dsPIC30F2010

dsPIC30F2010 Family
Silicon Errata and Data Sheet Clarification

The dsPIC30F2010 family devices that you have received conform functionally to the current Device Data Sheet (DS70118**H**), except for the anomalies described in this document.

The silicon issues discussed in the following pages are for silicon revisions with the Device and Revision IDs listed in Table 1. The silicon issues are summarized in Table 2.

The errata described in this document will be addressed in future revisions of the dsPIC30F2010 silicon.

Note:	This document summarizes all silicon errata issues from all revisions of silicon, previous as well as current. Only the issues indicated in the last column of Table 2 apply to the current silicon revision (**A4**).

Data Sheet clarifications and corrections start on page 23, following the discussion of silicon issues.

The silicon revision level can be identified using the current version of MPLAB® IDE and Microchip's programmers, debuggers and emulation tools, which are available at the Microchip corporate web site (www.microchip.com).

For example, to identify the silicon revision level using MPLAB IDE in conjunction with MPLAB ICD 2 or PICkit™ 3:

1. Using the appropriate interface, connect the device to the MPLAB ICD 2 programmer/debugger or PICkit 3.
2. From the main menu in MPLAB IDE, select *Configure>Select Device*, and then select the target part number in the dialog box.
3. Select the MPLAB hardware tool (*Debugger>Select Tool*).
4. Perform a "Connect" operation to the device (*Debugger>Connect*). Depending on the development tool used, the part number *and* Device Revision ID value appear in the **Output** window.

Note:	If you are unable to extract the silicon revision level, please contact your local Microchip sales office for assistance.

The Device and Revision ID values for the various dsPIC30F2010 silicon revisions are shown in Table 1.

TABLE 1: SILICON DEVREV VALUES

Part Number	Device ID(1)	Revision ID for Silicon Revision(2)				
		A0	A1	A2	A3	A4
dsPIC30F2010	0x0040	0x1000	0x1001	0x1002	0x1003	0x1004

Note 1: The Device and Revision IDs (DEVID and DEVREV) are located at the last two implemented addresses in program memory.

2: Refer to the *"dsPIC30F Flash Programming Specification"* (DS70102) for detailed information on Device and Revision IDs for your specific device.

TABLE 2: SILICON ISSUE SUMMARY

Module	Feature	Item Number	Issue Summary	Affected Revisions(1)				
				A0	A1	A2	A3	A4
CPU	Y Data Space	1.	When an instruction that writes to a location in the address range of Y data memory is immediately followed by a MAC-type DSP instruction that reads a location also resident in Y data memory, the operations will not be performed as specified.	X	X	X	X	X
CPU	MAC Class Instructions with ±4 Address Modification	2.	Sequential MAC instructions, which prefetch data from Y data space using ±4 address modification will cause an address error trap.	X	X	X	X	X
CPU	DAW.b Instruction	3.	The Decimal Adjust instruction, DAW.b, may improperly clear the Carry bit, C (SR<0>).	X	X	X	X	X
PSV Operations	—	4.	In certain instructions, fetching one of the operands from program memory using Program Space Visibility (PSV) will corrupt specific bits in the STATUS Register, SR.	X	X	X	X	X
CPU	Nested DO Loops	5.	When using two DO loops in a nested fashion, terminating the inner-level DO loop by setting the EDT bit (CORCON<11>) will produce unexpected results.	X	X	X	X	X
CPU	REPEAT Loop	6.	When a REPEAT loop is interrupted by two or more interrupts in a nested fashion, an address error trap may be caused.	X	X	X	X	X
CPU	DISI Instruction	7.	The DISI instruction will not disable interrupts if a DISI instruction is executed in the same instruction cycle that the DISI counter decrements to zero.	X	X	X	X	X
Timer	32-bit Mode	8.	The 32-bit general purpose timers do not function as specified for prescaler ratios other than 1:1.	X	X	X	X	X
Output Compare	PWM Mode	9.	The Output Compare module will produce a glitch when loading a 0% duty cycle in PWM mode. It will also miss the next compare after the glitch.	X	X	X	X	X
Output Compare	—	10.	The Output Compare module will produce a glitch on the output when an I/O pin is initially set high and the module is configured to drive the pin low at a specified time.	X	X	X	X	X
ADC	Triggered Conversion	11.	Sampling multiple channels sequentially using any conversion trigger other than the auto-convert feature requires the SAMC bits to be non-zero.	X	X	X	X	X
ADC	Sleep Mode	12.	ADC event triggers from the INT0 pin will not wake-up the device from Sleep mode if the SMPIx bits are non-zero.	X	X	X	X	X
Watchdog Timer	—	13.	The Watchdog Timer does not function as specified.	X	X	X	X	X
PLL	4x Mode	14.	The 4x PLL mode of operation may not function correctly for certain input frequencies.	X	X	X	X	X
Interrupt Controller	—	15.	An interrupt occurring immediately after modifying the CPU IPL, interrupt IPL, interrupt enable or interrupt flag may cause an address error trap.	X	X	X	X	X
PLL	8x Mode	16.	If 8x PLL mode is used, the input frequency range is 5 MHz-10 MHz instead of 4 MHz-10 MHz.	X	X	X	X	X

Note 1: Only those issues indicated in the last column apply to the current silicon revision.

TABLE 2: SILICON ISSUE SUMMARY (CONTINUED)

Module	Feature	Item Number	Issue Summary	Affected Revisions(1)				
				A0	A1	A2	A3	A4
SPI	—	17.	When enabled, the SPI module does not disable RF2 as a general I/O pin.	X	X	X	X	X
QEI	Interrupt Generation	18.	The Quadrature Encoder Interface (QEI) module does not generate an interrupt in a particular overflow condition.	X	X	X	X	X
Sleep Mode	—	19.	Execution of the Sleep instruction (`PWRSAV #0`) may cause incorrect program operation after the device wakes up from Sleep. The current consumption during Sleep may also increase beyond the specifications listed in the device data sheet.	X	X	X	X	X
I^2C™	Slave Mode	20.	The I^2C module loses incoming data bytes when operating as an I^2C slave.	X	X	X	X	X
PWM	Debug Mode	21.	PTMR does not continue counting down after halting code execution in Debug mode.	X	X	X	X	X
I/O	Port Pin Multiplexed with IC1	22.	The Port I/O pin multiplexed with the Input Capture 1 (IC1) function cannot be used as a digital input pin when the UART auto-baud feature is enabled.	X	X	X	X	X
FRC	—	23.	Internal FRC accuracy does not perform to specification.	X	X	X	X	X
I^2C	10-bit Addressing Mode	24.	When the I^2C module is configured for 10-bit addressing using the same address bits (A10 and A9) as other I^2C devices, the A10 and A9 bits may not work as expected.	X	X	X	X	X
Timer	Sleep Mode	25.	Clock switching prevents the device from waking up from Sleep.	X	X	X	X	X
PLL	Lock Status bit	26.	The PLL LOCK Status bit (OSCCON<5>) can occasionally get cleared and generate an oscillator failure trap even when the PLL is still locked and functioning correctly.	X	X	X		
PSV Operations	—	27.	An address error trap occurs in certain addressing modes when accessing the first four bytes of any PSV page.	X	X	X	X	X
I^2C	10-bit Addressing Mode	28.	The 10-bit slave does not set the RBF flag or load the I2CxRCV register on address match if the Least Significant bits (LSbs) of the address are the same as the 7-bit reserved addresses.	X	X	X	X	X
I^2C	10-bit Addressing Mode	29.	When the I^2C module is configured as a 10-bit slave with an address of 0x102, the I2CxRCV register content for the lower address byte is 0x01 rather than 0x02.	X	X	X	X	X
I^2C	Bus Collision	30.	When the I^2C module is enabled, the dsPIC® DSC device generates a glitch on the SDA and SCL pins, causing a false communication start in a single-master configuration or a bus collision in a multi-master configuration.	X	X	X	X	X
Program Flash Memory	RTSP Operation	31.	Run-Time Self-Programming (RTSP) operations need to be timed by the application software. Self-timed write operations are not supported.	X				

Note 1: Only those issues indicated in the last column apply to the current silicon revision.

TABLE 2: SILICON ISSUE SUMMARY (CONTINUED)

Module	Feature	Item Number	Issue Summary	Affected Revisions(1)				
				A0	A1	A2	A3	A4
Data EEPROM	Write/Erase Operation	32.	Write/Erase operations performed on Data EEPROM need to be timed by the application software. Self-timed write operations are not supported.	X				
Program Flash Memory	RTSP Operation	33.	When a device Reset occurs while an RTSP operation is in progress, code execution may lead to an Address Error trap.	X				
Data EEPROM	—	34.	Data EEPROM is operational at a device throughput of up to 25 MIPS.	X				
Interrupt Controller	—	35.	A specific write sequence for the Interrupt Priority Control 2 (IPC2) Special Function Register (SFR) is required.	X				
Sleep Mode	IPD Sleep Current	36.	The device exhibits IPD less than 0.1 µA. However, certain work arounds are required to achieve IPD in this range.	X				
QEI	Timer Gated Accumulation Mode	37.	When Timer Gated Accumulation is enabled, the QEI module does not generate an interrupt on every falling edge.	X	X	X	X	X
QEI	Timer Gated Accumulation Mode	38.	When Timer Gated Accumulation is enabled, and an external signal is applied, the POSCNT increments and generates an interrupt after a match with MAXCNT.	X	X	X	X	X
ADC	Current Consumption in Sleep Mode	39.	If the ADC module is in an enabled state when the device enters Sleep Mode, the power-down current (IPD) of the device may exceed the device data sheet specifications.	X	X	X	X	X

Note 1: Only those issues indicated in the last column apply to the current silicon revision.

Silicon Errata Issues

Note: This document summarizes all silicon errata issues from all revisions of silicon, previous as well as current. Only the issues indicated by the shaded column in the following tables apply to the current silicon revision (**A4**).

1. Module: CPU

When an instruction that writes to a location in the address range of Y data memory (addresses between 0x0900 and 0x09FF) is immediately followed by a MAC-type DSP instruction that reads a location also resident in Y data memory, the two operations will not be executed as specified. This is demonstrated in Example 1.

EXAMPLE 1: INCORRECT RESULTS

```
MOV     #0x090A, W0     ;Load address > =
                        ;0x900 into W0
MOV     #0x09B0, W10    ;Load address >=
                        ;0x900 into W10
MOV     W2, [W0++]      ;Perform indirect
                        ;write via W0 to
                        ;address >= 0x900
MAC     W4*W5, A, [W10]+=2, W5 ;Perform
                        ;read operation
                        ;using Y-AGU
```

Work arounds

Work around 1:

Insert a NOP between the two instructions as shown in Example 2.

EXAMPLE 2: CORRECT RESULTS

```
MOV     #0x090A, W0     ;Load address > =
                        ;0x900 into W0
MOV     #0x09B0, W10    ;Load address >=
                        ;0x900 into W10
MOV     W2, [W0++]      ;Perform indirect
                        ;write via W0 to
                        ;address >= 0x900
NOP                     ;No operation
MAC     W4*W5, A, [W10]+=2, W5 ;Perform
                        ;read operation
                        ;using Y-AGU
```

Work around 2:

If Work around 1 is not feasible due to real-time application constraints, the user may take precautions to ensure that a write operation performed on a location in Y data memory is not immediately followed by a DSP MAC-type instruction that performs a read operation of a location in Y data memory.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

2. Module: CPU

Sequential MAC class instructions, which prefetch data from Y data space using ±4 address modification, will cause an address error trap. The trap occurs only when all of the following conditions are true:

1. Two sequential MAC class instructions (or a MAC class instruction executed in a REPEAT or DO loop) that prefetch from Y data space.
2. Both instructions prefetch data from Y data space using the + = 4 or – = 4 address modification.
3. Neither of the instruction uses an accumulator write back.

Work around

This problem can be avoided using any of the following methods:

1. Inserting any other instruction between the two MAC class instructions.
2. Adding an accumulator write back (a dummy write back if needed) to either of the MAC class instructions.
3. Do not use the + = 4 or – = 4 address modification.
4. Do not prefetch data from Y data space.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

3. Module: CPU

The Decimal Adjust instruction, DAW.b, may improperly clear the Carry bit, C (SR<0>), when executed.

Work around

Check the state of the Carry bit prior to executing the DAW.b instruction. If the Carry bit is already set, set the Carry bit again after executing the DAW.b instruction. Example 3 shows how the application should process the Carry bit during a BCD addition operation.

EXAMPLE 3: CHECK CARRY BIT BEFORE DAW.b

```
    .include "p30f6010.inc"
    .......
    mov.b   #0x80, w0   ;First BCD number
    mov.b   #0x80, w1   ;Second BCD number
    add.b   w0, w1, w2  ;Perform addition
    bra     NC, L0      ;If C set go to L0
    daw.b   w2          ;If not,do DAW and
    bset.b  SR, #C      ;set the carry bit
    bra     L1          ;and exit
L0:daw.b    w2
L1: ....
```

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

4. Module: PSV Operations

When one of the operands of instructions shown in Table 3 is fetched from program memory using Program Space Visibility (PSV), the STATUS register, SR and/or the results may be corrupted.

These instructions are identified in Table 3. Example 4 demonstrates a scenario where this occurs.

Also, always use Work around 2 if the C compiler is used to generate code for dsPIC30F2010 devices.

TABLE 3: AFFECTED INSTRUCTIONS

Instruction(1)	Examples of Incorrect Operation(2)	Data Corruption IN
ADDC	ADDC W0, [W1++], W2 ;	SR<1:0> bits(3), Result in W2
SUBB	SUBB.b W0, [++W1], W3 ;	SR<1:0> bits(3), Result in W3
SUBBR	SUBBR.b W0, [++W1], W3 ;	SR<1:0> bits(3), Result in W3
CPB	CPB W0, [W1++], W4 ;	SR<1:0> bits(3)
RLC	RLC [W1], W4 ;	SR<1:0> bits(3), Result in W4
RRC	RRC [W1], W2 ;	SR<1:0> bits(3), Result in W2
ADD (Accumulator-based)	ADD [W1++], A ;	SR<1:0> bits(3)
LAC	LAC [W1], A ;	SR<15:10> bits(4)

Note 1: Refer to the "*dsPIC30F/33F Programmer's Reference Manual*" (DS70157) for details on the dsPIC30F Instruction set.

2: The errata only affects these instructions when a PSV access is performed to fetch one of the source operands in the instruction. A PSV access is performed when the Effective Address of the source operand is greater than 0x8000 and the PSV bit (CORCON<2>) is set to '1'. In the examples shown, the data access from program memory is made via the W1 register.

3: SR<1:0> bits represent the Sticky Zero and Carry Status bits, respectively.

4: SR<15:10> bits represent the Accumulator Overflow and Saturation Status bits.

EXAMPLE 4: INCORRECT RESULTS

```
.include "p30fxxxx.inc"
.......
MOV.B   #0x00, W0   ;Load PSVPAG register
MOV.B   WREG, PSVPAG
BSET    CORCON, #PSV;Enable PSV
....
MOV     #0x8200, W1;Set up W1 for
                   ;indirect PSV access
                   ;from 0x000200
ADD     W3, [W1++], W5 ;This instruction
                       ;works ok
ADDC    W4, [W1++], W6 ;Carry flag and
                       ;W6 gets
                       ;corrupted here!
```

EXAMPLE 5: CORRECT RESULTS

```
.include "p30fxxxx.inc"
.......
MOV.B   #0x00, w0   ;Load PSVPAG register
MOV.B   WREG, PSVPAG
BSET    CORCON, #PSV;Enable PSV
....
MOV     #0x8200, W1;Set up W1 for
                   ;indirect PSV access
                   ;from 0x000200
ADD     W3, [W1++], W5 ;This instruction
                       ;works ok
MOV     [W1++], W2  ;Load W2 with data
                    ;from program memory
ADDC    W4, W2, W6 ;Carry flag and W4
                   ;results are okay!
```

Work arounds

Work around 1: For Assembly Language Source Code

To work around the erratum in the MPLAB ASM30 assembler, the application may perform a PSV access to move the source operand from program memory to RAM or a W register prior to performing the operations listed in Table 3. The work around for Example 4 is demonstrated in Example 5.

Work around 2: For C Language Source Code

For applications using C language, MPLAB C Compiler for dsPIC® DSCs (formerly known as the MPLAB C30 C Compiler), versions 1.20.04 or higher, provides the following command-line switch that implements a work around for the erratum.

```
-merrata=psv
```

Refer to the "readme.txt" file in the MPLAB C Compiler for dsPIC DSCs for further details.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

5. Module: CPU

When using two DO loops in a nested fashion, terminating the inner-level DO loop by setting the EDT bit (CORCON<11>) will produce unexpected results. Specifically, the device may continue executing code within the outer DO loop forever. This erratum does not affect the operation of the MPLAB C Compiler for dsPIC DSCs.

Work around

The application should save the DCOUNT Special Function Register (SFR) prior to entering the inner DO loop and restore it upon exiting the inner DO loop. This work around is shown in Example 6.

EXAMPLE 6: SAVE AND RESTORE DCOUNT

```
        .include "p30fxxxx.inc"
        .......
        DO #CNT1, LOOP0    ;Outer loop start
        ....
        PUSH    DCOUNT     ;Save DCOUNT
        DO      #CNT2, LOOP1   ;Inner loop
        ....               ;starts
        BTSS    Flag, #0
        BSET    CORCON, #EDT;Terminate inner
        ....               ;DO-loop early
        ....
LOOP1:  MOV     W1, W5     ;Inner loop ends
        POP     DCOUNT     ;Restore DCOUNT
        ...
LOOP0:  MOV     W5, W8     ;Outer loop ends
```

Note: For details on the functionality of the EDT bit, see **2.9.2.4 "Early Termination of the DO Loop"** in **Section 2. "CPU"** (DS70049) of the *"dsPIC30F Family Reference Manual"*.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

6. Module: CPU

When interrupt nesting is enabled (or the NSTDIS bit (INTCON1<15>) is '0'), the following sequence of events will lead to an address error trap:

1. REPEAT loop is active.
2. An interrupt is generated during the execution of the REPEAT loop.
3. The CPU executes the Interrupt Service Routine (ISR) of the source causing the interrupt.
4. Within the ISR, when the CPU is executing the first instruction cycle of the 3-cycle RETFIE (Return from Interrupt) instruction, a second interrupt is generated by a source with a higher interrupt priority.

Work around

Processing of Interrupt Service Routines should be disabled while the RETFIE instruction is being executed. This may be accomplished in two different ways:

1. Place a DISI instruction immediately before the RETFIE instruction in all Interrupt Service Routines of interrupt sources that may be interrupted by other higher priority interrupt sources (with priority levels 1 through 6). This is shown in Example 7 in the Timer1 ISR. In this example, a DISI instruction inhibits level 1 through level 6 interrupts for 2 instruction cycles, while the RETFIE instruction is executed.

EXAMPLE 7: DISI BEFORE RETFIE

```
__T1Interrupt:     ;Timer1 ISR
    PUSH    W0     ;This line optional
    .......
    BCLR    IFS0, #T1IF
    POP     W0     ;This line optional
    DISI    #1
    RETFIE     ;Another interrupt occurs
               ;here and it is processed
               ;correctly
```

2. Immediately prior to executing the RETFIE instruction, increase the CPU priority level by modifying the IPL<2:0> bits (SR<7:5>) to '111' as shown in Example 8. This will disable all interrupts between priority levels 1 through 7.

EXAMPLE 8: RAISE IPL BEFORE RETFIE

```
__T1Interrupt:     ;Timer1 ISR
    PUSH    W0
    .......
    BCLR    IFS0, #T1IF
    MOV.B   #0xE0, W0
    MOV.B   WREG, SR
    POP     W0
    RETFIE     ;Another interrupt occurs
               ;here and it is processed
               ;correctly
```

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

7. Module: CPU

The DISI instruction will not disable interrupts when a DISI instruction is executed in the same instruction cycle that the DISI counter decrements to zero. For example, when user code executes a DISI #7, interrupts for 7 + 1 cycles (7 + the DISI instruction itself) are disabled. In that case, the DISI instruction uses a counter that counts down from 7 to 0. The counter is loaded with 7 at the end of the DISI instruction.

If the user code executes another DISI on the instruction cycle where the DISI counter has become zero, the new DISI count is loaded, but the DISI state machine does not properly re-engage and continues to disable interrupts. At this point, all interrupts are enabled. The next time the user code executes a DISI instruction, the feature will act normally and block interrupts.

To summarize, it is only when a DISI execution is coincident with the current DISI count = 0, that the issue occurs. Executing a DISI instruction before the DISI counter reaches zero will not produce this error. In this case, the DISI counter is loaded with the new value, and interrupts remain disabled until the counter becomes zero.

Work around

When executing multiple DISI instructions within the source code, make sure that subsequent DISI instructions have at least one instruction cycle between the time that the DISI counter decrements to zero and the next DISI instruction. Alternatively, ensure that the subsequent DISI instructions are called before the DISI counter decrements to zero.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

8. Module: Timer

Pairs of 16-bit timers may be combined to form 32-bit timers. For example, Timer2 and Timer3 are combined into a single 32-bit timer. For this release of silicon, when a 32-bit timer is prescaled by ratios other than 1:1, unexpected results may occur.

Work around

None. The application may only use the 1:1 prescaler for 32-bit timers.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

9. Module: Output Compare

If the desired duty cycle is '0' (OCxRS = 0), the module will generate a high level glitch of 1 TCY. The second problem is that on the next cycle after the glitch, the OC pin does not go high, or in other words, it misses the next compare for any value written on OCxRS.

Work around

There are two possible solutions to this problem:

1. Load a value greater than '0' to the OCxRS register when operating in PWM mode. In this case, no 0% duty cycle is achievable.
2. If the application requires 0% duty cycles, the Output Compare module can be disabled for 0% duty cycles, and re-enabled for non-zero percent duty cycles.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

10. Module: Output Compare

A glitch will be produced on an output compare pin under the following conditions:

- The user software initially drives the I/O pin high using the Output Compare module or a write to the associated PORT register.
- The Output Compare module is configured and enabled to drive the pin low at some later time (OCxCON = Ox0002 or OCxCON = 0x0003).

When these events occur, the Output Compare module will drive the pin low for one instruction cycle (TCY) after the module is enabled.

Work around

None. However, the user may use a timer interrupt, and write to the associated PORT register to control the pin manually.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

11. Module: ADC

Sampling multiple channels sequentially using any conversion trigger source other than the auto-convert feature requires the SAMC bits to be non-zero. Therefore, if the following conditions are all satisfied, the module may not operate as specified:

- Multiple S&H channels are sampled sequentially

  CHPS (ADCON2<9:8>) is not equal to '00' and SIMSAM (ADCON1<3>) = 0

- Auto-convert option is not chosen as the conversion trigger

  SSRC (ADCON1<7:5>) is not equal to '111'

- SAMC (ADCON3<12:8>) is equal to '00000'

Work around

Set the value of the SAMC bits to anything other than '00000'. The ADC module will now operate as specified.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

12. Module: ADC

ADC event triggers from the INT0 pin will not wake-up the device from Sleep mode if the SMPIx bits are non-zero. This implies that if the ADC is configured to generate an interrupt after a certain number of INT0 triggered conversions, the ADC conversions will not be triggered and the device will remain in Sleep. The ADC will perform conversions and wake-up the device only if it is configured to generate an interrupt after each INT0 triggered conversion (SMPI<3:0> = 0000).

Work around

None. If an ADC event trigger from the INT0 pin is required, initialize SMPI<3:0> to '0000' (interrupt on every conversion).

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

13. Module: Watchdog Timer

The Watchdog Timer does not function as specified. If the CLRWDT instruction is not executed before the Watchdog Timer is half-expired or greater, the device will reset.

Work around

The user must always issue the CLRWDT instruction before the Watchdog Timer is half-expired. For instance, if the Watchdog time-out period is configured for 2 ms, the CLRWDT instruction must be executed faster than every 1 ms.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

14. Module: PLL

When the 4x PLL mode of operation is selected, the specified input frequency range of 4 MHz-10 MHz is not fully supported.

When device VDD is 2.5V-3.0V, the 4x PLL input frequency must be in the range of 4 MHz-5 MHz. When device VDD is 3.0V-3.6V, the 4x PLL input frequency must be in the range of 4 MHz-6 MHz for both industrial and extended temperature ranges.

Work around

1. Use 8x PLL or 16x PLL mode of operation and set final device clock speed using the POST<1:0> Oscillator Postscaler Control bits (OSCCON<7:6>).
2. Use the EC without PLL Clock mode with a suitable clock frequency to obtain the equivalent 4x PLL clock rate.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

15. Module: Interrupt Controller

The following sequence of events will lead to an address error trap. The generic term “Interrupt 1” is used to represent any enabled dsPIC30F interrupt.

1. User software performs one of the following operations:
   - CPU IPL is raised to Interrupt 1 IPL level or higher, or
   - Interrupt 1 IPL is lowered to CPU IPL level or lower, or
   - Interrupt 1 is disabled (Interrupt 1 IE bit set to ‘0’), or
   - Interrupt 1 flag is cleared.
2. Interrupt 1 occurs between 2 and 4 instruction cycles after any of the operations listed above.

Work arounds

Work around 1: For Assembly Language Source Code

The user may disable interrupt nesting, disable interrupts before modifying the Interrupt 1 setting, or execute a `DISI` instruction before modifying the CPU IPL or Interrupt 1. A minimum DISI value of 4 is required if the `DISI` instruction is executed immediately before the CPU IPL or Interrupt 1 is modified, as shown in Example 9. It is necessary to have DISI active for four cycles after the CPU IPL or Interrupt 1 is modified.

EXAMPLE 9: USING `DISI`

```
.include "p30fxxxx.inc"
...
DISI #4 ; protect the disable
; of INT1
BCLR IEC1, #INT1IE ; disable interrupt 1
... ; next instruction
; protected by DISI
```

Work around 2: For C Language Source Code

For applications using the C language, MPLAB C Compiler for dsPIC DSCs versions 1.32 and higher, provide several macros for modifying the CPU IPL. The `SET_CPU_IPL` macro provides the ability to safely modify the CPU IPL, as shown in Example 10.

EXAMPLE 10: USING `SET_CPU_IPL` MACRO

```
// Note: Macro defined in device include
// files
#define SET_CPU_IPL (ipl){ \
int DISI_save; \
\
DISI_save = DISICNT; \
asm volatile ("disi #0x3FFF");\
SRbits.IPL = ipl; \
__builtin_nop();   \
__builtin_nop();   \
DISICNT = DISI_save; } (void) 0;

#include "p30fxxxx.h"
. . .
SET_CPU_IPL (3)
. . .
```

There is one level of DISI, so this macro saves and restores the DISI state. For temporarily modifying and restoring the CPU IPL, the macros `SET_AND_SAVE_CPU_IPL` and `RESTORE_CPU_IPL` can be used, as shown in Example 11. These macros also make use of the `SET_CPU_IPL` macro.

EXAMPLE 11: USING `SET_AND_SAVE_CPU_IPL` AND `RESTORE_CPU_IPL` MACROS

```
// Note: Macros defined in device include files
#define SET_AND_SAVE_CPU_IPL (save_to, ipl){ \
save_to = SRbits.IPL; \
SET_CPU_IPL (ipl); } (void) 0;

#define RESTORE_CPU_IPL (saved_to) SET_CPU_IPL (saved_to)

#include "p30fxxxx.h"
. . .
int save_to;
SET_AND_SAVE_CPU_IPL (save_to, 3)
. . .
RESTORE_CPU_IPL (save_to)
```

For modification of the Interrupt 1 setting, the `INTERRUPT_PROTECT` macro can be used. This macro disables interrupts before executing the desired expression, as shown in Example 12. This macro is not distributed with the compiler.

EXAMPLE 12: USING `INTERRUPT_PROTECT` MACRO

```
#define INTERRUPT_PROTECT (x) { \
int save_sr; \
SET_AND_SAVE_CPU_IPL (save_sr, 7);\
x; \
RESTORE_CPU_IPL (save_sr); } (void) 0;

. . .
INTERRUPT_PROTECT (IEC0bits.U1TXIE=0);
```

Note: If you are using a MPLAB C Compiler for dsPIC DSCs version earlier than version 1.32, you may still use the macros by adding them to your application.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

16. Module: PLL

If 8x PLL mode is used, the input frequency range is 5 MHz-10 MHz instead of 4 MHz-10 MHz.

Work around

None. If 8x PLL is used, ensure that the input crystal or clock frequency is 5 MHz or greater.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

17. Module: SPI

The SPI module does not have full control of the RF2 pin when the SPIEN bit is set. This means that RF2 can be used as general I/O when the SPI module is enabled.

Work around

It is recommended to avoid using the RF2 pin as I/O if the SPIEN bit is set.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

18. Module: QEI

The Quadrature Encoder Interface (QEI) module does not generate an interrupt when MAXCNT is set to 0xFFFF and the following events occur:

1. POSCNT underflows from 0x0000 to 0xFFFF.
2. POSCNT stops.
3. POSCNT overflows from 0xFFFF to 0x0000.

This sequence of events occurs when the motor is running in one direction, which causes POSCNT to underflow to 0xFFFF. Once this happens, the motor stops and starts to run in the opposite direction, which generates an overflow from 0xFFFF to 0x0000. The QEI module does not generate an interrupt when this condition occurs.

Work around

To prevent this condition from occurring, set MAXCNT to 0x7FFF, which will cause an interrupt to be generated by the QEI module.

In addition, a global variable can be used to keep track of bit 15, so that when an overflow or underflow condition is present on POSCNT, the variable will toggle bit 15. Example 13 shows the code required for this global variable.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

EXAMPLE 13:

```
unsigned int POSCNT_b15 = 0;
unsigned int Motor_Position = 0;

int main(void)
{
            // ... User's code

            MAXCNT = 0x7FFF;       // Instead of 0xFFFF

            Motor_Position = POSCNT_b15 + POSCNT;

            // ... User's code
}
void __attribute__((__interrupt__)) _QEIInterrupt(void)
{
            IFSxbits.QEIIF = 0;    // Clear QEI interrupt flag
                                   // x=2 for dsPIC30F
                                   // x=3 for dsPIC33F
            POSCNT_b15 ^= 0x8000;  // Overflow or Underflow
}
```

19. Module: Sleep Mode

Execution of the Sleep instruction (`PWRSAV #0`) may cause incorrect program operation after the device wakes up from Sleep. The current consumption during Sleep may also increase beyond the specifications listed in the device data sheet.

Work arounds

To avoid this issue, implement any of the following three work arounds, depending on the application requirements.

Work around 1:

Ensure that the `PWRSAV #0` instruction is located at the end of the last row of program Flash memory available on the target device, and fill the remainder of the row with `NOP` instructions.

This can be accomplished by replacing all occurrences of the `PWRSAV #0` instruction with a function call to a suitably aligned subroutine. The `address( )` attribute provided by the MPLAB ASM30 assembler can be used to correctly align the instructions in the subroutine. For an application written in C, the function call would be `GotoSleep( )`. While for an assembly language application, the function call would be `CALL _GotoSleep`.

The address error Trap Service Routine (TSR) software can then replace the invalid return address saved on the stack with the address of the instruction immediately following the `_GotoSleep` or `GotoSleep( )` function call. This ensures that the device continues executing the correct code sequence after waking up from Sleep mode.

Example 14 demonstrates the work around described above.

EXAMPLE 14:

```
; ------------------------------------------------------------------------------------------------
.global __reset
.global _main
.global _GotoSleep
.global __AddressError
.global __INT1Interrupt
; ------------------------------------------------------------------------------------------------
    .section *, code
_main:
    BSET    INTCON2, #INT1EP  ; Set up INT pins to detect falling edge
    BCLR    IFS1, #INT1IF     ; Clear interrupt pin interrupt flag bits
    BSET    IEC1, #INT1IE     ; Enable ISR processing for INT pins
    CALL    _GotoSleep        ; Call function to enter SLEEP mode
_continue:
    BRA  _continue
; ------------------------------------------------------------------------------------------------
; Address Error Trap
__AddressError:
    BCLR    INTCON1, #ADDRERR
    ; Set program memory return address to _continue
    POP.D   W0
    MOV.B   #tblpage (_continue), W1
    MOV     #tbloffset (_continue), W0
    PUSH.D  W0
    RETFIE
; ------------------------------------------------------------------------------------------------
__INT1Interrupt:
    BCLR    IFS1, #INT1IF         ; Ensure flag is reset
    RETFIE                        ; Return from Interrupt Service Routine
; ------------------------------------------------------------------------------------------------
    .section *, code, address (0x1FC0)
_GotoSleep:
; fill remainder of the last row with NOP instructions
    .rept 31
        NOP
    .endr
; Place SLEEP instruction in the last word of program memory
    PWRSAV #0
```

Work around 2:

Instead of executing a `PWRSAV #0` instruction to put the device into Sleep mode, perform a clock switch to the 512 kHz Low-Power RC (LPRC) Oscillator with a 64:1 postscaler mode. This enables the device to operate at 0.002 MIPS, thereby significantly reducing the current consumption of the device. Similarly, instead of using an interrupt to wake-up the device from Sleep mode, perform another clock switch back to the original oscillator source to resume normal operation. Depending on the device, refer to **Section 7. "Oscillator"** (DS70054) or **Section 29. "Oscillator"** (DS70268) in the "*dsPIC30F Family Reference Manual*" (DS70046) for more details on performing a clock switch operation.

Note:	The above work around is recommended for users for whom application hardware changes are not possible.

Work around 3:

Instead of executing a `PWRSAV #0` instruction to put the device into Sleep mode, perform a clock switch to the 32 kHz LPRC with a 64:1 postscaler mode. This enables the device to operate at 0.000125 MIPS, thereby significantly reducing the current consumption of the device. Similarly, instead of using an interrupt to wake-up the device from Sleep mode, perform another clock switch back to the original oscillator source to resume normal operation. Depending on the device, refer to **Section 7. "Oscillator"** (DS70054) or **Section 29. "Oscillator"** (DS70268) in the "*dsPIC30F Family Reference Manual*" (DS70046) for more details on performing a clock switch operation.

Note:	The above work around is recommended for users for whom application hardware changes are possible, and also for users whose application hardware already includes a 32 kHz LP Oscillator crystal.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

20. Module: I²C

When the I²C module is configured as a slave, either in single-master or multi-master mode, the I²C receiver buffer is filled whether a valid slave address is detected or not. Therefore, an I²C receiver overflow condition occurs and this condition is indicated by the I2COV flag in the I2CSTAT register.

This overflow condition inhibits the ability to set the I²C receive interrupt flag (SI2CF) when the last valid data byte is received. Therefore, the I²C slave Interrupt Service Routine is not called and the I²C receiver buffer is not read prior receiving the next data byte.

Work arounds

To avoid this issue, either of the following two work arounds can be implemented, depending on the application requirements.

Work around 1:

For applications in which the I²C receiver interrupt is not required, the following procedure can be used to receive valid data bytes:

1. Wait until the RBF flag is set.
2. Poll the I²C receiver interrupt SI2CIF flag.
3. If SI2CF is not set in the corresponding Interrupt Flag Status register (IFSx), a valid address or data byte has not been received for the current slave. Execute a dummy read of the I²C receiver buffer, I2CRCV; this will clear the RBF flag. Go back to step 1 until SI2CF is set and then continue to Step 4.
4. If the SI2CF is set in the corresponding Interrupt Flag Status register (IFSx), valid data has been received. Check the D_A flag to verify that an address or a data byte has been received.
5. Read the I2CRCV buffer to recover valid data bytes. This will also clear the RBF flag.
6. Clear the I²C receiver interrupt flag SI2CF.
7. Go back to step 1 to continue receiving incoming data bytes.

Work around 2:

Use this work around for applications in which the I²C receiver interrupt is required. Assuming that the RBF and the I2COV flags in the I2CSTAT register are set due to previous data transfers in the I²C bus (i.e., between master and other slaves); the following procedure can be used to receive valid data bytes:

1. When a valid slave address byte is detected, SI2CF bit is set and the I²C slave interrupt service routine is called; however, the RBF and I2COV bits are already set due to data transfers between other I²C nodes.
2. Check the status of the D_A flag and the I2COV flag in the I2CSTAT register when executing the I²C slave service routine.
3. If the D_A flag is cleared and the I2COV flag is set, an invalid data byte was received but a valid address byte was received. The overflow condition occurred because the I²C receive buffer was overflowing with previous I²C data transfers between other I²C nodes. This condition only occurs after a valid slave address was detected.
4. Clear the I2COV flag and perform a dummy read of the I²C receiver buffer, I2CRCV, to clear the RBF bit and recover the valid address byte. This action will also avoid the loss of the next data byte due to an overflow condition.
5. Verify that the recovered address byte matches the current slave address byte. If they match, the next data to be received is a valid data byte.
6. If the D_A flag and the I2COV flag are both set, a valid data byte was received and a previous valid data byte was lost. It will be necessary to code for handling this overflow condition.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

21. Module: PWM

If the PTDIR bit is set (when PTMR is counting down), and the CPU execution is halted (after a breakpoint is reached), PTMR will start counting up as if PTDIR was zero.

Work around

None.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

22. Module: I/O

If the user application enables the auto-baud feature in the UART module, the I/O pin multiplexed with the IC1 (Input Capture 1) pin cannot be used as a digital input. However, the external interrupt function (INT1) can be used.

Work around

None.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

23. Module: FRC

Internal FRC accuracy is outside the specification documented in **"Electrical Characteristics"**, Table 22-17 "AC Characteristics: Internal RC Accuracy" of the "*dsPIC30F2010 Data Sheet*" (DS70118).

The actual internal FRC accuracy is:

- ±4% for 25°C
- ±5% for -40°C and 85°C
- ±6% for 125°C

Work around

None.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

24. Module: I^2C

If there are two I^2C devices on the bus, one of them acts as the master receiver and the other acts as the slave transmitter. If both devices are configured for 10-bit addressing mode, and have the same value in the A10 and A9 bits of their addresses, when the slave select address is sent from the master, both the master and slave acknowledge it. When the master sends out the read operation, both the master and the slave enter into Read mode, and both of them transmit the data. The resultant data will be the ANDing of the two transmissions.

Work around

In all I^2C devices, the addresses, as well as bits A10 and A9, should be different.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

25. Module: Timer

When the timer is being operated in Asynchronous mode using the Secondary Oscillator (32.768 kHz) and the device is put into Sleep mode, a clock switch to any other oscillator mode before putting the device to Sleep prevents the timer from waking the device from Sleep.

Work around

Do not clock switch to any other oscillator mode if the timer is being used in Asynchronous mode using the Secondary Oscillator (32.768 kHz).

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

26. Module: PLL

The PLL LOCK Status bit (OSCCON<5>) can occasionally get cleared and generate an oscillator failure trap even when the PLL is still locked and functioning correctly.

Work around

The user application must include an oscillator failure Trap Service Routine. In the Trap Service Routine, first inspect the status of the Clock Failure Status bit (OSCCON<3>). If this bit is clear, return from the Trap Service Routine immediately and continue program execution.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X					

27. Module: PSV Operations

An address error trap occurs in certain addressing modes when accessing the first four bytes of a PSV page. This only occurs when using the following addressing modes:

- MOV.D
- Register Indirect Addressing (word or byte mode) with pre/post-decrement

Work around

Do not perform PSV accesses to any of the first four bytes using the above addressing modes. For applications using the C language, MPLAB C Compiler for dsPIC DSCs, version 3.11 or higher, provides the following command-line switch that implements a work around for the erratum.

```
-merrata=psv_trap
```

Refer to the `readme.txt` file in the MPLAB C Compiler for dsPIC DSCs for further details.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

28. Module: I^2C

In 10-bit Addressing mode, some address matches do not set the RBF flag or load the receive register I2CxRCV, if the lower address byte matches the reserved addresses. In particular, these include all addresses with the form XX0000XXXX and XX1111XXXX, with the following exceptions:

- 001111000X
- 011111001X
- 101111010X
- 111111011X

Work around

Ensure that the lower address byte in 10-bit Addressing mode does not match any 7-bit reserved addresses.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

29. Module: I^2C

If the I^2C module is configured as a 10-bit slave with an address of 0x102, the I2CxRCV register content for the lower address byte is 0x01, rather than 0x02. However, the I^2C module acknowledges both address bytes.

Work around

None.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

30. Module: I²C

When the I²C module is enabled by setting the I2CEN bit in the I2CCON register, the dsPIC DSC device generates a glitch on the SDA and SCL pins. This glitch falsely indicates "Communication Start" to all devices on the I²C bus, and can cause a bus collision in a multi-master configuration.

Additionally, when the I2CEN bit is set, the S and P bits of the I²C module are set to values '1' and '0', respectively, which indicate a "Communication Start" condition.

Work arounds

To avoid this issue, either of the following two work arounds can be implemented, depending on the application requirements.

Work around 1:

In a single-master environment, add a delay between enabling the I²C module and the first data transmission. The delay should be equal to or greater than the time it takes to transmit two data bits.

In the multi-master configuration, in addition to the delay, all other I²C masters should be synchronized and wait for the I²C module to be initialized before initiating any kind of communication.

Work around 2:

In dsPIC DSC devices in which the I²C module is multiplexed with other modules that have precedence in the use of the pin, it is possible to avoid this glitch by enabling the higher priority module before enabling the I²C module.

Use the following procedure to implement this work around:

1. Enable the higher priority peripheral module that is multiplexed on the same pins as the I²C module.
2. Set up and enable the I²C module.
3. Disable the higher priority peripheral module that was enabled in step 1.

Note: Work around 2 works only for devices that share the SDA and SCL pins with another peripheral that has a higher precedence over the port latch, such as the UART. The priority is shown in the pin diagram located in the data sheet. For example, if the SDA and SCL pins are shared with the UART and SPI pins, and the UART has higher precedence on the port latch pin.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

31. Module: Program Flash Memory

When performing Run-Time Self-Programming (RTSP) operations on program Flash memory or write operations on Data EEPROM, the device automatically times the erase/write operation. For this revision of silicon, this method of timing the erase/write operation is not supported.

Note that this erratum does not affect programming Flash memory using a device programmer, such as MPLAB ICD 2 or PRO MATE®.

Work around

When updating program Flash memory, the programming cycle time must be controlled using an on-chip timer resource. Setting the TWRI bit (NVMCON<8>) to a logic '1' enables the program Flash programming cycle time to be terminated by the next acknowledged interrupt source. Therefore, the user must ensure that a single timer is configured to generate a CPU recognized interrupt and terminate the programming cycle.

The timer cycle should be set for a value greater than 2 ms but less than 5 ms.

Example 15 demonstrates this work around for a programming operation. A similar work around may be applied for an erase operation.

EXAMPLE 15:

```
;The following code example assumes that the
;Write-latches have been pre-loaded and
;Timer1 has been set up to interrupt at the
;end of the programming cycle.
    CLR     MyFlag          ;Clear a flag
    CLR     TMR1            ;Clear Timer1
    BSET    T1CON, #TON     ;Turn Timer1 On
    DISI    #8
    MOV     #0X4101, W0     ;Load NVMCON with
    MOV     W0, NVMCON      ;bit8 set
    MOV     #0X55, W0       ;Perform Unlock
    MOV     W0, NVMKEY      ;sequence
    MOV     #0XAA, W0
    MOV     W0, NVMKEY
    BSET    NVMCON, #WR     ;Set the WR bit
    NOP                     ;CPU stalls until
    NOP                     ;next interrupt
L1: BTSS    MyFlag, #0      ;Optionally wait
    BRA     L1              ;for flag set
                            ;by Timer1 ISR
    BCLR    T1CON, #TON     ;Turn off Timer1
    ......                  ;Continue

__T1Interrupt:              ;Timer1 ISR
    SETM    MyFlag          ;Set a flag
    BCLR    IFS0, #T1IF     ;Clear T1IF and
    RETFIE                  ;return from ISR
```

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X							

32. Module: Data EEPROM

When performing write/erase operations on Data EEPROM, the device automatically times the write/erase operation. For this revision of silicon, this method of timing the erase/write operation is not supported.

Note that this erratum does not affect writing to Data EEPROM using a device programmer, such as MPLAB ICD 2 or PRO MATE.

Work around

When updating Data EEPROM, the write cycle time must be controlled using an on-chip timer resource. Setting the TWRI bit (NVMCON<8>) to a logic '1' enables the Data EEPROM write cycle time to be terminated by the next acknowledged interrupt source. Therefore, the user must ensure that a single timer is configured to generate a CPU recognized interrupt and terminate the write cycle.

The timer cycle should be set for a value greater than 2 ms but less than 5 ms.

Example 16 demonstrates this work around. A similar work around may be applied for an erase operation.

EXAMPLE 16:

```
;The following code example assumes that the
;Write-latches have been pre-loaded and
;Timer1 has been set up to interrupt at the
;end of the write/erase cycle.
    CLR     MyFlag          ;Clear a flag
    CLR     TMR1            ;Clear Timer1
    BSET    T1CON, #TON     ;Turn Timer1 On
    DISI    #8
    MOV     #0X4105, W0     ;Load NVMCON with
    MOV     W0, NVMCON      ;bit8 set
    MOV     #0X55, W0       ;Perform Unlock
    MOV     W0, NVMKEY      ;sequence
    MOV     #0XAA, W0
    MOV     W0, NVMKEY
    BSET    NVMCON, #WR     ;Set the WR bit
    NOP
    NOP
L1: BTSS    MyFlag, #0      ;Optionally, wait
    BRA     L1              ;for flag set
                            ;by Timer1 ISR
    BCLR    T1CON, #TON     ;Turn off Timer1
    ......                  ;Continue

__T1Interrupt:              ;Timer1 ISR
    SETM    MyFlag          ;Set a flag
    BCLR    IFS0, #T1IF     ;Clear T1IF and
    RETFIE                  ;return from ISR
```

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X							

33. Module: Program Flash Memory

If a device Reset occurs while an RTSP operation is in progress, code execution after the reset may lead to an Address Error Trap.

Work around

The user should define an address Error Trap Service Routine as shown in Example 17 in order to resume normal code execution.

EXAMPLE 17:

```
__AddressError:
    bclr    RCON, #TRAPR    ;Clear the Trap
                            ;Reset Flag Bit
    bclr    INTCON1, #ADDRERR  ;Clear the
                            ;Address Error
                            ;trap flag bit
    reset                   ;Software reset
```

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X							

34. Module: Data EEPROM

At device throughput greater than 25 MIPS, read operations performed on Data EEPROM may not function correctly.

Work around

When reading data from Data EEPROM, the application should perform a clock-switch operation to lower the frequency of the system clock so that the throughput is less than 25 MIPS. This may be easily performed at any time via the Oscillator Postscaler bits (POST), (OSCCON<7:6>), that allow the application to divide the system clock down by a factor of 4, 16 or 64.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X							

35. Module: Interrupt Controller

A specific write sequence for the Interrupt Priority Control 2 (IPC2) SFR is required to prevent possible data corruption in the Interrupt Enable Control 2 (IEC2) SFR. Interrupts must be disabled during this IPC2 SFR write sequence.

Work around

An example of this write sequence is shown in Example 18.

EXAMPLE 18:

```
mov  #IPC2, w0      ;Point w0 to IPC2
mov  #0x4444, w1    ;Write data to go to IPC2
disi #2             ;Disable interrupts for
                    ;next two cycles
mov  w1, IPC2       ;Write the data to IPC2
mov  #IPC2, w0      ;Target w1 to keep IPC2
                    ;address on bus
```

When coding in C, the write sequence shown above can be implemented using inline Assembly instructions. The equivalent write sequence using the MPLAB C Compiler for dsPIC DSCs is shown in Example 19.

EXAMPLE 19:

```
asm volatile(   "push.d   w0\n\t"
                "mov  #IPC2,w0\n\t"
                "mov  #0x4444,w1\n\t"
                "disi #2\n\t"
                "mov  w1, IPC2\n\t"
                "mov  #IPC2, w0\n\t"
                "pop.d  w0");
//Note: There are no commas between
//      the quoted strings in the code
//      segment above.
```

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X							

36. Module: Sleep Mode

The device exhibits IPD of approximately 100 µA.

Work around

If the application does not use the on-chip A/D converter, it is possible to reduce the IPD to values below 0.1 µA. The following additional measures need to be taken in these circumstances:

1. In the application hardware, the VREF+/RB0 pin (pin 2) on the dsPIC30F2010 should be connected to the circuit ground (GND).
2. In the application software, the code sequence shown in Example 20 should be executed to bring the device into the power-saving Sleep mode.

EXAMPLE 20:

```
    .include "p30f2010.inc"
    .......
    BCLR    ADCON1, #ADON   ;Required code
    MOV     #0x2000, W0     ;sequence for
    MOV     W0, ADCON2      ;low power-down
    BCLR    PMD1, #ADCMD    ;current.
    PWRSAV  #SLEEP_MODE     ;Device enters
                            ;SLEEP mode here
```

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X							

37. Module: QEI

When the TQCS and TQGATE bits in the QEIxCON register are set, a QEI interrupt should be generated after an input pulse on the QEA input. This interrupt is not generated in the affected silicon.

Work around

None.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

38. Module: QEI

When the TQCS and TQGATE bits in the QEIxCON register are set, the POSCNT counter should not increment but erroneously does, and if allowed to increment to match MAXCNT, a QEI interrupt will be generated.

Work around

To prevent the erroneous increment of POSCNT while running the QEI in Timer Gated Accumulation mode, initialize MAXCNT = 0.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

39. Module: ADC

If the ADC module is in an enabled state when the device enters Sleep mode as a result of executing a PWRSAV #0 instruction, the device power-down current (I_{PD}) may exceed the specifications listed in the device data sheet. This may happen even if the ADC module is disabled by clearing the ADON bit prior to entering Sleep mode.

Work around

In order to remain within the I_{PD} specifications listed in the device data sheet, the user software must completely disable the ADC module by setting the ADC Module Disable bit in the corresponding Peripheral Module Disable register (PMDx), prior to executing a PWRSAV #0 instruction.

Affected Silicon Revisions

A0	A1	A2	A3	A4			
X	X	X	X	X			

Data Sheet Clarifications

The following typographic corrections and clarifications are to be noted for the latest version of the device data sheet (DS70118**H**):

Note:	Corrections are shown in **bold**. Where possible, the original bold text formatting has been removed for clarity.

1. Module: DC Characteristics: I/O Pin Input Specifications

The maximum value for parameter DI19 (V_{IL} specifications for SDAx and SCLx pins) and the minimum value for parameter DI29 (V_{IH} specifications for SDAx and SCLx pins) were stated incorrectly in Table 22-8 of the current device data sheet. The correct values are shown in bold type in Table 4.

TABLE 4: DC CHARACTERISTICS: I/O PIN INPUT SPECIFICATIONS

DC CHARACTERISTICS			Standard Operating Conditions: 3.3V and 5.0V (±10%) (unless otherwise stated) Operating temperature -40°C ≤TA ≤+85°C for Industrial -40°C ≤TA ≤+125°C for Extended				
Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
DI19	V_{IL}	Input Low Voltage SDA, SCL	V_{SS}	—	**0.8**	V	SMbus enabled
DI29	V_{IH}	Input High Voltage SDA, SCL	**2.1**	—	V_{DD}	V	SMbus enabled

2. Module: DC Characteristics: Power-Down Current (I_{PD})

The maximum value for parameter DC60g (I_{PD} Power-Down Current current at 125ºC), was stated incorrectly in Table 27-7 of the current device data sheet. The correct value is shown in bold type in Table 5.

TABLE 5: DC CHARACTERISTICS: POWER-DOWN CURRENT (I_{PD})

DC CHARACTERISTICS			Standard Operating Conditions: 2.5V to 5.5V (unless otherwise stated) Operating temperature -40°C ≤TA ≤+85°C for Industrial -40°C ≤TA ≤+125°C for Extended			
Param No.	Typical	Max	Units	Conditions		
Power-Down Current (I_{PD})(1)						
DC60g	40	**200**	µA	125°C	5V	Base Power-Down Current

Note 1: Base I_{PD} is measured with all peripherals and clocks shut down. All I/Os are configured as inputs and pulled high. LVD, BOR, WDT, etc. are all switched off.

APPENDIX A: REVISION HISTORY

Rev A Document (4/2009)

Initial release of this document; issued for revision A0, A1, A2, A3 and A4 silicon.

Includes silicon issues 1-3 (CPU), 4 (PSV Operations), 5-7 (CPU), 8 (Timer), 9-10 (Output Compare), 11-12 (ADC), 13 (Watchdog Timer), 14 (PLL), 15 (Interrupt Controller), 16 (PLL), 17 (SPI), 18 (QEI), 19 (Sleep Mode), 20 (I^2C), 21 (PWM), 22 (I/O), 23 (FRC), 24 (I^2C), 25 (Timer), 26 (PLL), 27 (PSV Operations), 28-30 (I^2C), 31 (Program Flash Memory), 32 (Data EEPROM), 33 (Program Flash Memory), 34 (Data EEPROM), 35 (Interrupt Controller) and 36 (Sleep Mode).

This document replaces the following errata documents:

- DS80178, *"dsPIC30F2010 Rev. A0 Silicon Errata"*
- DS80186, *"dsPIC30F2010 Rev. A1 Silicon Errata"*

Rev B Document (7/2009)

Updated silicon issue 15 (Interrupt Controller).

Added silicon issues 37 (QEI) and 38 (QEI).

Rev C Document (2/2010)

Updated silicon issue 15 (Interrupt Controller).

Rev D Document (6/2010)

Added silicon issue 39 (ADC) and data sheet clarification 1 (DC Characteristics: I/O Pin Input Specifications).

Rev E Document (10/2010)

Added data sheet clarification 2 (DC Characteristics: Power-Down Current (I_{PD})).

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

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

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

Trademarks

Printed on recycled paper.

ISBN: 978-1-60932-583-1

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

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

Worldwide Sales and Service

AMERICAS

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

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

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

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

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

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

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

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

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

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

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

ASIA/PACIFIC

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

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

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

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

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

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

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

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

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

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

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

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

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

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

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

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

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

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

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

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

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

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

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

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

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

Taiwan - Hsin Chu
Tel: 886-3-6578-300
Fax: 886-3-6578-370

Taiwan - Kaohsiung
Tel: 886-7-213-7830
Fax: 886-7-330-9305

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

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

EUROPE

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

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

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

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

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

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

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

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

08/04/10