits may be set or cleared in software. Setting one of these bits in software does not cause a device Reset.

2: If the $\overline{\text{LPCFG}}$ Configuration bit is '1' (unprogrammed), the retention regulator is disabled and the RETEN bit has no effect.

3: Re-enabling the regulator after it enters Standby mode will add a delay, T_{VREG}, when waking up from Sleep. Applications that do not use the voltage regulator should set this bit to prevent this delay from occurring.

4: If the FWDTEN Configuration bit is '1' (unprogrammed), the WDT is always enabled**,** regardless of the SWDTEN bit setting.

REGISTER 7-2: RCON2: RESET AND SYSTEM CONTROL REGISTER 2

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	r-0	R/CO-1	R/CO-1	R/CO-1	R/CO-0
—	—	—	—	VDDBOR(1)	VDDPOR(1,2)	VBPOR(1,3)	VBAT(1)
bit 7							bit 0

Legend:	CO = Clearable Only 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 15-5 **Unimplemented:** Read as '0'

bit 4 **Reserved:** Maintain as '0'

bit 3 **VDDBOR:** VDD Brown-out Reset Flag bit(1)
1 = A VDD Brown-out Reset has occurred (set by hardware)
0 = A VDD Brown-out Reset has not occurred

bit 2 **VDDPOR:** VDD Power-on Reset Flag bit(1,2)
1 = A VDD Power-on Reset has occurred (set by hardware)
0 = A VDD Power-on Reset has not occurred

bit 1 **VBPOR:** VBPOR Flag bit(1,3)
1 = A VBAT POR has occurred (no battery is connected to the VBAT pin or VBAT power below the Deep Sleep Semaphore register retention level is set by hardware)
0 = A VBAT POR has not occurred

bit 0 **VBAT:** VBAT Flag bit(1)
1 = A POR exit has occurred while power was applied to the VBAT pin (set by hardware)
0 = A POR exit from VBAT has not occurred

Note 1: This bit is set in hardware only; it can only be cleared in software.
2: This bit indicates a VDD Power-on Reset. Setting the POR bit (RCON<0>) indicates a VCORE Power-on Reset.
3: This bit is set when the device is originally powered up, even if power is present on VBAT.

TABLE 7-1: RESET FLAG BIT OPERATION

Flag Bit	Setting Event	Clearing Event
TRAPR (RCON<15>)	Trap Conflict Event	POR
IOPUWR (RCON<14>)	Illegal Opcode or Uninitialized W Register Access	POR
CM (RCON<9>)	Configuration Mismatch Reset	POR
EXTR (RCON<7>)	$\overline{\text{MCLR}}$ Reset	POR
SWR (RCON<6>)	RESET Instruction	POR
WDTO (RCON<4>)	WDT Time-out	CLRWDT, PWRSAV Instruction, POR
SLEEP (RCON<3>)	PWRSAV #0 Instruction	POR
DPSLP (RCON<10>)	PWRSAV #0 Instruction while DSEN bit is Set	POR
IDLE (RCON<2>)	PWRSAV #1 Instruction	POR
BOR (RCON<1>)	POR, BOR	—
POR (RCON<0>)	POR	—

Note: All Reset flag bits may be set or cleared by the user software.

8.3 Interrupt Control and Status Registers

The PIC24FJ128GA204 family of devices implements a total of 43 registers for the interrupt controller:

- INTCON1
- INTCON2
- IFS0 through IFS7
- IEC0 through IEC7
- IPC0 through IPC16, IPC18 through IPC22, IPC26 and IPC29
- INTTREG

Global interrupt control functions are controlled from INTCON1 and INTCON2. INTCON1 contains the Interrupt Nesting Disable (NSTDIS) bit, as well as the control and status flags for the processor trap sources. The INTCON2 register controls the external interrupt request signal behavior and the use of the Alternate Interrupt Vector Table (AIVT).

The IFSx registers maintain all of the interrupt request flags. Each source of interrupt has a status bit, which is set by the respective peripherals or an external signal and is cleared via software.

The IECx registers maintain all of the interrupt enable bits. These control bits are used to individually enable interrupts from the peripherals or external signals.

The IPCx registers are used to set the Interrupt Priority Level (IPL) for each source of interrupt. Each user interrupt source can be assigned to one of eight priority levels.

The INTTREG register contains the associated interrupt vector number and the new CPU Interrupt Priority Level, which are latched into the Vector Number (VECNUM<7:0>) and the Interrupt Priority Level (ILR<3:0>) bit fields in the INTTREG register. The new Interrupt Priority Level is the priority of the pending interrupt.

The interrupt sources are assigned to the IFSx, IECx and IPCx registers in the order of their vector numbers, as shown in Table 8-2. For example, the INT0 (External Interrupt 0) is shown as having a vector number and a natural order priority of 0. Thus, the INT0IF status bit is found in IFS0<0>, the INT0IE enable bit in IEC0<0> and the INT0IP<2:0> priority bits in the first position of IPC0 (IPC0<2:0>).

Although they are not specifically part of the interrupt control hardware, two of the CPU Control registers contain bits that control interrupt functionality. The ALU STATUS Register (SR) contains the IPL<2:0> bits (SR<7:5>). These indicate the current CPU Interrupt Priority Level. The user can change the current CPU priority level by writing to the IPLx bits.

The CORCON register contains the IPL3 bit, which together with the IPL<2:0> bits, indicates the current CPU priority level. IPL3 is a read-only bit so that trap events cannot be masked by the user software.

The interrupt controller has the Interrupt Controller Test register, INTTREG, which displays the status of the interrupt controller. When an interrupt request occurs, its associated vector number and the new Interrupt Priority Level are latched into INTTREG. This information can be used to determine a specific interrupt source if a generic ISR is used for multiple vectors (such as when ISR remapping is used in bootloader applications) or to check if another interrupt is pending while in an ISR.

All Interrupt registers are described in Register 8-1 through Register 8-45 in the succeeding pages.

REGISTER 8-10: IFS5: INTERRUPT FLAG STATUS REGISTER 5

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	SPI3TXIF	SPI3IF	U4TXIF	U4RXIF
bit 15							bit 8

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
U4ERIF	—	I2C2BCIF	I2C1BCIF	U3TXIF	U3RXIF	U3ERIF	—
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 15-12 **Unimplemented:** Read as '0'

bit 11 **SPI3TXIF:** SPI3 Transmit Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 10 **SPI3IF:** SPI3 General Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 9 **U4TXIF:** UART4 Transmitter Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 8 **U4RXIF:** UART4 Receiver Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 7 **U4ERIF:** UART4 Error Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 6 **Unimplemented:** Read as '0'

bit 5 **I2C2BCIF:** I2C2 Bus Collision Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 4 **I2C1BCIF:** I2C1 Bus Collision Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 3 **U3TXIF:** UART3 Transmitter Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 2 **U3RXIF:** UART3 Receiver Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 1 **U3ERIF:** UART3 Error Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 0 **Unimplemented:** Read as '0'

REGISTER 8-31: IPC10: INTERRUPT PRIORITY CONTROL REGISTER 10

U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-0	R/W-0
—	—	—	—	—	OC6IP2	OC6IP1	OC6IP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	OC5IP2	OC5IP1	OC5IP0	—	IC6IP2	IC6IP1	IC6IP0
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 15-11 **Unimplemented:** Read as '0'

bit 10-8 **OC6IP<2:0>:** Output Compare Channel 6 Interrupt Priority bits
111 = Interrupt is Priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is Priority 1
000 = Interrupt source is disabled

bit 7 **Unimplemented:** Read as '0'

bit 6-4 **OC5IP<2:0>:** Output Compare Channel 5 Interrupt Priority bits
111 = Interrupt is Priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is Priority 1
000 = Interrupt source is disabled

bit 3 **Unimplemented:** Read as '0'

bit 2-0 **IC6IP<2:0>:** Input Capture Channel 6 Interrupt Priority bits
111 = Interrupt is Priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is Priority 1
000 = Interrupt source is disabled

REGISTER 11-1: ANSA: PORTA ANALOG FUNCTION SELECTION REGISTER

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-1	R/W-1	R/W-1	R/W-1
—	—	—	—	ANSA<3:0>			
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 15-4 **Unimplemented:** Read as ‘0’

bit 3-0 **ANSA<3:0>:** PORTA Analog Function Selection bits
1 = Pin is configured in Analog mode; I/O port read is disabled
0 = Pin is configured in Digital mode; I/O port read is enabled

REGISTER 11-2: ANSB: PORTB ANALOG FUNCTION SELECTION REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	U-0	U-0	R/W-1	U-0
ANSB<15:12>				—	—	ANSB9	—
bit 15							bit 8

U-0	R/W-1	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
—	ANSB6	—	—	ANSB<3:0>			
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 15-12 **ANSB<15:12>:** PORTB Analog Function Selection bits
1 = Pin is configured in Analog mode; I/O port read is disabled
0 = Pin is configured in Digital mode; I/O port read is enabled

bit 11-10 **Unimplemented:** Read as ‘0’

bit 9 **ANSB9:** PORTB Analog Function Selection bit
1 = Pin is configured in Analog mode; I/O port read is disabled
0 = Pin is configured in Digital mode; I/O port read is enabled

bit 8-7 **Unimplemented:** Read as ‘0’

bit 6 **ANSB6:** PORTB Analog Function Selection bit
1 = Pin is configured in Analog mode; I/O port read is disabled
0 = Pin is configured in Digital mode; I/O port read is enabled

bit 5-4 **Unimplemented:** Read as ‘0’

bit 3-0 **ANSB<3:0>:** PORTB Analog Function Selection bits
1 = Pin is configured in Analog mode; I/O port read is disabled
0 = Pin is configured in Digital mode; I/O port read is enabled

REGISTER 11-29: RPOR6: PERIPHERAL PIN SELECT OUTPUT REGISTER 6

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP13R5	RP13R4	RP13R3	RP13R2	RP13R1	RP13R0
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	RP12R5	RP12R4	RP12R3	RP12R2	RP12R1	RP12R0
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 13-8 **RP13R<5:0>:** RP13 Output Pin Mapping bits
Peripheral Output Number n is assigned to pin, RP13 (see Table 11-4 for peripheral function numbers).

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **RP12R<5:0>:** RP12 Output Pin Mapping bits
Peripheral Output Number n is assigned to pin, RP12 (see Table 11-4 for peripheral function numbers).

REGISTER 11-30: RPOR7: PERIPHERAL PIN SELECT OUTPUT REGISTER 7

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP15R5	RP15R4	RP15R3	RP15R2	RP15R1	RP15R0
bit 15							bit 8

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP14R5	RP14R4	RP14R3	RP14R2	RP14R1	RP14R0
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 13-8 **RP15R<5:0>:** RP15 Output Pin Mapping bits
Peripheral Output Number n is assigned to pin, RP15 (see Table 11-4 for peripheral function numbers).

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **RP14R<5:0>:** RP14 Output Pin Mapping bits
Peripheral Output Number n is assigned to pin, RP14 (see Table 11-4 for peripheral function numbers).

REGISTER 14-1: ICxCON1: INPUT CAPTURE x CONTROL REGISTER 1

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0
—	—	ICSIDL	ICTSEL2	ICTSEL1	ICTSEL0	—	—
bit 15							bit 8

U-0	R/W-0	R/W-0	R-0, HSC	R-0, HSC	R/W-0	R/W-0	R/W-0
—	ICI1	ICI0	ICOV	ICBNE	ICM2(1)	ICM1(1)	ICM0(1)
bit 7							bit 0

Legend:	HSC = Hardware Settable/Clearable 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 15-14 **Unimplemented:** Read as '0'

bit 13 **ICSIDL:** Input Capture x Module Stop in Idle Control bit
1 = Input capture module halts in CPU Idle mode
0 = Input capture module continues to operate in CPU Idle mode

bit 12-10 **ICTSEL<2:0>:** Input Capture x Timer Select bits
111 = System clock (FOSC/2)
110 = Reserved
101 = Reserved
100 = Timer1
011 = Timer5
010 = Timer4
001 = Timer2
000 = Timer3

bit 9-7 **Unimplemented:** Read as '0'

bit 6-5 **ICI<1:0>:** Select Number of Captures per Interrupt bits
11 = Interrupt on every fourth capture event
10 = Interrupt on every third capture event
01 = Interrupt on every second capture event
00 = Interrupt on every capture event

bit 4 **ICOV:** Input Capture x Overflow Status Flag bit (read-only)
1 = Input capture overflow has occurred
0 = No input capture overflow has occurred

bit 3 **ICBNE:** Input Capture x Buffer Empty Status bit (read-only)
1 = Input capture buffer is not empty, at least one more capture value can be read
0 = Input capture buffer is empty

bit 2-0 **ICM<2:0>:** Input Capture x Mode Select bits(1)
111 = Interrupt mode: Input capture functions as an interrupt pin only when the device is in Sleep or Idle mode (rising edge detect only, all other control bits are not applicable)
110 = Unused (module is disabled)
101 = Prescaler Capture mode: Capture on every 16th rising edge
100 = Prescaler Capture mode: Capture on every 4th rising edge
011 = Simple Capture mode: Capture on every rising edge
010 = Simple Capture mode: Capture on every falling edge
001 = Edge Detect Capture mode: Capture on every edge (rising and falling); ICI<1:0> bits do not control interrupt generation for this mode
000 = Input capture module is turned off

Note 1: The ICx input must also be configured to an available RPn/RPIn pin. For more information, see **Section 11.4 "Peripheral Pin Select (PPS)"**.

REGISTER 16-7: SPIxIMSKH: SPIx INTERRUPT MASK REGISTER HIGH

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RXWIEN	—	RXMSK5(1)	RXMSK4(1,4)	RXMSK3(1,3)	RXMSK2(1,2)	RXMSK1(1)	RXMSK0(1)
bit 15							bit 8

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TXWIEN	—	TXMSK5(1)	TXMSK4(1,4)	TXMSK3(1,3)	TXMSK2(1,2)	TXMSK1(1)	TXMSK0(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 **RXWIEN:** Receive Watermark Interrupt Enable bit
1 = Triggers receive buffer element watermark interrupt when RXMSK<5:0> ≤ RXELM<5:0>
0 = Disables receive buffer element watermark interrupt

bit 14 **Unimplemented:** Read as '0'

bit 13-8 **RXMSK<5:0>:** RX Buffer Mask bits(1,2,3,4)
RX mask bits; used in conjunction with the RXWIEN bit.

bit 7 **TXWIEN:** Transmit Watermark Interrupt Enable bit
1 = Triggers transmit buffer element watermark interrupt when TXMSK<5:0> = TXELM<5:0>
0 = Disables transmit buffer element watermark interrupt

bit 6 **Unimplemented:** Read as '0'

bit 5-0 **TXMSK<5:0>:** TX Buffer Mask bits(1,2,3,4)
TX mask bits; used in conjunction with the TXWIEN bit.

Note 1: Mask values higher than FIFODEPTH are not valid. The module will not trigger a match for any value in this case.
2: RXMSK2 and TXMSK2 bits are only present when FIFODEPTH = 8 or higher.
3: RXMSK3 and TXMSK3 bits are only present when FIFODEPTH = 16 or higher.
4: RXMSK4 and TXMSK4 bits are only present when FIFODEPTH = 32.

17.0 INTER-INTEGRATED CIRCUIT™ (I^2C™)

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"*, **"Inter-Integrated Circuit™ (I^2C™)"** (DS70000195). The information in this data sheet supersedes the information in the FRM.

The Inter-Integrated Circuit™ (I^2C™) module is a serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, display drivers, A/D Converters, etc.

The I^2C module supports these features:

- Independent master and slave logic
- 7-bit and 10-bit device addresses
- General call address as defined in the I^2C protocol
- Clock stretching to provide delays for the processor to respond to a slave data request
- Both 100 kHz and 400 kHz bus specifications
- Configurable address masking
- Multi-Master modes to prevent loss of messages in arbitration
- Bus Repeater mode, allowing the acceptance of all messages as a slave regardless of the address
- Automatic SCL

A block diagram of the module is shown in Figure 17-1.

17.1 Communicating as a Master in a Single Master Environment

The details of sending a message in Master mode depends on the communication protocols for the device being communicated with. Typically, the sequence of events is as follows:

1. Assert a Start condition on SDAx and SCLx.
2. Send the I^2C device address byte to the slave with a write indication.
3. Wait for and verify an Acknowledge from the slave.
4. Send the first data byte (sometimes known as the command) to the slave.
5. Wait for and verify an Acknowledge from the slave.
6. Send the serial memory address low byte to the slave.
7. Repeat Steps 4 and 5 until all data bytes are sent.
8. Assert a Repeated Start condition on SDAx and SCLx.
9. Send the device address byte to the slave with a read indication.
10. Wait for and verify an Acknowledge from the slave.
11. Enable master reception to receive serial memory data.
12. Generate an ACK or NACK condition at the end of a received byte of data.
13. Generate a Stop condition on SDAx and SCLx.

17.2 Setting Baud Rate when Operating as a Bus Master

To compute the Baud Rate Generator reload value, use Equation 17-1.

EQUATION 17-1: COMPUTING BAUD RATE RELOAD VALUE(1)

$$I2CxBRG = \left(\left(\frac{1}{FSCL} - PGDx\right) \times \frac{FCY}{2}\right) - 2$$

Note 1: Based on FCY = FOSC/2; Doze mode and PLL are disabled.

17.3 Slave Address Masking

The I2CxMSK register (Register 17-4) designates address bit positions as "don't care" for both 7-Bit and 10-Bit Addressing modes. Setting a particular bit location (= 1) in the I2CxMSK register causes the slave module to respond, whether the corresponding address bit value is a '0' or a '1'. For example, when I2CxMSK is set to '0010000000', the slave module will detect both addresses, '0000000000' and '0010000000'.

To enable address masking, the Intelligent Peripheral Management Interface (IPMI) must be disabled by clearing the STRICT bit (I2CxCONL<11>).

Note: As a result of changes in the I²C™ protocol, the addresses in Table 17-1 are reserved and will not be Acknowledged in Slave mode. This includes any address mask settings that include any of these addresses.

TABLE 17-1: I2Cx RESERVED ADDRESSES(1)

Slave Address	R/$\overline{W}$ Bit	Description
0000 000	0	General Call Address(2)
0000 000	1	Start Byte
0000 001	x	Cbus Address
0000 01x	x	Reserved
0000 1xx	x	HS Mode Master Code
1111 0xx	x	10-Bit Slave Upper Byte(3)
1111 1xx	x	Reserved

Note 1: The address bits listed here will never cause an address match independent of address mask settings.
2: This address will be Acknowledged only if GCEN = 1.
3: A match on this address can only occur on the upper byte in 10-Bit Addressing mode.

REGISTER 18-5: UxSCCON: UARTx SMART CARD CONTROL REGISTER(1)

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

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	TXRPT1(2)	TXRPT0(2)	CONV	T0PD(2)	PTRCL	SCEN
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 15-6 **Unimplemented:** Read as '0'

bit 5-4 **TXRPT<1:0>:** Transmit Repeat Selection bits(2)
- 11 = Retransmits the error byte four times
- 10 = Retransmits the error byte three times
- 01 = Retransmits the error byte twice
- 00 = Retransmits the error byte once

bit 3 **CONV:** Logic Convention Selection bit
- 1 = Inverse logic convention
- 0 = Direct logic convention

bit 2 **T0PD:** Pull-Down Duration for T = 0 Error Handling bit(2)
- 1 = 2 ETU
- 0 = 1 ETU

bit 1 **PTRCL:** Smart Card Protocol Selection bit
- 1 = T = 1
- 0 = T = 0

bit 0 **SCEN:** Smart Card Mode Enable bit
- 1 = Smart Card mode is enabled if UARTEN (UxMODE<15>) = 1
- 0 = Smart Card mode is disabled

Note 1: This register is only available for UART1 and UART2.

2: These bits are applicable to T = 0 only, see the PTRCL bit (UxSCCON<1>).

NOTES:

REGISTER 22-4: CRYOTP: CRYPTOGRAPHIC OTP PAGE PROGRAM CONTROL REGISTER

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

R/HSC-x(1)	R/W-0(1)	R/S/HC-1	R/W-0(1)	R/W-0(1)	R/W-0(1)	R/W-0(1)	R/S/HC-0(2)
PGMTST	OTPIE	CRYREAD(3,4)	KEYPG3	KEYPG2	KEYPG1	KEYPG0	CRYWR(3,4)
bit 7							bit 0

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

bit 15-8 **Unimplemented:** Read as '0'

bit 7 **PGMTST:** Key Storage/Configuration Program Test bit(1)

This bit mirrors the state of the TSTPGM bit and is used to test the programming of the secure OTP array after programming.

1 = TSTPGM (CFGPAGE<30>) is programmed ('1')
0 = TSTPGM is not programmed ('0')

bit 6 **OTPIE:** Key Storage/Configuration Program Interrupt Enable bit(1)

1 = Generates an interrupt when the current programming or read operation completes
0 = Does not generate an interrupt when the current programming or read operation completes; software must poll the CRYWR, CRYREAD or CRYBSY bit to determine when the current programming operation is complete

bit 5 **CRYREAD:** Cryptographic Key Storage/Configuration Read bit(3,4)

1 = This bit is set to start a read operation; read operation is in progress while this bit is set and CRYGO = 1
0 = Read operation has completed

bit 4-1 **KEYPG<3:0>:** Key Storage/Configuration Program Page Select bits(1)

1111
•
• = Reserved
•
1001
1000 = OTP Page 8
0111 = OTP Page 7
0110 = OTP Page 6
0101 = OTP Page 5
0100 = OTP Page 4
0011 = OTP Page 3
0010 = OTP Page 2
0001 = OTP Page 1
0000 = Configuration Page (CFGPAGE, OTP Page 0)

bit 0 **CRYWR:** Cryptographic Key Storage/Configuration Program bit(2,3,4)

1 = Programs the Key Storage/Configuration bits with the value found in CRYTXTC<63:0>
0 = Program operation has completed

Note 1: These bits are reset on systems Resets or whenever the CRYMD bit is set.

2: These bits are reset on systems Resets, when the CRYMD bit is set or when CRYGO is cleared.

3: Set this bit only when CRYON = 1 and CRYGO = 0. Do not set CRYREAD or CRYWR both, at any given time.

4: Do not clear CRYON or these bits while they are set; always allow the hardware operation to complete and clear the bit automatically.

INDEX

A

A/D
- Control Registers ... 314
- Extended DMA Operations ... 313
- Operation ... 311
- Transfer Functions
  - 10-Bit ... 330
  - 12-Bit ... 329

AC Characteristics
- A/D Conversion Timing ... 406
- A/D Module Specifications ... 405
- and Timing Parameters ... 387
- Capacitive Loading on Output Pins ... 387
- CLKO and I/O Timing Requirements ... 394
- External Clock Timing Requirements ... 388
- I^2C Bus Data (Master Mode) ... 390, 391
- I^2C Bus Data (Slave Mode) ... 393
- I^2C Bus Start/Stop Bit (Slave Mode) ... 392
- Input Capture x Timing Requirements ... 398
- Internal RC Accuracy ... 389
- Load Conditions and Requirements for Specifications ... 387
- Output Compare x Requirements ... 398
- PLL Clock Timing Specifications ... 389
- RC Oscillator Start-up Time ... 394
- Reset and Brown-out Reset Requirements ... 395
- Simple OCx/PWM Mode Requirements ... 399
- SPIx Master Mode (CKE = 0) Requirements ... 400
- SPIx Master Mode (CKE = 1) Requirements ... 401
- SPIx Slave Mode (CKE = 0) Requirements ... 402
- SPIx Slave Mode (CKE = 1) Requirements ... 404
- Timer1 External Clock Requirements ... 396
- Timer2 and Timer4 External Clock Requirements ... 397
- Timer3 and Timer5 External Clock Requirements ... 397

Alternate Interrupt Vector Table (AIVT) ... 87

Assembler
- MPASM Assembler ... 364

B

Block Diagrams
- 10-Bit A/D Converter Analog Input Model ... 328
- 12-Bit A/D Converter ... 312
- 16-Bit Asynchronous Timer3/5 ... 201
- 16-Bit Synchronous Timer2/4 ... 201
- 16-Bit Timer1 Module ... 195
- Accessing Program Space Using Table Instructions ... 64
- Addressing for Table Registers ... 75
- Buffer Address Generation in PIA Mode ... 315
- CALL Stack Frame ... 61
- Comparator Voltage Reference Module ... 337
- CPU Programmer's Model ... 29
- CRC Module ... 305
- CRC Shift Engine Detail ... 305
- Cryptographic Engine ... 289
- CTMU Connections and Internal Configuration for Capacitance Measurement ... 340
- CTMU Typical Connections and Internal Configuration for Pulse Delay Generation ... 341
- CTMU Typical Connections and Internal Configuration for Time Measurement ... 341
- Data Access from Program Space Address Generation ... 63
- Data Signal Modulator ... 257
- Direct Memory Access (DMA) ... 67
- EDS Address Generation for Read ... 59
- EDS Address Generation for Write ... 60
- Extended Data Space (EDS) ... 58
- High/Low-Voltage Detect (HLVD) ... 347
- I2Cx Module ... 238
- Individual Comparator Configurations, CREF = 0 ... 332
- Individual Comparator Configurations, CREF = 1, CVREFP = 0 ... 333
- Individual Comparator Configurations, CREF = 1, CVREFP = 1 ... 333
- Input Capture x Module ... 205
- $\overline{MCLR}$ Pin Connections ... 22
- On-Chip Regulator Connections ... 359
- Output Compare x (16-Bit Mode) ... 212
- Output Compare x (Double-Buffered, 16-Bit PWM Mode) ... 214
- PIC24F CPU Core ... 28
- PIC24FJ128GA204 Family (General) ... 13
- PSV Operation Access (Lower Word) ... 66
- PSV Operation Access (Upper Word) ... 66
- Recommended Minimum Connections ... 21
- Reset System ... 81
- RTCC Module ... 275
- Shared I/O Port Structure ... 167
- Smart Card Subsystem Connection ... 249
- SPIx Master, Frame Master Connection ... 235
- SPIx Master, Frame Slave Connection ... 236
- SPIx Master/Slave Connection (Enhanced Buffer Modes) ... 235
- SPIx Master/Slave Connection (Standard Mode) ... 234
- SPIx Module (Enhanced Mode) ... 223
- SPIx Module (Standard Mode) ... 222
- SPIx Slave, Frame Master Connection ... 236
- SPIx Slave, Frame Slave Connection ... 236
- System Clock ... 141
- Timer2/3 and Timer4/5 (32-Bit) ... 200
- Triple Comparator Module ... 331
- UARTx (Simplified) ... 246
- Watchdog Timer (WDT) ... 360

C

C Compilers
- MPLAB XC Compilers ... 364

Charge Time Measurement Unit. *See* CTMU.