Microchip Technology - DSPIC33FJ32GP202T-I/SS 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	dsPIC
Core Size	16-Bit
Speed	40 MIPs
Connectivity	I²C, IrDA, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DCI, DMA, I²S, POR, PWM, WDT
Number of I/O	21
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 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-SSOP (0.209", 5.30mm Width)
Supplier Device Package	28-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/dspic33fj32gp202t-i-ss

4.8 Interfacing Program and Data Memory Spaces

The dsPIC33FJ32GP202/204 and dsPIC33FJ16GP304 architecture uses a 24-bit wide program space and a 16-bit wide data space. The architecture is also a modified Harvard scheme, meaning that data can also be present in the program space. To use this data successfully, it must be accessed in a way that preserves the alignment of information in both spaces.

Aside from normal execution, the dsPIC33FJ32GP202/204 and dsPIC33FJ16GP304 architecture provides two methods by which program space can be accessed during operation:

- Using table instructions to access individual bytes or words anywhere in the program space
- Remapping a portion of the program space into the data space (Program Space Visibility)

Table instructions allow an application to read or write to small areas of the program memory. This capability makes the method ideal for accessing data tables that need to be updated periodically. It also allows access to all bytes of the program word. The remapping method allows an application to access a large block of data on a read-only basis, which is ideal for look ups from a large table of static data. The application can only access the least significant word of the program word.

4.8.1 ADDRESSING PROGRAM SPACE

Since the address ranges for the data and program spaces are 16 and 24 bits, respectively, a method is needed to create a 23-bit or 24-bit program address from 16-bit data registers. The solution depends on the interface method to be used.

For table operations, the 8-bit Table Page register (TBLPAG) is used to define a 32K word region within the program space. This is concatenated with a 16-bit EA to arrive at a full 24-bit program space address. In this format, the Most Significant bit of TBLPAG is used to determine if the operation occurs in the user memory (TBLPAG<7> = 0) or the configuration memory (TBLPAG<7> = 1).

For remapping operations, the 8-bit Program Space Visibility register (PSVPAG) is used to define a 16K word page in the program space. When the Most Significant bit of the EA is '1', PSVPAG is concatenated with the lower 15 bits of the EA to form a 23-bit program space address. Unlike table operations, this limits remapping operations strictly to the user memory area.

Table 4-25 and Figure 4-7 show how the program EA is created for table operations and remapping accesses from the data EA. Here, P<23:0> refers to a program space word, and D<15:0> refers to a data space word.

TABLE 4-25: PROGRAM SPACE ADDRESS CONSTRUCTION

Access Type	Access Space	Program Space Address				
		<23>	<22:16>	<15>	<14:1>	<0>
Instruction Access (Code Execution)	User	0	PC<22:1>			0
		0xx xxxx xxxx xxxx xxxx xxx0				
TBLRD/TBLWT (Byte/Word Read/Write)	User	TBLPAG<7:0>		Data EA<15:0>		
		0xxx xxxx		xxxx xxxx xxxx xxxx		
	Configuration	TBLPAG<7:0>		Data EA<15:0>		
		1xxx xxxx		xxxx xxxx xxxx xxxx		
Program Space Visibility (Block Remap/Read)	User	0	PSVPAG<7:0>		Data EA<14:0>[1]	
		0	xxxx xxxx		xxx xxxx xxxx xxxx	

Note 1: Data EA<15> is always '1' in this case, but is not used in calculating the program space address. Bit 15 of the address is PSVPAG<0>.

NOTES:

5.5.1 PROGRAMMING ALGORITHM FOR FLASH PROGRAM MEMORY

Programmers can program one row of program Flash memory at a time. To do this, it is necessary to erase the 8-row erase page that contains the desired row. The general process is:

1. Read eight rows of program memory (512 instructions) and store in data RAM.
2. Update the program data in RAM with the desired new data.
3. Erase the block (see Example 5-1):
    a) Set the NVMOP bits (NVMCON<3:0>) to ‘`0010`’ to configure for block erase. Set the ERASE bit (NVMCON<6>) and the WREN bit (NVMCON<14>).
    b) Write the starting address of the page to be erased into the TBLPAG and W registers.
    c) Write 0x55 to NVMKEY.
    d) Write 0xAA to NVMKEY.
    e) Set the WR bit (NVMCON<15>). The erase cycle begins and the CPU stalls for the duration of the erase cycle. When the erase is done, the WR bit is cleared automatically.
4. Write the first 64 instructions from data RAM into the program memory buffers (see Example 5-2).
5. Write the program block to Flash memory:
    a) Set the NVMOP bits to ‘`0001`’ to configure for row programming. Clear the ERASE bit and set the WREN bit.
    b) Write 0x55 to NVMKEY.
    c) Write 0xAA to NVMKEY.
    d) Set the WR bit. The programming cycle begins and the CPU stalls for the duration of the write cycle. When the write to Flash memory is done, the WR bit is cleared automatically.
6. Repeat steps 4 and 5, using the next available 64 instructions from the block in data RAM by incrementing the value in TBLPAG, until all 512 instructions are written back to Flash memory.

For protection against accidental operations, the write initiate sequence for NVMKEY must be used to allow any erase or program operation to proceed. After the programming command has been executed, the user application must wait for the programming time until programming is complete. The two instructions following the start of the programming sequence should be `NOP`s, as shown in Example 5-3.

EXAMPLE 5-1: ERASING A PROGRAM MEMORY PAGE

```
; Set up NVMCON for block erase operation
        MOV     #0x4042, W0                   ;
        MOV     W0, NVMCON                    ; Initialize NVMCON
; Init pointer to row to be ERASED
        MOV     #tblpage(PROG_ADDR), W0       ;
        MOV     W0, TBLPAG                    ; Initialize PM Page Boundary SFR
        MOV     #tbloffset(PROG_ADDR), W0     ; Initialize in-page EA[15:0] pointer
        TBLWTL  W0, [W0]                      ; Set base address of erase block
        DISI    #5                            ; Block all interrupts with priority <7
                                              ; for next 5 instructions

        MOV     #0x55, W0
        MOV     W0, NVMKEY                    ; Write the 55 key
        MOV     #0xAA, W1                     ;
        MOV     W1, NVMKEY                    ; Write the AA key
        BSET    NVMCON, #WR                   ; Start the erase sequence
        NOP                                   ; Insert two NOPs after the erase
        NOP                                   ; command is asserted
```

7.0 INTERRUPT CONTROLLER

> **Note 1:** This data sheet summarizes the features of the dsPIC33FJ32GP202/204 and dsPIC33FJ16GP304 family of devices. It is not intended to be a comprehensive reference source. To complement the information in this data sheet, refer to **Section 6. "Interrupts"** (DS70184) of the *"dsPIC33F/PIC24H Family Reference Manual"*, which is available from the Microchip web site (www.microchip.com).
>
> **2:** Some registers and associated bits described in this section may not be available on all devices. Refer to **Section 4.0 "Memory Organization"** in this data sheet for device-specific register and bit information.

The dsPIC33FJ32GP202/204 and dsPIC33FJ16GP304 interrupt controller reduces the numerous peripheral interrupt request signals to a single interrupt request signal to the dsPIC33FJ32GP202/204 and dsPIC33FJ16GP304 CPU. It has the following features:

- Up to eight processor exceptions and software traps
- Seven user-selectable priority levels
- Interrupt Vector Table (IVT) with up to 118 vectors
- A unique vector for each interrupt or exception source
- Fixed priority within a specified user priority level
- Alternate Interrupt Vector Table (AIVT) for debug support
- Fixed interrupt entry and return latencies

7.1 Interrupt Vector Table

The Interrupt Vector Table is shown in Figure 7-1. The IVT resides in program memory, starting at location 000004h. The IVT contains 126 vectors consisting of 8 nonmaskable trap vectors plus up to 118 sources of interrupt. In general, each interrupt source has its own vector. Each interrupt vector contains a 24-bit wide address. The value programmed into each interrupt vector location is the starting address of the associated Interrupt Service Routine (ISR).

Interrupt vectors are prioritized in terms of their natural priority; this priority is linked to their position in the vector table. Lower addresses generally have a higher natural priority. For example, the interrupt associated with vector 0 will take priority over interrupts at any other vector address.

dsPIC33FJ32GP202/204 and dsPIC33FJ16GP304 devices implement up to 21 unique interrupts and four nonmaskable traps. These are summarized in Table 7-1 and Table 7-2.

7.1.1 ALTERNATE INTERRUPT VECTOR TABLE

The Alternate Interrupt Vector Table (AIVT) is located after the IVT, as shown in Figure 7-1. Access to the AIVT is provided by the ALTIVT control bit (INTCON2<15>). If the ALTIVT bit is set, all interrupt and exception processes use the alternate vectors instead of the default vectors. The alternate vectors are organized in the same manner as the default vectors.

The AIVT supports debugging by providing a means to switch between an application and a support environment without requiring the interrupt vectors to be reprogrammed. This feature also enables switching between applications for evaluation of different software algorithms at run time. If the AIVT is not needed, the AIVT should be programmed with the same addresses used in the IVT.

7.2 Reset Sequence

A device Reset is not a true exception because the interrupt controller is not involved in the Reset process. The dsPIC33FJ32GP202/204 and dsPIC33FJ16GP304 device clears its registers in response to a Reset, which forces the PC to zero. The digital signal controller then begins program execution at location 0x000000. The user application can use a `GOTO` instruction at the Reset address which redirects program execution to the appropriate start-up routine.

> **Note:** Any unimplemented or unused vector locations in the IVT and AIVT should be programmed with the address of a default interrupt handler routine that contains a `RESET` instruction.

REGISTER 7-7: IFS4: INTERRUPT FLAG STATUS 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	U-0	R/W-0	U-0
—	—	—	—	—	—	U1EIF	—
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 15-2 **Unimplemented:** Read as ‘0’

bit 1 **U1EIF:** UART1 Error Interrupt Flag Status bit

1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 0 **Unimplemented:** Read as ‘0’

REGISTER 7-18: IPC16: INTERRUPT PRIORITY CONTROL REGISTER 16

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

U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	U1EIP<2:0>			—	—	—	—
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 15-7 **Unimplemented:** Read as ‘0’

bit 6-4 **U1EIP<2:0>:** UART1 Error Interrupt Priority bits

111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

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

7.5 Interrupt Setup Procedures

7.5.1 INITIALIZATION

To configure an interrupt source at initialization:

1. Set the NSTDIS bit (INTCON1<15>) if nested interrupts are not desired.
2. Select the user-assigned priority level for the interrupt source by writing the control bits in the appropriate IPCx register. The priority level will depend on the specific application and type of interrupt source. If multiple priority levels are not desired, the IPCx register control bits for all enabled interrupt sources can be programmed to the same non-zero value.

> **Note:** At a device Reset, the IPCx registers are initialized such that all user interrupt sources are assigned to priority level 4.

3. Clear the interrupt flag status bit associated with the peripheral in the associated IFSx register.
4. Enable the interrupt source by setting the interrupt enable control bit associated with the source in the appropriate IECx register.

7.5.2 INTERRUPT SERVICE ROUTINE

The method used to declare an Interrupt Service Routine (ISR) and initialize the IVT with the correct vector address depends on the programming language (C or Assembler) and the language development toolsuite used to develop the application.

In general, the user application must clear the interrupt flag in the appropriate IFSx register for the source of interrupt that the ISR handles. Otherwise, the program will re-enter the ISR immediately after exiting the routine. If the ISR is coded in assembly language, it must be terminated using a `RETFIE` instruction to unstack the saved PC value, SRL value and old CPU priority level.

7.5.3 TRAP SERVICE ROUTINE

A Trap Service Routine (TSR) is coded like an ISR, except that the appropriate trap status flag in the INTCON1 register must be cleared to avoid re-entry into the TSR.

7.5.4 INTERRUPT DISABLE

All user interrupts can be disabled using this procedure:

1. Push the current SR value onto the software stack using the `PUSH` instruction.
2. Force the CPU to priority level 7 by inclusive ORing the value OEh with SRL.

To enable user interrupts, the `POP` instruction can be used to restore the previous SR value.

> **Note:** Only user interrupts with a priority level of 7 or lower can be disabled. Trap sources (level 8-level 15) cannot be disabled.

The `DISI` instruction provides a convenient way to disable interrupts of priority levels 1-6 for a fixed period of time. Level 7 interrupt sources are not disabled by the `DISI` instruction.

REGISTER 8-1: OSCCON: OSCILLATOR CONTROL REGISTER[1,3] (CONTINUED)

bit 3 **CF:** Clock Fail Detect bit (read/clear by application)
1 = FSCM has detected clock failure
0 = FSCM has not detected clock failure

bit 2 **Unimplemented:** Read as '0'

bit 1 **LPOSCEN:** Secondary (LP) Oscillator Enable bit
1 = Enable secondary oscillator
0 = Disable secondary oscillator

bit 0 **OSWEN:** Oscillator Switch Enable bit
1 = Request oscillator switch to selection specified by NOSC<2:0> bits
0 = Oscillator switch is complete

Note 1: Writes to this register require an unlock sequence. Refer to **Section 7. "Oscillator"** (DS70186) in the *"dsPIC33F/PIC24H Family Reference Manual"* for details.

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 transition clock source between the two PLL modes.

3: This register is reset only on a Power-on Reset (POR).

11.1 Timer Resources

Many useful resources are provided on the main product page of the Microchip web site for the devices listed in this data sheet. This product page, which can be accessed using this link, contains the latest updates and additional information.

> **Note:** In the event you are not able to access the product page using the link above, enter this URL in your browser: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en530331

11.1.1 KEY RESOURCES

- **Section 11. “Timers”** (DS70205)
- Code Samples
- Application Notes
- Software Libraries
- Webinars
- All related dsPIC33F/PIC24H Family Reference Manuals Sections
- Development Tools

NOTES:

12.0 TIMER2/3 FEATURE

> **Note 1:** This data sheet summarizes the features of the dsPIC33FJ32GP202/204 and dsPIC33FJ16GP304 family of devices. It is not intended to be a comprehensive reference source. To complement the information in this data sheet, refer to **Section 11. “Timers”** (DS70205) of the *“dsPIC33F/PIC24H Family Reference Manual”*, which is available from the Microchip web site (www.microchip.com).
>
> **2:** Some registers and associated bits described in this section may not be available on all devices. Refer to **Section 4.0 “Memory Organization”** in this data sheet for device-specific register and bit information.

The Timer2/3 feature has 32-bit timers that can also be configured as two independent 16-bit timers with selectable operating modes.

As a 32-bit timer, the Timer2/3 feature permits operation in three modes:

- Two Independent 16-bit timers (Timer2 and Timer3) with all 16-bit operating modes (except Asynchronous Counter mode)
- Single 32-bit timer (Timer2/3)
- Single 32-bit synchronous counter (Timer2/3)

The Timer2/3 feature also supports:

- Timer gate operation
- Selectable Prescaler Settings
- Timer operation during Idle and Sleep modes
- Interrupt on a 32-bit Period Register Match
- Time Base for Input Capture and Output Compare Modules (Timer2 and Timer3 only)
- ADC1 Event Trigger (Timer2/3 only)

Individually, all eight of the 16-bit timers can function as synchronous timers or counters. They also offer the features listed above, except for the event trigger. The operating modes and enabled features are determined by setting the appropriate bit(s) in the T2CON and T3CON registers. T2CON registers are shown in generic form in Register 12-1. T3CON registers are shown in Register 12-2.

For 32-bit timer/counter operation, Timer2 is the least significant word (lsw), and Timer3 is the most significant word (msw) of the 32-bit timers.

> **Note:** For 32-bit operation, T3CON control bits are ignored. Only T2CON control bit is used for setup and control. Timer2 clock and gate inputs are used for the 32-bit timer modules, but an interrupt is generated with the Timer3 interrupt flags.

12.1 32-Bit Operation

To configure the Timer2/3 feature for 32-bit operation:

1. Set the corresponding T32 control bit.
2. Select the prescaler ratio for Timer2 using the TCKPS<1:0> bits.
3. Set the Clock and Gating modes using the corresponding TCS and TGATE bits.
4. Load the timer period value. PR3 contains the most significant word of the value, while PR2 contains the least significant word.
5. If interrupts are required, set the interrupt enable bit, T3IE. Use the priority bits T3IP<2:0> to set the interrupt priority. While Timer2 controls the timer, the interrupt appears as a Timer3 interrupt.
6. Set the corresponding TON bit.

The timer value at any point is stored in the register pair TMR3:TMR2. TMR3 always contains the most significant word of the count, while TMR2 contains the least significant word.

To configure any of the timers for individual 16-bit operation:

1. Clear the T32 bit corresponding to that timer.
2. Select the timer prescaler ratio using the TCKPS<1:0> bits.
3. Set the Clock and Gating modes using the TCS and TGATE bits.
4. Load the timer period value into the PRx register.
5. If interrupts are required, set the interrupt enable bit, TxIE. Use the priority bits, TxIP<2:0>, to set the interrupt priority.
6. Set the TON bit.

14.3 Output Compare Register

REGISTER 14-1: OCxCON: OUTPUT COMPARE x CONTROL REGISTER

U-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
—	—	OCSIDL	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	R-0 HC	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	OCFLT	OCTSEL	OCM<2:0>		
bit 7							bit 0

Legend:	HC = Cleared in Hardware	HS = Set in Hardware	
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 **OCSIDL:** Stop Output Compare in Idle Mode Control bit

1 = Output Compare x will halt in CPU Idle mode
0 = Output Compare x will continue to operate in CPU Idle mode

bit 12-5 **Unimplemented:** Read as ‘0’

bit 4 **OCFLT:** PWM Fault Condition Status bit

1 = PWM Fault condition has occurred (cleared in hardware only)
0 = No PWM Fault condition has occurred
(This bit is only used when OCM<2:0> = 111.)

bit 3 **OCTSEL:** Output Compare Timer Select bit

1 = Timer3 is the clock source for Compare x
0 = Timer2 is the clock source for Compare x

bit 2-0 **OCM<2:0>:** Output Compare Mode Select bits

111 = PWM mode on OCx, Fault pin enabled
110 = PWM mode on OCx, Fault pin disabled
101 = Initialize OCx pin low, generate continuous output pulses on OCx pin
100 = Initialize OCx pin low, generate single output pulse on OCx pin
011 = Compare event toggles OCx pin
010 = Initialize OCx pin high, compare event forces OCx pin low
001 = Initialize OCx pin low, compare event forces OCx pin high
000 = Output compare channel is disabled

REGISTER 16-1: I2CxCON: I2Cx CONTROL REGISTER

R/W-0	U-0	R/W-0	R/W-1 HC	R/W-0	R/W-0	R/W-0	R/W-0
I2CEN	—	I2CSIDL	SCLREL	IPMIEN	A10M	DISSLW	SMEN
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0 HC	R/W-0 HC	R/W-0 HC	R/W-0 HC	R/W-0 HC
GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

Legend:	U = Unimplemented bit, read as ‘0’		
R = Readable bit	W = Writable bit	HS = Set in hardware	HC = Cleared in hardware
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15 **I2CEN:** I2Cx Enable bit

1 = Enables the I2Cx module and configures the SDAx and SCLx pins as serial port pins
0 = Disables the I2Cx module. All I^{2}C pins are controlled by port functions

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **I2CSIDL:** Stop in Idle Mode bit

1 = Discontinue module operation when device enters an Idle mode
0 = Continue module operation in Idle mode

bit 12 **SCLREL:** SCLx Release Control bit (when operating as I^{2}C™ slave)

1 = Release SCLx clock
0 = Hold SCLx clock low (clock stretch)

If STREN = 1:
Bit is R/W (i.e., software can write ‘0’ to initiate stretch and write ‘1’ to release clock). Hardware clear at beginning of slave transmission. Hardware clear at end of slave reception.

If STREN = 0:
Bit is R/S (i.e., software can only write ‘1’ to release clock). Hardware clear at beginning of slave transmission.

bit 11 **IPMIEN:** Intelligent Peripheral Management Interface (IPMI) Enable bit

1 = IPMI mode is enabled; all addresses Acknowledged
0 = IPMI mode disabled

bit 10 **A10M:** 10-bit Slave Address bit

1 = I2CxADD is a 10-bit slave address
0 = I2CxADD is a 7-bit slave address

bit 9 **DISSLW:** Disable Slew Rate Control bit

1 = Slew rate control disabled
0 = Slew rate control enabled

bit 8 **SMEN:** SMBus Input Levels bit

1 = Enable I/O pin thresholds compliant with SMBus specification
0 = Disable SMBus input thresholds

bit 7 **GCEN:** General Call Enable bit (when operating as I^{2}C slave)

1 = Enable interrupt when a general call address is received in the I2CxRSR (module is enabled for reception)
0 = General call address disabled

bit 6 **STREN:** SCLx Clock Stretch Enable bit (when operating as I^{2}C slave)

Used in conjunction with SCLREL bit.
1 = Enable software or receive clock stretching
0 = Disable software or receive clock stretching

17.1 UART Helpful Tips

1. In multi-node direct-connect UART networks, UART receive inputs react to the complementary logic level defined by the URXINV bit (UxMODE<4>), which defines the idle state, the default of which is logic high, (i.e., URXINV = `0`). Because remote devices do not initialize at the same time, it is likely that one of the devices, because the RX line is floating, will trigger a start bit detection and will cause the first byte received after the device has been initialized to be invalid. To avoid this situation, the user should use a pull-up or pull-down resistor on the RX pin depending on the value of the URXINV bit.
   a) If URXINV = `0`, use a pull-up resistor on the RX pin.
   b) If URXINV = `1`, use a pull-down resistor on the RX pin.
2. The first character received on a wake-up from Sleep mode caused by activity on the UxRX pin of the UART module will be invalid. In Sleep mode, peripheral clocks are disabled. By the time the oscillator system has restarted and stabilized from Sleep mode, the baud rate bit sampling clock relative to the incoming UxRX bit timing is no longer synchronized, resulting in the first character being invalid. This is to be expected.

17.2 UART Resources

Many useful resources are provided on the main product page of the Microchip web site for the devices listed in this data sheet. This product page, which can be accessed using this link, contains the latest updates and additional information.

Note:	In the event you are not able to access the product page using the link above, enter this URL in your browser: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en530331

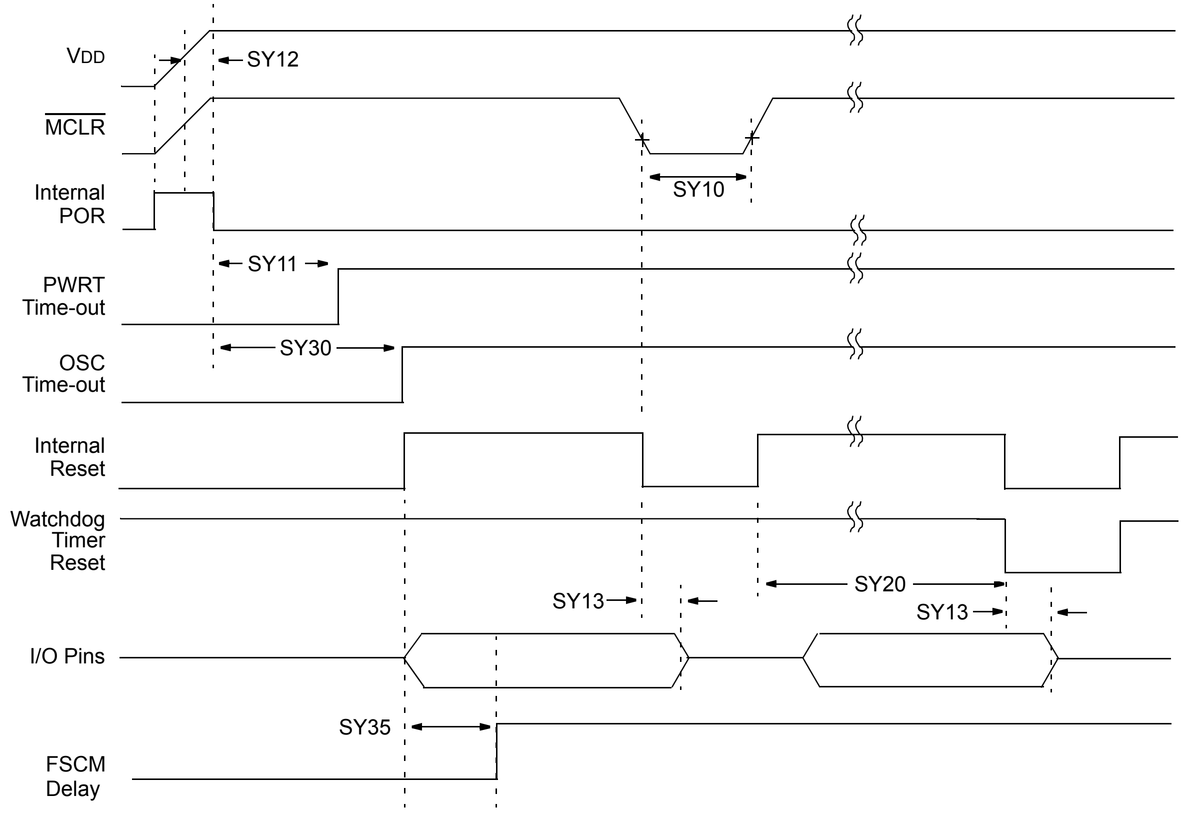
17.2.1 KEY RESOURCES

- **Section 17. “UART”** (DS70188)
- Code Samples
- Application Notes
- Software Libraries
- Webinars
- All related dsPIC33F/PIC24H Family Reference Manuals Sections
- Development Tools

TABLE 20-2: INSTRUCTION SET OVERVIEW (CONTINUED)

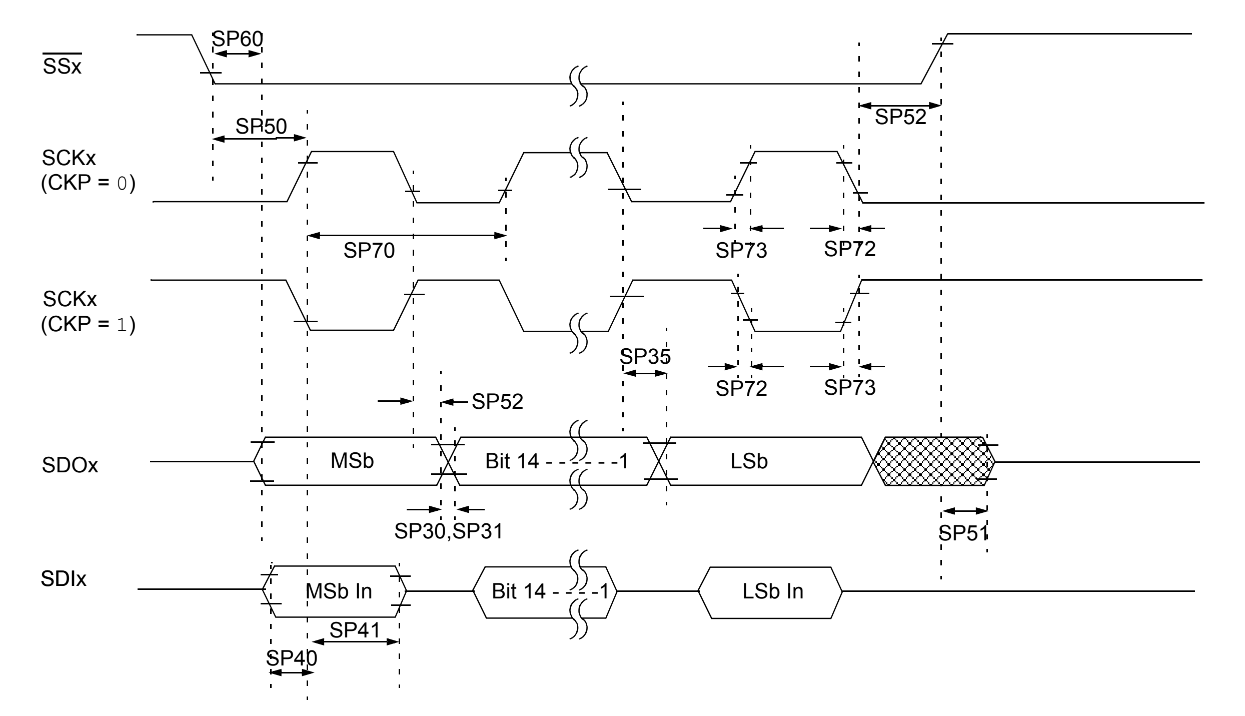
Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles	Status Flags Affected
29	DIV	DIV.S Wm,Wn	Signed 16/16-bit Integer Divide	1	18	N,Z,C,OV
		DIV.SD Wm,Wn	Signed 32/16-bit Integer Divide	1	18	N,Z,C,OV
		DIV.U Wm,Wn	Unsigned 16/16-bit Integer Divide	1	18	N,Z,C,OV
		DIV.UD Wm,Wn	Unsigned 32/16-bit Integer Divide	1	18	N,Z,C,OV
30	DIVF	DIVF Wm,Wn	Signed 16/16-bit Fractional Divide	1	18	N,Z,C,OV
31	DO	DO #lit14,Expr	Do code to PC + Expr, lit14 + 1 times	2	2	None
		DO Wn,Expr	Do code to PC + Expr, (Wn) + 1 times	2	2	None
32	ED	ED Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean Distance (no accumulate)	1	1	OA,OB,OAB, SA,SB,SAB
33	EDAC	EDAC Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean Distance	1	1	OA,OB,OAB, SA,SB,SAB
34	EXCH	EXCH Wns,Wnd	Swap Wns with Wnd	1	1	None
35	FBCL	FBCL Ws,Wnd	Find Bit Change from Left (MSb) Side	1	1	C
36	FF1L	FF1L Ws,Wnd	Find First One from Left (MSb) Side	1	1	C
37	FF1R	FF1R Ws,Wnd	Find First One from Right (LSb) Side	1	1	C
38	GOTO	GOTO Expr	Go to address	2	2	None
		GOTO Wn	Go to indirect	1	2	None
39	INC	INC f	f = f + 1	1	1	C,DC,N,OV,Z
		INC f,WREG	WREG = f + 1	1	1	C,DC,N,OV,Z
		INC Ws,Wd	Wd = Ws + 1	1	1	C,DC,N,OV,Z
40	INC2	INC2 f	f = f + 2	1	1	C,DC,N,OV,Z
		INC2 f,WREG	WREG = f + 2	1	1	C,DC,N,OV,Z
		INC2 Ws,Wd	Wd = Ws + 2	1	1	C,DC,N,OV,Z
41	IOR	IOR f	f = f .IOR. WREG	1	1	N,Z
		IOR f,WREG	WREG = f .IOR. WREG	1	1	N,Z
		IOR #lit10,Wn	Wd = lit10 .IOR. Wd	1	1	N,Z
		IOR Wb,Ws,Wd	Wd = Wb .IOR. Ws	1	1	N,Z
		IOR Wb,#lit5,Wd	Wd = Wb .IOR. lit5	1	1	N,Z
42	LAC	LAC Wso,#Slit4,Acc	Load Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
43	LNK	LNK #lit14	Link Frame Pointer	1	1	None
44	LSR	LSR f	f = Logical Right Shift f	1	1	C,N,OV,Z
		LSR f,WREG	WREG = Logical Right Shift f	1	1	C,N,OV,Z
		LSR Ws,Wd	Wd = Logical Right Shift Ws	1	1	C,N,OV,Z
		LSR Wb,Wns,Wnd	Wnd = Logical Right Shift Wb by Wns	1	1	N,Z
		LSR Wb,#lit5,Wnd	Wnd = Logical Right Shift Wb by lit5	1	1	N,Z
45	MAC	MAC Wm*Wn,Acc,Wx,Wxd,Wy,Wyd , AWB	Multiply and Accumulate	1	1	OA,OB,OAB, SA,SB,SAB
		MAC Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square and Accumulate	1	1	OA,OB,OAB, SA,SB,SAB
46	MOV	MOV f,Wn	Move f to Wn	1	1	None
		MOV f	Move f to f	1	1	N,Z
		MOV f,WREG	Move f to WREG	1	1	None
		MOV #lit16,Wn	Move 16-bit literal to Wn	1	1	None
		MOV.b #lit8,Wn	Move 8-bit literal to Wn	1	1	None
		MOV Wn,f	Move Wn to f	1	1	None
		MOV Wso,Wdo	Move Ws to Wd	1	1	None
		MOV WREG,f	Move WREG to f	1	1	None
		MOV.D Wns,Wd	Move Double from W(ns):W(ns + 1) to Wd	1	2	None
		MOV.D Ws,Wnd	Move Double from Ws to W(nd + 1):W(nd)	1	2	None
47	MOVSAC	MOVSAC Acc,Wx,Wxd,Wy,Wyd,AWB	Prefetch and store accumulator	1	1	None

FIGURE 22-4: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING CHARACTERISTICS



Note: Refer to Figure 22-1 for load conditions.

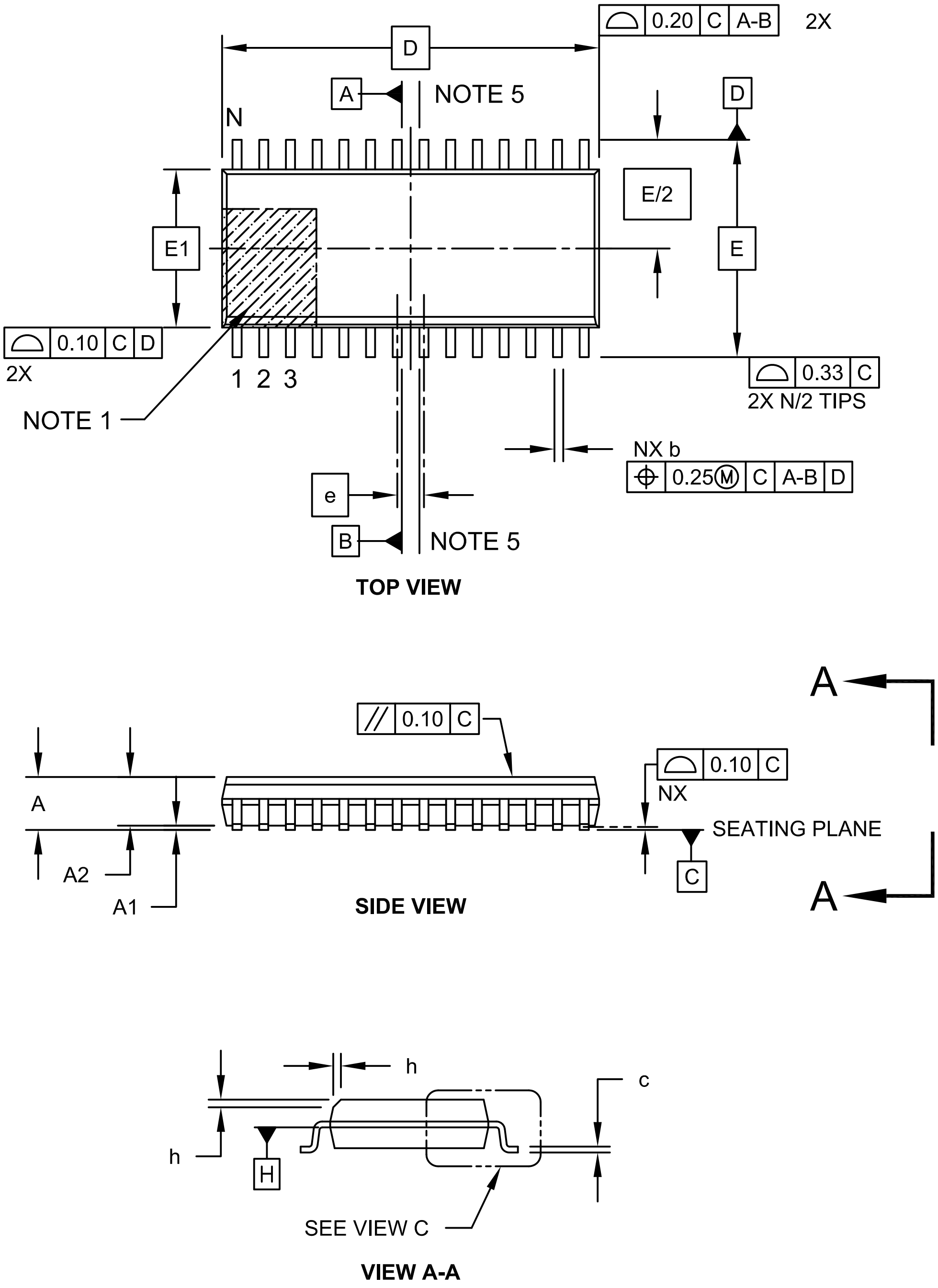
FIGURE 22-14: SPIx SLAVE MODE (FULL-DUPLEX, CKE = 1, CKP = 1, SMP = 0) TIMING CHARACTERISTICS



Note: Refer to Figure 22-1 for load conditions.

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



TOP VIEW

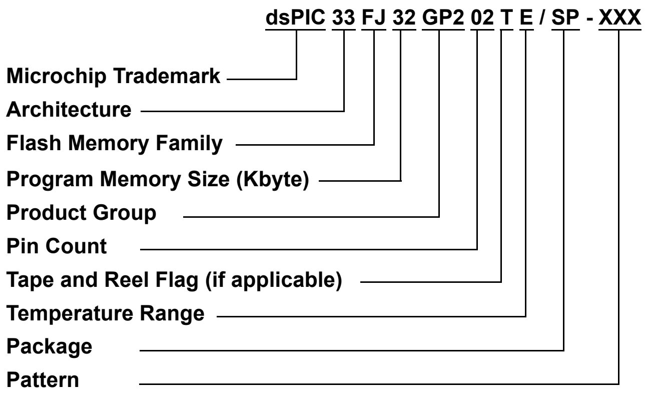
SIDE VIEW

VIEW A-A

Microchip Technology Drawing C04-052C Sheet 1 of 2

PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.



Architecture:	33	=	16-bit Digital Signal Controller
Flash Memory Family:	FJ	=	Flash program memory, 3.3V
Product Group:	GP2 GP3	= =	General purpose family General purpose family
Pin Count:	02 03	= =	28-pin 44-pin
Temperature Range:	I E H	= = =	-40° C to +85° C (Industrial) -40° C to +125° C (Extended) -40° C to +150° C (High)
Package:	SP SO SS ML PT MM	= = = = = =	Skinny Plastic Dual In-Line - 300 mil body (SPDIP) Plastic Small Outline - Wide - 7.5 mm body (SOIC) Plastic Shrink Small Outline - 5.3 mm body (SSOP) Plastic Quad, No Lead Package - 8x8 mm body (QFN) Plastic Thin Quad Flatpack - 10x10x1 mm body (TQFP) Plastic Quad, No Lead Package - 6x6 mm body (QFN-S)

Examples:

a) dsPIC33FJ32GP202-E/SP:
General-purpose dsPIC33, 32-Kbyte program memory, 28-pin, Extended temp., SPDIP package.