dsPIC30F6011/6012/6013/6014

dsPIC30F6011/6012/6013/6014 Rev. B2 Silicon Errata

The dsPIC30F6011/6012/6013/6014 (Rev. B2) 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*"
- DS70117 – "*dsPIC30F6011/6012/6013/6014 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:

- dsPIC30F6011
- dsPIC30F6012
- dsPIC30F6013
- dsPIC30F6014

dsPIC30F601X Rev. B2 silicon is identified by performing a "Reset and Connect" operation to the device using MPLAB® ICD 2 within the MPLAB IDE. The following text is then visible under the MPLAB ICD 2 section in the output window within MPLAB IDE:

```
MPLAB ICD 2 Ready
Connecting to MPLAB ICD 2
...Connected
Setting Vdd source to target
Target Device dsPIC30F6014 found,
revision = mss1.b rev b2
...Reading ICD Product ID
Running ICD Self Test
...Passed
MPLAB ICD 2 Ready
```

The errata described in this section will be addressed in future revisions of dsPIC30F6011, dsPIC30F6012, dsPIC30F6013 and dsPIC30F6014 silicon.

Silicon Errata Summary

The following list summarizes the errata described in this document:

1. Data EEPROM

   Data EEPROM is operational at 20 MIPS.

2. Unsigned MAC

   The Unsigned Integer mode for the MAC type DSP instructions does not function as specified.

3. 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.

4. Decimal Adjust Instruction

   The decimal adjust instruction, DAW.b, may improperly clear the Carry bit, C (SR<0>).

5. 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.

6. 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.

7. Reset During Run-Time Self-Programming (RTSP) of Program Flash Memory

   When a device Reset occurs while an RTSP operation is ongoing, code execution may lead into an address error trap.

8. Y Data Space Dependency

   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.

9. Catastrophic Overflow Traps

   When a catastrophic overflow of any of the accumulators causes an arithmetic (math) error trap, the Overflow Status bits need to be cleared to exit the trap handler.

10. Interrupting a REPEAT Loop

    When a REPEAT loop is interrupted by two or more interrupts in a nested fashion, an address error trap may be caused.

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.

© 2008 Microchip Technology Inc.

12. 32-bit General Purpose Timers

    The 32-bit general purpose timers do not function as specified for prescaler ratios other than 1:1.

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. 12-bit 100 ksps Analog-to-Digital Converter (ADC)

    The 12-bit ADC scans one channel less than that specified when configured to perform channel scanning on MUX A inputs and alternately converting a fixed MUX B input.

15. Data Converter Interface – Slave Mode

    In Slave mode, the DCI module does not function correctly when data communication is configured to start one serial clock after the frame synchronization pulse.

16. DCI – Stop in Idle mode

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

17. CAN SFR Reads

    Read operations performed on CAN module Special Function Registers (SFRs) may yield incorrect results at operation over 20 MIPS.

18. High I_{DD} During Row Erase of Program Flash Memory

    This release of silicon exhibits a current draw (I_{DD}) of approximately 370 mA during a Row Erase operation performed on program Flash memory.

19. Regulating Voltage for 5V/30 MIPS Applications

    For this release of silicon, applications operating off 5 volts V_{DD} at 30 MIPS should ensure that the V_{DD} remains within 5% of 5 volts.

20. dsPIC30F6011/6013 Code Protection

    Addresses in the range 0x6000 through 0xFFFF may not be code-protected for this revision of dsPIC30F6011 and dsPIC30F6013 silicon.

21. 4x PLL Operation

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

22. Sequential Interrupts

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

23. 8x PLL Mode

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

24. 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.

25. I^2C™ Module

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

26. 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.

27. 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.

28. Timer Module

    Clock switching prevents the device from waking up from Sleep.

29. 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.

30. PSV Operations

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

31. 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.

32. 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.

33. 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 erratas and the work around to these erratas, where they may apply.

© 2008 Microchip Technology Inc.

1. Module: Data EEPROM – Speed

At device throughput greater than 20 MIPS for VDD, in the range 4.75V to 5.5V (or 10 MIPS for VDD in the range 3V to 3.6V), table read instructions (TBLRDL/TBLRDH) and instructions that use Program Space Visibility (PSV) do not function correctly when reading data from data EEPROM.

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 20 MIPS. This may be easily performed at any time via the Oscillator Postscaler bits, POST<1:0> (OSCCON<7:6>), that allow the application to divide the system clock down by a factor of 4, 16 or 64.

2. Module: CPU – Unsigned MAC

The US (CORCON<12>) bit controls whether MAC type DSP instructions operate in Signed or Unsigned mode. The device defaults to a Signed mode on power-up (US = 0).

For this revision of silicon, MAC type DSP instructions do not function as specified in Unsigned mode (US = 1). Also, for this revision, the US bit will always read as '0'.

Work around

Ensure that the US bit is not set by the application. In order to perform unsigned integer multiplications, use the MCU Multiply instruction, MUL.UU.

3. 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.

4. 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 "p30fxxxx.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: ....
```

5. 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 a scenario where this occurs.

Also, always use Work around 2 if the C compiler is used to generate code for dsPIC30F6011/6012/6013/6014 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*", DS70046, 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!
```

<u>**Work arounds**</u>

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.

© 2008 Microchip Technology Inc.

6. 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.
```

7. Module: Reset During RTSP of Program Flash Memory

If a device Reset occurs while an RTSP operation is ongoing, 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 5, in order to allow normal code execution to continue.

EXAMPLE 5: TRAP SERVICE ROUTINE

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

8. Module: Y Data Space Dependency

When an instruction that writes to a location in the address range of Y data memory (addresses between 0x1800 and 0x27FF) 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 6.

EXAMPLE 6: INCORRECT RESULTS

```
    MOV     #0x190A, W0     ;Load address > =
                            ;0x1800 into W0
    MOV     #0x19B0, W10    ;Load address >=
                            ;0x1800 into W10
    MOV     W2, [W0++]      ;Perform indirect
                            ;write via W0 to
                            ;address >= 0x1800
    MAC     W4*W5, A, [W10]+=2, W5 ;Perform
                            ;read operation
                            ;using Y-AGU
:Unexpected Results!
```

Work arounds

Work around 1:

Insert a NOP between the two instructions as shown in Example 7.

EXAMPLE 7: CORRECT RESULTS

```
    MOV     #0x190A, W0     ;Load address > =
                            ;0x1800 into W0
    MOV     #0x19B0, W10    ;Load address >=
                            ;0x1800 into W10
    MOV     W2, [W0++]      ;Perform indirect
                            ;write via W0 to
                            ;address >= 0x1800
    NOP                     ;No operation
    MAC     W4*W5, A, [W10]+=2, W5 ;Perform
                            ;read operation
                            ;using Y-AGU
:Correct Results!
```

Work around 2:

If work around #1 is not feasible due to application real-time 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.

9. Module: Interrupt Controller –Traps

Catastrophic accumulator overflow traps are enabled as follows:

- COVTE (INTCON1<8>) = 1
- SATA/SATB (CORCON <7:6>) = 0

A carry generated out of bit 39 in the accumulator causes a catastrophic overflow of the accumulator since the sign bit has been destroyed. If a math error trap handler has been defined, the processor will vector to the math error trap handler upon a catastrophic overflow.

If the respective Accumulator Overflow status bit, OA or OB (SR<15/14>), is not cleared within the trap handler routine prior to exiting the trap handler routine, the processor will immediately re-enter the trap handler routine.

Work around

If a math error trap occurs due to a catastrophic accumulator overflow, the overflow status flags, OA and/or OB (SR<15:14>), should be cleared within the trap handler routine. Subsequently, the MATHERR (INTCON1<4>) flag bit should be cleared within the trap handler prior to executing the RETFIE instruction.

Since the OA and OB bits are read-only bits, it will be necessary to execute a dummy accumulator-based instruction within the trap service routine in order to clear these status bits, and eventually clear the MATHERR trap flag. This is shown in Example 8.

EXAMPLE 8: USING DUMMY DSP INSTRUCTION

```
.global     __MathError
__MathError:   BTSC    SR, #OA
               CLR     A
               BTSC    SR, #OB
               CLR     B
               BCLR    INTCON1, #MATHERR
               RETFIE
```

10. Module: Interrupting a REPEAT Loop

When interrupt nesting is enabled (or NSTDIS (INTCON1<15>) bit 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 9 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 9: 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> (SR<7:5>) bits to '111' as shown in Example 10. This will disable all interrupts between priority levels 1 through 7.

EXAMPLE 10: 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
```

 © 2008 Microchip Technology Inc.

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

12. Module: 32-bit General Purpose Timers

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.

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: 12-bit 100 ksps ADC

Input channel scanning allows the ADC to acquire and convert signals on a selected set of “MUX A” input pins in sequence. This function is controlled by the CSCNA (ADCON2<10>) bit and the ADCSSL SFR.

The ALTS (ADCON2<0>) bit, when set, allows the ADC to alternately acquire and convert a “MUX A” input signal and a “MUX B” input signal in an interleaved fashion.

When both CSCNA and ALTS are set, the ADC module should scan MUX A input pins while alternating with a fixed MUX B input pin. However, for this release of silicon, when both features are enabled simultaneously, the last input pin enabled for channel scanning in the ADCSSL SFR is not scanned. Thus, the ADC converts one channel less than the number specified in the scan sequence. Note that this erratum does not affect devices that have a 10-bit 500 ksps ADC.

Work around

The user may enable an extra (“dummy”) input pin in the channel-scanning sequence. For example, if it is desirable to scan pins AN3, AN4 and AN5 on the set of MUX A inputs while interleaving conversion from AN6 on the MUX B input, the user may configure the ADC as follows:

- ADCON2 = 0x041D
- ADCHS = 0x0600
- ADCSSL = 0x8038

For the configuration above, AN15 is the dummy input that will not be scanned. On the A/D interrupt, the A/D buffer will contain conversions from the following pins in sequence:

- ADCBUF0 = AN3
- ADCBUF1 = AN6
- ADCBUF2 = AN4
- ADCBUF3 = AN6
- ADCBUF4 = AN5
- ADCBUF5 = AN6
- ADCBUF6 = AN3
- ADCBUF7 = AN6

15. Module: Data Converter Interface – Slave Mode

The Data Converter Interface (DCI) module does not function correctly in Slave mode when the following conditions are true:

- The DCI module is configured to transmit/receive one serial clock (bit clock) after the frame synchronization pulse, DJST (DCICON1<5>) = `0`.
- The frame length chosen is longer than 1 word, COFSG<3:0> (DCICON2<8:5>) > ‘`0000`’.

Work around

The following work around may be applied to enable DCI communication in Slave mode when it is configured to transmit one serial clock after the frame synchronization pulse is received in a multi-word frame:

1. Set the DJST bit to ‘`1`’.
2. Enable an additional time slot immediately following each time slot intended for communication.
3. Enable an additional transmit/receive buffer word (modify COFSG bits) or an additional bit per word (modify WS) for each time slot intended for communication.
4. Shift the data word by 1-bit to the right and load the transmit buffer word(s), such that the Least Significant bit (LSb) of the original data word to be transmitted is loaded into the additionally enabled bit of the Transmit Buffer register, TXBUFn, or the Most Significant bit (MSb) of the additionally enabled transmit buffer, TXBUFn + 1.

This work around is now demonstrated by an example.

Assume, the application needs the DCI module to act as a Slave transmitting 1 serial clock after the frame synchronization pulse is received. Further, assume that the application needs to transmit 16-bit data word on Time Slot 0 and the communication is over a 256*Fs channel. In order to reduce interrupt frequency, we enable all 4 transmit buffers. The DCI module SFRs should be initialized as follows before being enabled:

- DCICON1 = 0x0720, DCICON2 = 0x0DEF
  DCICON3 = 0x0000,
  TSCON = RSCON = 0x0003

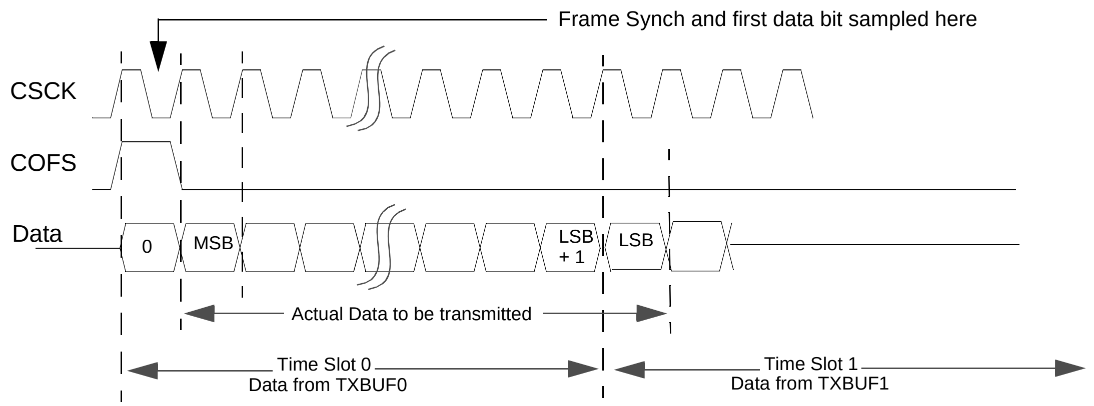
An example of loading the DCI transmit buffers for the configuration above is shown in Example 11. A timing diagram in Figure 1 illustrates the various signals for this example. A similar rule may be applied to reading the received data from the RXBUFn SFRs.

© 2008 Microchip Technology Inc.

EXAMPLE 11: DCI SLAVE WORK AROUND

```
BCLR    SR, #C
MOV     My1stTxDataWord, W0
RRC     W0, W0
RRC     W1, W1
MOV     W0, TXBUF0
MOV     W1, TXBUF1
MOV     My2ndTxDataWord, W0
RRC     W0, W0
RRC     W1, W1
MOV     W0, TXBUF2
MOV     W1, TXBUF3
```

FIGURE 1: DCI SLAVE WORK AROUND



Note 1: The Slave mode operation shown in this figure uses the DCI module's operation with DJST = `1`, to create a work around for the erratum associated with the DCI module when DJST = `0`.

2: Note that the actual data intended for transmission on Time Slot 0 is now straddled across two time slots – Time Slot 0 and Time Slot 1. The MSb of Time Slot 0 is '`0`'; while the MSb of Time Slot 1 is actually the LSb of the data intended for transmission.

3: Data loaded into TXBUF0 contains 15 MSbs of the actual 16-bit data to be transmitted, while the MSb of TXBUF0 is cleared.

4: Not all serial clock pulses are shown in this timing diagram.

16. 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.

17. Module: CAN – Read Operations on SFRs

Data read from the CAN module Special Function Registers may not be correct at device operation greater than 20 MIPS for V_{DD} in the range 4.75V to 5.5V (or 10 MIPS for V_{DD} in the range 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 arounds while reading CAN SFRs.

Applications that use Microchip's dsPIC30F Peripheral Library and Vector Informatik's CANbedded software, should operate the device at 20 MIPS or less.

Work arounds

Work around 1: For Assembly Language Source Code

When reading any CAN SFR, perform two consecutive read operations of that SFR. The work around is demonstrated in Example 12. 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 disabled so that the two consecutive reads do not get interrupted.

EXAMPLE 12: CONSECUTIVE READS

```
.include “p30f6014.inc”
....
disi    #1
mov     C1RXF0SIDL, w0 ; first SFR read
mov     C1RXF0SIDL, w0 ; second SFR read
```

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 any CAN module SFRs. 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 function argument is the address of a 16-bit SFR. This function should only be used to read the CAN Special Function Registers.

18. Module: High I_{DD} During Row Erase of Program Flash Memory

This release of silicon draws a current (I_{DD}) of approximately 370 mA during any Row Erase operation performed on program Flash memory.

Work arounds

Work around 1:

Supply the V_{DD} pin using a voltage regulator capable of sourcing a minimum of 300 mA of current.

Work around 2:

When using a voltage regulator capable of driving 150 mA current and if Brown-out Reset (BOR) is enabled for a V_{DD} greater than or equal to 4.2V, then connect a 1000 μF Electrolytic capacitor across the V_{DD} pin and ground.

If the Row Erase operation is performed as part of a Run-Time Self Programming (RTSP) operation, the user should ensure that the device is operating at less than 10 MIPS prior to the erase operation. To ensure that the device is operating at less than 10 MIPS, the application may post-scale the system clock or switch to the internal FRC oscillator.

© 2008 Microchip Technology Inc.

19. Module: Regulating Voltage for 5V/30 MIPS Applications

For this release of silicon, applications operating off 5 volts VDD at 30 MIPS should ensure the VDD remains between 4.75V and 5.5V. For 5V applications, Table 2 summarizes the maximum MIPS that can be achieved across various temperatures.

Work around

For 5 volt applications, use a voltage regulator that ensures VDD is in the range 4.75V to 5.5V, in order to achieve 30 MIPS operation.

TABLE 2: OPERATING MIPS VS. VOLTAGE

VDD Range (in volts)	Temp Range (in °C)	Max MIPS		
		dsPIC30FXXX-30I	dsPIC30FXXX-20I	dsPIC30FXXX-20E
4.75 to 5.5	-40 to +85	30	20	—
4.75 to 5.5	-40 to +125	—	—	20

Note: Applications that use the CAN peripherals and data EEPROM should also refer to Errata module 1 and 17.

20. Module: dsPIC30F6011/dsPIC30F6013 Code Protection

Addresses in the range, 0x6000 through 0xFFFF, may not be code-protected for this revision of dsPIC30F6011 and dsPIC30F6013 silicon.

Work around

None.

21. 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 2.5V-3.0V, the 4x PLL input frequency must be in the range of 4-5 MHz. When device VDD is 3.0V-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.

22. 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 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 2 is required if the DISI is executed immediately before the CPU IPL or Interrupt 1 is modified, as shown in Example 13. It is necessary to have DISI active for the cycle after the CPU IPL or Interrupt 1 is modified.

Work around 2: For C Language Source Code

For applications using C language, MPLAB C30 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 14. 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 15. These macros make use of the SET_CPU_IPL macro.

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 16. This macro is not distributed with the compiler.

EXAMPLE 13: USING DISI

```
.include       "p30fxxxx.inc"
...
DISI    #2              ; protect the disable of INT1
BCLR    IEC1, #INT1IE   ; disable interrupt 1
...                     ; next instruction protected by DISI
```

EXAMPLE 14: USING SET_CPU_IPL MACRO

```
// note: macro defined in p30f6014.h
#define SET_CPU_IPL (ipl){        \
    int DISI_save;                \
                                  \
    DISI_save = DISICNT;          \
    asm volatile ("disi #0x3FFF");\
    SRbits.IPL = ipl;             \
    DISICNT = DISI_save; } (void) 0;

#include "p30f6014.h"
. . .
SET_CPU_IPL (3)
. . .
```


© 2008 Microchip Technology Inc.

EXAMPLE 15: USING SET_AND_SAVE_CPU_IPL MACRO

```
// note: macros defined in p30f6014.h
#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 “p30f6014.h”
. . .
int save_to;
SET_AND_SAVE_CPU_IPL (save_to, 3)
. . .
RESTORE_CPU_IPL (save_to)
```

EXAMPLE 16: 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 C30 compiler version earlier than version 1.32, you may still use the macros by adding them to your application.

23. Module: 8x PLL Mode

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

Work around

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

24. 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 17 demonstrates the work around described above, as it would apply to a dsPIC30F6014 device.

EXAMPLE 17:

```
; -------------------------------------------------------------------------------------------
.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 (0x17FC0)
_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
```


© 2008 Microchip Technology Inc.

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.

25. 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 I^2C 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.

© 2008 Microchip Technology Inc.

26. 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.

27. Module: I^2C

If there are two I^2C 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 I^2C devices, the addresses as well as bits A10 and A9 should be different.

28. 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).

29. 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.

30. 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.

31. Module: I^2C

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.

32. Module: I^2C

When the I^2C 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.

33. Module: I^2C

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

Work around 2:

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

© 2008 Microchip Technology Inc.

APPENDIX A: REVISION HISTORY

Revision A (7/2004)

Original version of the document.

Revision B (11/2004)

Added errata 4, 5 and 17.

Revision C (3/2005)

Added errata 18 and 19.

Revision D (5/2005)

Added work arounds for assembly language and C language source code in errata issue 19.

Revision E (10/2006)

Added errata 3, 11, 13 and 23.

Revision F (9/2007)

Added silicon issue 24 (Sleep Mode).

Revision G (12/2007)

Updated silicon issue 5 (PSV Operations Using SR), and added silicon issues 25 and 26 (I^2C), and 27 (I/O Port – Port Pin Multiplexed with IC1).

Revision H (5/2008)

Added silicon issues 28 and 29 (I^2C), and 30 (Timer).

Revision J (9/2008)

Replaced issues 25 and 28 (I^2C) with issue 33 (I^2C). Added silicon issues 29 (PLL Lock Status Bit), 30 (PSV Operations) and 31-33 (I^2C).

© 2008 Microchip Technology Inc.

NOTES:

© 2008 Microchip Technology Inc.

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.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE**.** Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, rfPIC, SmartShunt and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, PICkit, PICDEM, PICDEM.net, PICtail, PIC32 logo, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, Select Mode, Total Endurance, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2008, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.

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

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

© 2008 Microchip Technology Inc.