dsPIC30F5011/5013 Rev. A3 Silicon Errata

The dsPIC30F5011/5013 (Rev. A3) samples that you have received were found to conform to the specifications and functionality described in the following documents:

- DS70157 – "*dsPIC30F/33F Programmer's Reference Manual*"
- DS70116 – "*dsPIC30F5011, dsPIC30F5013 Data Sheet*"
- DS70046 – "*dsPIC30F Family Reference Manual*"

The exceptions to the specifications in the documents listed above are described in this section. These exceptions are described for the specific devices listed below:

- dsPIC30F5011
- dsPIC30F5013

These devices may be identified by the following message that appears in the MPLAB® ICD 2 Output Window under MPLAB IDE, when a "Reset and Connect" operation is performed within MPLAB IDE:

```
Setting Vdd source to target
Target Device dsPIC30F5013 found,
revision = 0x1003
...Reading ICD Product ID
Running ICD Self Test
...Passed
MPLAB ICD 2 Ready
```

The errata described in this section will be fixed in future revisions of dsPIC30F5011 and dsPIC30F5013 devices.

Silicon Errata Summary

The following list summarizes the errata described in this document:

1. `MAC` Class Instructions with ±4 Address Modification

   Sequential `MAC` instructions, which prefetch data from Y data space using ±4 address modification, will cause an address error trap.

2. Decimal Adjust Instruction

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

3. PSV Operations Using SR

   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.

4. Early Termination of Nested `DO` Loops

   When using two `DO` loops in a nested fashion, terminating the inner-level `DO` loop by setting the EDT (CORCON<11>) bit will produce unexpected results.

5. I^2C™ – Read Operations on I2CCON SFR

   Read operations performed on the I2CCON SFR may yield incorrect results at operation over 20 MIPS.

6. I^2C – Write Operations on I2CTRN SFR

   Write operations performed on the I2CTRN SFR may yield incorrect results at operation over 20 MIPS.

7. UART – Write Operations on U1MODE and U2MODE SFRs

   Write operations performed on the U1MODE and U2MODE SFRs may yield incorrect results at operation over 20 MIPS.

8. DCI – Stop in Idle mode

   The DCI module should not be stopped when the device enters Idle mode.

9. 4x PLL Operation

   The 4x PLL mode of operation may not function correctly for certain input frequencies.

10. Sequential Interrupts

    Sequential interrupts after modifying the CPU IPL, interrupt IPL, interrupt enable or interrupt flag may cause an address error trap.

11. `DISI` Instruction

    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.

12. Output Compare Module in PWM Mode

    Output compare will produce a glitch when loading 0% duty cycle in PWM mode. It will also miss the next compare after the glitch.

13. Output Compare Module

    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.

14. INT0, ADC and Sleep Mode

    ADC event triggers from the INT0 pin will not wake-up the device from Sleep mode if the SMPI bits are non-zero.

15. 8x PLL Mode

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

16. 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.

17. I^2C Module

    The I^2C module loses incoming data bytes when operating as an I^2C slave.

18. I/O Port – Port Pin Multiplexed with IC1

    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.

19. I^2C Module: 10-bit Addressing Mode

    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.

20. Timer Module

    Clock switching prevents the device from waking up from Sleep.

21. PLL Lock Status Bit

    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.

22. PSV Operations

    An address error trap occurs in certain addressing modes when accessing the first four bytes of any PSV page.

23. I^2C Module: 10-bit Addressing Mode

    The 10-bit slave does not set the RBF flag or load the I2CxRCV register on address match if the Least Significant bits of the address are the same as the 7-bit reserved addresses.

24. I^2C Module: 10-bit Addressing Mode

    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.

25. I^2C Module

    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.

The following sections will describe the errata and work around to these errata, where they may apply.

1. Module: MAC Class Instructions with ±4 Address Modification

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

The problem described above can be avoided by 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.

2. Module: CPU – DAW.b Instruction

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 set, set the Carry bit again after executing the DAW.b instruction. Example 1 shows how the application should process the Carry bit during a BCD addition operation.

EXAMPLE 1: CHECK CARRY BIT BEFORE DAW.b

```
    .include "p30f5013.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: ....
```

3. Module: PSV Operations Using SR

When one of the operands of instructions shown in Table 1 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 1. Example 2 demonstrates one scenario where this occurs.

Also, always use Work around 2 if the C compiler is used to generate code for dsPIC30F5011/5013 devices.

TABLE 1: 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 (CORCON<2>) bit 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 Sticky Zero and Carry Status bits, respectively.

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

EXAMPLE 2: 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 3: 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 ok!
```

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 1. The work around for Example 2 is demonstrated in Example 3.

Work around 2: For C Language Source Code

For applications using C language, MPLAB C30 versions 1.20.04 or higher provide the following command-line switch that implements a work around for the erratum.

```
-merrata=psv
```

Refer to the "`readme.txt`" file in the MPLAB C30 v1.20.04 toolsuite for further details.

4. Module: Early Termination of Nested DO Loops

When using two DO loops in a nested fashion, terminating the inner-level DO loop by setting the EDT (CORCON<11>) bit 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 C30 compiler.

Work around

The application should save the DCOUNT SFR prior to entering the inner DO loop and restore it upon exiting the inner DO loop. This work around is shown in Example 4.

EXAMPLE 4: 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
        EDT bit, see section 2.9.2.4
        in the dsPIC30F Family Reference
        Manual.
```

5. Module: I^2C – Read Operations on I2CCON SFR

Data read from the I2CCON Special Function Register (SFR) may not be correct at device operation greater than 20 MIPS for V_{DD} in the range of 4.5V to 5.5V (or 10 MIPS V_{DD} in the range of 3V to 3.6V).

If the dsPIC DSC device needs to operate at a throughput higher than 20 MIPS, the user should incorporate the suggested work around while reading the I2CCON SFR.

Applications that use I^2C software functions from Microchip's dsPIC30F Peripheral Library, should operate the device at 20 MIPS or less.

Work arounds

Work around 1: For Assembly Language Source Code

When reading the I2CCON SFR, perform two consecutive read operations of the same SFR. The work around is demonstrated in Example 5. In this example, a Memory Direct Addressing mode is used to read the SFR. The application may use any addressing mode to perform the read operation. Note that interrupts must be temporarily disabled as shown, so that the two consecutive reads do not get interrupted.

EXAMPLE 5: CONSECUTIVE READS

```
    .include “p30fxxxx.inc”
    .......
    PUSH    SR
    BSET    SR, #IPL2
    BSET    SR, #IPL1
    BSET    SR, #IPL0
    MOV     I2CCON, W0  ; first SFR read
    MOV     I2CCON, W0  ; second SFR read
    POP     SR
```

Work around 2: For C Language Source Code

For C programmers, the MPLAB C30 v1.20.02 toolsuite provides a built-in function that may be incorporated in the application source code. This function may be used to read the I2CCON SFR. Some examples of usage are shown in the “readme.txt” file provided with the MPLAB C30 v1.20.02 toolsuite. The function has the following prototype:

```
unsigned __builtin_readsfr(volatile void *);
```

The special argument is the address of a 16-bit SFR (I2CCON in this case). This function should only be used to read the I2CCON Special Function Register. For example, the I2CCON register can be read using a function call:

```
reg_value = __builtin_readsfr(&I2CCON);
```

where ‘reg_value’ is the 16-bit value read from the SFR.

6. Module: I^2C – Write Operations on I2CTRN SFR

Data writes to the I2CTRN Special Function Register (SFR) may not be correct at device operation greater than 20 MIPS for VDD in the range of 4.5V to 5.5V (or 10 MIPS VDD in the range of 3V to 3.6V).

If the dsPIC DSC device needs to operate at a throughput higher than 20 MIPS, the user should incorporate the suggested work around while writing to the I2CTRN SFR.

Applications that use I^2C software functions from Microchip's dsPIC30F Peripheral Library should operate the device at 20 MIPS or less.

Work arounds

Work around 1: For Assembly Language Source Code

When writing to the I2CTRN SFR, the user must follow the write sequence shown in Example 6. In this example, a Memory Direct Addressing mode is used to write to the SFR. The application may use any addressing mode to perform the write operation. Note that interrupts must be temporarily disabled as shown, so that this write sequence does not get interrupted.

EXAMPLE 6: SPECIAL WRITE SEQUENCE

```
.include “p30fxxxx.inc”
.......
MOV    #reg_value, W1 ;I2CTRN value
PUSH   SR
BSET   SR,  #IPL2
BSET   SR,  #IPL1
BSET   SR,  #IPL0
MOV    #I2CTRN, W0    ;write I2CTRN
                      ;address to W0
MOV    W0, W0         ;perform a direct
                      ;write to W0
MOV    W1, I2CTRN     ;write to I2CTRN
POP    SR
```

Work around 2: For C Language Source Code

For C programmers, the MPLAB C30 v1.30 toolsuite provides a built-in function that may be incorporated in the application source code. This function may be used to write to the I2CTRN SFR. Some examples of usage are shown in the “`readme.txt`” file provided with the MPLAB C30 v1.30 toolsuite. The function has the following prototype:

```
void __builtin_writesfr(volatile void *,
unsigned int);
```

The special argument is the address of a 16-bit SFR (I2CTRN in this case). For example, the I2CTRN register can be written using a function call:

```
__builtin_writesfr(&I2CTRN, reg_value);
```

where ‘`reg_value`’ is the 16-bit value to be written to the SFR.

7. Module: UART – Write Operations on U1MODE and U2MODE SFRs

Data writes to the U1MODE and U2MODE Special Function Registers (SFRs) may not be correct at device operation greater than 20 MIPS for VDD in the range of 4.5V to 5.5V (or 10 MIPS VDD in the range of 3V to 3.6V).

If the dsPIC DSC device needs to operate at a throughput higher than 20 MIPS, the user should incorporate the suggested work around while writing to the U1MODE or U2MODE SFR.

Applications that use UART software functions from Microchip's dsPIC30F Peripheral Library should operate the device at 20 MIPS or less.

Work arounds

Work around 1: For Assembly Language Source Code

When writing to the U1MODE (or U2MODE) SFR, the user must follow the write sequence shown in Example 7. In this example, a Memory Direct Addressing mode is used to write to the SFR. The application may use any addressing mode to perform the write operation. Note that interrupts must be temporarily disabled as shown, so that this write sequence does not get interrupted.

EXAMPLE 7: SPECIAL WRITE SEQUENCE

```
.include “p30fxxxx.inc”
.......
MOV    #reg_value, W1 ;U1MODE value
PUSH   SR
BSET   SR,  #IPL2
BSET   SR,  #IPL1
BSET   SR,  #IPL0
MOV    #U1MODE, W0    ;write U1MODE
                      ;address to W0
MOV    W0, W0         ;perform a direct
                      ;write to W0
MOV    W1,  U1MODE    ;write to U1MODE
POP    SR
```

Work around 2: For C Language Source Code

For C programmers, the MPLAB C30 v1.30 toolsuite provides a built-in function that may be incorporated in the application source code. This function may be used to write to the U1MODE and U2MODE SFRs. Some examples of usage are shown in the “`readme.txt`” file provided with the MPLAB C30 v1.30 toolsuite. The function has the following prototype:

```
void __builtin_writesfr(volatile void *,
unsigned int);
```

The special argument is the address of a 16-bit SFR (U1MODE or U2MODE in this case). For example, the U1MODE register can be written using a function call:

```
__builtin_writesfr(&U1MODE, reg_value);
```

where ‘reg_value’ is the 16-bit value to be written to the SFR.

8. Module: Data Converter Interface – Idle

For this release of silicon, the DCI module should not be stopped when the device enters Idle mode.

Work around

Do not set the DCISIDL (DCICON1<13>) bit. This will ensure the DCI module continues to run when the device enters Idle mode.

9. Module: 4x PLL Operation

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

When device VDD is 2.5-3.0V, the 4x PLL input frequency must be in the range of 4-5 MHz. When device VDD is 3.0-3.6V, the 4x PLL input frequency must be in the range of 4-6 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.

10. Module: Interrupt Controller – Sequential Interrupts

When interrupt nesting is enabled (or NSTDIS (INTCON1<15>) bit is ‘0’), the following sequence of events will lead to an address error trap. The generic terms “Interrupt 1” and “Interrupt 2” are used to represent any two enabled dsPIC30F interrupts.

1. Interrupt 1 processing begins.
2. Interrupt 1 is negated by user software by one of the following methods:
   - 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
3. Interrupt 2 occurs with a priority higher than Interrupt 1.

Work around

The user may disable interrupt nesting or execute a DISI instruction before modifying the CPU IPL or Interrupt 1 setting. A minimum DISI value of 2 is required if the DISI is executed immediately before the CPU IPL or Interrupt 1 is modified, as shown in Example 8. If the MPLAB C30 compiler is being used, one must inspect the Disassembly Listing in the MPLAB IDE file to determine the exact number of cycles to disable level 1-6 interrupts. One may use a large DISI value and then set the DISICNT register to zero, as shown in Example 9. A macro may also be used to perform this task, as shown in Example 10.

EXAMPLE 8: USING DISI

```
.include       “p30fxxxx.inc”
...
DISI#2  ; protect the disable of INT1
BCLRIEC1, #INT1IE; disable interrupt 1
...    ; next instruction protected by DISI
```

EXAMPLE 9: RAISING CPU INTERRUPT PRIORITY LEVEL

```
.include       “p30fxxxx.h”
...
__asm__ volatile (“DISI #0x1FFF”);   // protect CPU IPL modification
SRbits.IPL = 0x5;                    // set CPU IPL to 5
DISICNT = 0x0;                       // remove DISI protection
```

EXAMPLE 10: USING MACRO

```
#define DISI_PROTECT(X) {\
    __asm__ volatile (“DISI #0x1FFF”);\
    X;                 \
    DISICNT = 0; }

DISI_PROTECT(SRbits.IPL = 0x5);   // safely modify the CPU IPL
```

11. Module: DISI Instruction

When a user executes a DISI #7, for example, this will disable interrupts for 7 + 1 cycles (7 + the DISI instruction itself). In this case, the DISI instruction uses a counter which 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 continue 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.

In summary, 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 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, make sure that the subsequent DISI instructions are called before the DISI counter decrements to zero.

12. Module: Output Compare in PWM Mode

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.

13. 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 = 0x0002 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.

14. Module: INT0, ADC and Sleep Mode

ADC event triggers from the INT0 pin will not wake-up the device from Sleep mode if the SMPI bits are non-zero. This means 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 ADC event trigger from the INT0 pin is required, initialize SMPI<3:0> to ‘0000’ (interrupt on every conversion).

15. Module: 8x PLL Mode

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, make sure the input crystal or clock frequency is 5 MHz or greater.

16. 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, any of the following three work arounds can be implemented, 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 utilized 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 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 11 demonstrates the work around described above, as it would apply to a dsPIC30F5011 device.

EXAMPLE 11:

```
; -----------------------------------------------------------------------------------------------
.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 (0xAFC0)
_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 Low-Power (LP) Oscillator 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.

17. Module: I^2C

When the I^2C module is configured as a slave, either in single-master or multi-master mode, the I^2C receiver buffer is filled whether a valid slave address is detected or not. Therefore, an I^2C 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 I^2C receive interrupt flag (SI2CF) when the last valid data byte is received. Therefore, the I^2C slave Interrupt Service Routine (ISR) is not called and the I^2C 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 I^2C 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 I^2C receiver interrupt SI2CIF flag.
3. If SI2CF is not set in the corresponding Interrupt Flag Status (IFSx) register, a valid address or data byte has not been received for the current slave. Execute a dummy read of the I^2C 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 (IFSx) register, 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 I^2C 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 I^2C 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 I^2C slave interrupt service routine is called; however, the RBF and I2COV bits are already set due to data transfers between other I^2C nodes.
2. Check the status of the D_A flag and the I2COV flag in the I2CSTAT register when executing the I^2C slave service routine.
3. If the D_A flag is cleared and the I2COV flag are set, an invalid data byte was received but a valid address byte was received. The overflow condition occurred because the I^2C receive buffer was overflowing with previous I^2C data transfers between other I^2C nodes. This condition only occurs after a valid slave address was detected.
4. Clear the I2COV flag and perform a dummy read of the I^2C 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.

18. Module: I/O Port – Port Pin Multiplexed with IC1

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

Work around

None.

19. Module: I²C

If there are two I²C devices on the bus, one of them is acting as the Master receiver and the other 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, then 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 I²C devices, the addresses as well as bits A10 and A9 should be different.

20. 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).

21. Module: PLL Lock Status Bit

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.

22. Module: PSV Operations

An address error trap occurs in certain addressing modes when accessing the first four bytes of an 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 C30 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 C30 v3.11 tool suite for further details.

23. Module: I²C

In 10-bit Addressing mode, some address matches don't 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.

24. Module: I²C

When the I²C module is configured as a 10-bit slave with and address of 0x102, the I2CxRCV register content for the lower address byte is 0x01 rather than 0x02; however, the module acknowledges both address bytes.

Work around

None.

25. Module: I^{2}C

When the I^{2}C 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 I^{2}C 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 I^{2}C 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 I^{2}C 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 I^{2}C masters should be synchronized and wait for the I^{2}C module to be initialized before initiating any kind of communication.

Work around 2:

In dsPIC DSC devices in which the I^{2}C 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 I^{2}C 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 I^{2}C module.
2. Set up and enable the I^{2}C module.

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.

APPENDIX A: REVISION HISTORY

Revision A (1/2005)

Original version of the document.

Revision B (3/2005)

Added silicon issues 8 (PLL) and 9 (Interrupt Controller – Sequential Interrupts).

Revision C (4/2005)

Added silicon issue 10 (Using OSC2/RC15 pin for Digital I/O).

Revision D (9/2006)

Added errata 1, 11, 12, 13, 15 and 16.

Revision E (9/2007)

Added silicon issue 17 (Sleep Mode).

Revision F (12/2007)

Updated silicon issue 3 (PSV Operations Using SR), and added silicon issues 18 and 19 (I^2C), and 20 (I/O Port – Port Pin Multiplexed with IC1).

Revision G (5/2008)

Added silicon issues 20 and 21 (I^2C), and 22 (Timer). Removed silicon issue 14 (Using OSC2/RC15 pin for Digital I/O).

Revision H (9/2008)

Replaced issues 17 and 20 (I^2C) with issue 25 (I^2C). Added silicon issues 21 (PLL Lock Status Bit), 22 (PSV Operations) and 23-25 (I^2C).

MICROCHIP

Worldwide Sales and Service

AMERICAS

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

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

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

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

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

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

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

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

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

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

ASIA/PACIFIC

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

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

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

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

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

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

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

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

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

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

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

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

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

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

ASIA/PACIFIC

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

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

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

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

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

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

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

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

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

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

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

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

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

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

EUROPE

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

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

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

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

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

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

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

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

01/02/08