dsPIC30F6011/ 6012/6013/6014

dsPIC30F6011/6012/6013/6014 Rev. B1 Silicon Errata

dsPIC30F6011, dsPIC30F6012, dsPIC30F6013, dsPIC30F6014 (Rev. B1) Silicon Errata

The dsPIC30F6011/6012/6013/6014 (Rev. B1) samples you have received were found to conform to the specifications and functionality described in the following documents:

- DS70030 - dsPIC30F Programmer's Reference Manual
- DS70117 - dsPIC30F6011, dsPIC30F6012, dsPIC30F6013, dsPIC30F6014 Data Sheet
- DS70046 - dsPIC30F Family Reference Manual

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

- dsPIC30F6011
- dsPIC30F6012
- dsPIC30F6013
- dsPIC30F6014

The errata in this document apply to all devices marked with year and week codes, "04" and "06", respectively, or later.

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

Silicon Errata Summary

The following list summarizes the errata described in further detail through the remainder of 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. Decimal Adjust Instruction

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

4. 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.

5. 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.

6. 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.

7. Y-Space Data 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.

8. IPC2 SFR Write Sequence

   A specific write sequence for IPC2 (Interrupt Priority Control 2) SFR is required.

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. 32-bit General Purpose Timers

    The 32-bit General Purpose Timers do not function as specified for prescaler ratios other than 1:1.

12. 12-bit 100 Ksps A/D Converter

    The 12-bit A/D converter 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.

13. 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.

14. DCI –Stop in Idle mode

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

15. CAN SFR Reads

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

16. High IDD During Row Erase of Program Flash Memory

    This release of silicon exhibits a current draw (IDD) of approximately 370 mA during a Row Erase operation performed on Program Flash memory.

17. 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 within 5% of 5 volts.

18. dsPIC30F6011/6013 Code Protection

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

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

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 (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: 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:

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

4. 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.

TABLE 1:

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

Note 1: 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.

2: Refer to the Programmer's Reference Manual for details on the dsPIC30F Instruction set.

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:

```
.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!
```

Work around

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.

EXAMPLE 3:

```
.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 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.

5. 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:

```
        .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.
```

6. 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:

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

7. Module: Y-Space Data 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:

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

Work around

Work around 1:

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

EXAMPLE 7:

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

8. Module: Interrupt Controller

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

Work around

An example of this write sequence is shown in Example 8.

EXAMPLE 8:

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

When coding in C, the write sequence shown above can be implemented using inline assembly instructions. The equivalent write sequence using the C30 compiler is shown in Example 9.

EXAMPLE 9:

```
asm volatile(   "mov.d  w0, [w15++]\n\t"
                "mov    #IPC2,w0\n\t"
                "mov    #0x4444,w1\n\t"
                "disi   #2\n\t"
                "mov    w1, IPC2\n\t"
                "mov    #IPC2, w0\n\t"
                "mov.d  [--w15], w0");
//Note: There are no commas between
//      the quoted strings in the code
//      segment above.
```

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 10.

EXAMPLE 10:

```
.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 11 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 11:

```
__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 12. This will disable all interrupts between priority levels 1 through 7.

EXAMPLE 12:

```
__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
```

11. 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.

12. Module: 12-bit 100 Ksps A/D Converter

Input Channel Scanning allows the A/D converter to acquire and convert signals on a selected set of "MUX A" input pins in sequence. This function is controlled by the CSCNA (ADCON2<11>) bit and the ADCSSL SFR.

The ALTS (ADCON2<0>) bit, when set, allows the A/D converter 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 A/D 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 A/D converter 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 A/D converter.

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 A/D converter 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

13. 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(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 LS Bit of the original data word to be transmitted is loaded into the additionally enabled bit of the transmit buffer register, TXBUFn, or the MS bit 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:

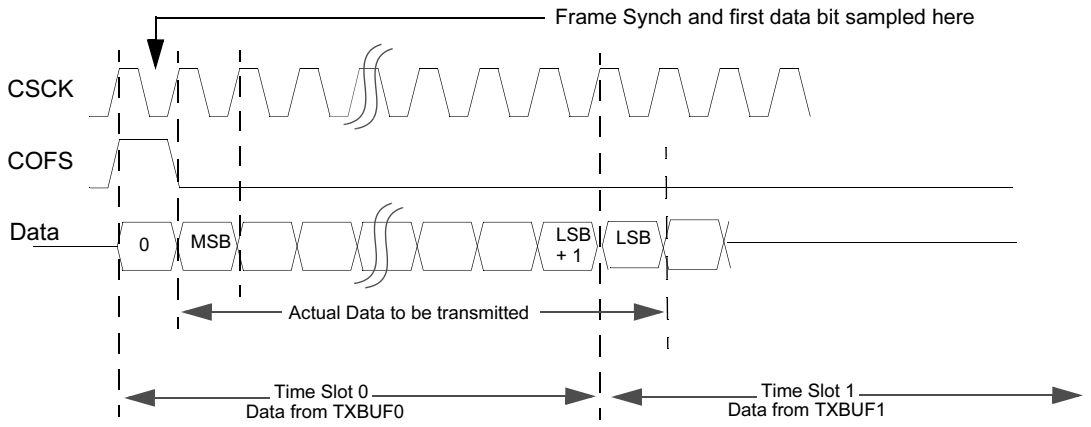
- DCICON1 = 0x0720, DCICON2 = 0x0DEF DCICON3 = 0x0000, TSCON = RSCON = 0x0003

An example of loading the DCI Transmit buffers for the configuration above is shown in Example 13. 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.

EXAMPLE 13:

```
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 MS bit of time slot 0 is '0'. While the MS bit of time slot 1 is actually the LS bit of the data intended for transmission.

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

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

14. 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.

15. Module: CAN – Read Operations on SFRs

Data read from the CAN module Special Function Registers (SFR) may not be correct at device operation 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).

If the dsPIC device needs to operate at a throughput higher than 20 MIPS, the user should incorporate the suggested workarounds 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 lesser.

Work around

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 14. 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 14:

```
.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.

16. Module: High IDD During Row Erase of Program Flash Memory

This release of silicon draws a current (IDD) of approximately 370 mA during any Row Erase operation performed on Program Flash memory.

Work around

Work around 1:

Supply the VDD 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 VDD greater than or equal to 4.2V, then connect a 1000 µF Electrolytic capacitor across the VDD 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 the device is operating at less than 10 MIPS, the application may post-scale the system clock or switch to the Internal FRC oscillator.

17. 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 1: Applications that use the CAN peripherals and Data EEPROM should also refer to Errata 1. and 15.

18. 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.

APPENDIX A: REVISION HISTORY

Revision A (2/2004)

Original version of the document.

Revision B (4/2004)

Document updated from “Confidential” to “Advance Information”.

Revision C (5/2004)

Added errata #3, #12, #13, #14 and #15.

Revision D (11/2004)

Added errata #4, #5 and #18.

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

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

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

Trademarks

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

Microchip received ISO/TS-16949:2002 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona and Mountain View, California in October 2003. The Company's quality system processes and procedures are for its PICmicro® 8-bit MCUs, 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
Alpharetta, GA
Tel: 770-640-0034
Fax: 770-640-0307

Boston
Westford, MA
Tel: 978-692-3848
Fax: 978-692-3821

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

San Jose
Mountain View, CA
Tel: 650-215-1444
Fax: 650-961-0286

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

ASIA/PACIFIC

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: 86-28-8676-6200
Fax: 86-28-8676-6599

China - Fuzhou
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

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

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 - Shunde
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

China - Qingdao
Tel: 86-532-502-7355
Fax: 86-532-502-7205

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-2229-0061
Fax: 91-80-2229-0062

India - New Delhi
Tel: 91-11-5160-8632
Fax: 91-11-5160-8632

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

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

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

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

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

Taiwan - Hsinchu
Tel: 886-3-572-9526
Fax: 886-3-572-6459

EUROPE

Austria - Weis
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

Denmark - Ballerup
Tel: 45-4420-9895
Fax: 45-4420-9910

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

Germany - Ismaning
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

England - Berkshire
Tel: 44-118-921-5869
Fax: 44-118-921-5820

09/27/04