dsPIC30F5011/5013 Rev. A1/A2 Silicon Errata

dsPIC30F5011/5013 (Rev. A1/A2) Silicon Errata

The dsPIC30F5011/5013 (Rev. A1/A2) samples you have received were found to conform to the specifications and functionality described in the following documents:

- DS70030 – dsPIC30F 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. The specific devices for which these exceptions are described are 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 = mg3 a1
...Reading ICD Product ID
Running ICD Self Test
...Passed
MPLAB ICD 2 Ready
```

Depending on the revision of silicon, the text in bold would either state "a1" or "a2".

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 further detail throughout the remainder of this document:

1. Data EEPROM
   The Most Significant bit of every 4th byte in data EEPROM may be corrupted.
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^{2}C™ – Read Operations on I2CCON SFR
   Read operations performed on the I2CCON SFR, may yield incorrect results at operation over 20 MIPS.
6. I^{2}C – 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. High IDD During Row Erase of Program Flash Memory
   This release of silicon exhibits a current draw (IDD) of approximately 190 mA during a Row Erase operation performed on program Flash memory.
10. x4 PLL Operation
    The x4 PLL mode of operation may not function correctly for certain input frequencies.
11. Sequential Interrupts
    Sequential interrupts after modifying the CPU IPL, interrupt IPL, interrupt enable or interrupt flag may cause an Address Error trap.
12. Using OSC2/RC15 pin for Digital I/O
    For this revision of silicon, if the pin RC15 is required for digital input/output, the FPR<4:0> bits in the FOSC Configuration register may not be set up for FRC w/PLL 4x/8x/16x modes.

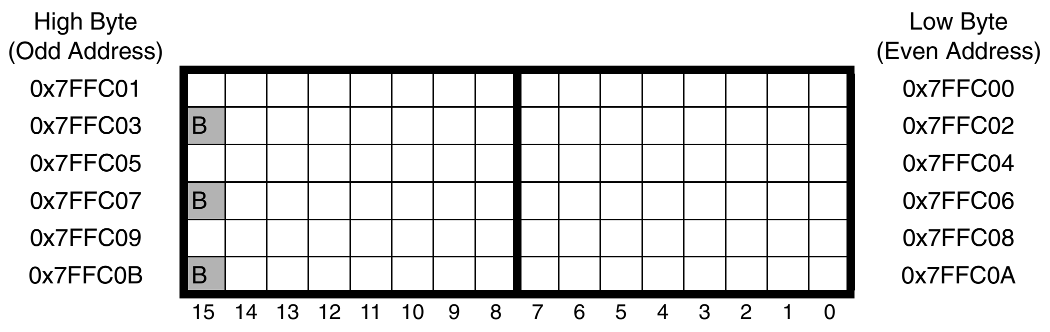
The following sections will describe the errata and work around to these errata, where they may apply.

1. Module: Data EEPROM

The Most Significant bit of every fourth byte in data EEPROM may be corrupted on any write operation. This write corruption may occur while using either PRO MATE®, MPLAB ICD 2 or Run-Time Self-Programming (RTSP).

Figure 1 shows the first twelve bytes in data EEPROM and indicates the affected bits.

FIGURE 1: dsPIC30F5011/5013 DATA EEPROM



Note 1: The shaded bits labelled "B" represent the bits that may be corrupted on a write operation.

2: The memory map shown here depicts only the first twelve bytes of device EEPROM.

Work around

Work Around 1:

Use program Flash memory instead of data EEPROM to store constant data.

Work Around 2:

Use less than 16 bits in each word in the available data EEPROM, excluding the Most Significant bit.

Work Around 3:

Avoid using every fourth byte. Example 1 shows how the ASM30 assembler can be used to allocate data in the EEPROM under this constraint.

EXAMPLE 1:

```
.include    "p30f5013.inc"
.section    .eedata, "r"
.align  4
.hword  0xF345
.byte   0x23
.byte   0xFF            ;Unused byte
.hword  0x1234
.byte   0x23
.byte   0xFF            ;Unused byte"
```

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 2 shows how the application should process the Carry bit during a BCD addition operation.

EXAMPLE 2:

```
    .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 3 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 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
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 3 is demonstrated in Example 4.

EXAMPLE 4:

```
.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.

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 5.

EXAMPLE 5:

```
       .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^{2}C – 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 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® 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^{2}C software functions from Microchip's dsPIC30F Peripheral Library, should operate the device at 20 MIPS or less.

Work around

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 6. 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 6:

```
    .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^{2}C – 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 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^{2}C software functions from Microchip's dsPIC30F Peripheral Library, should operate the device at 20 MIPS or less.

Work around

Work Around 1: For Assembly Language Source Code

When writing to the I2CTRN 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:

```
.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 (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 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 around

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 8. 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 8:

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

This release of silicon draws a current (I_{DD}) of approximately 190 mA during any Row Erase operation performed on program Flash memory.

Work around

Work Around 1:

Supply the V_{DD} pin using a voltage regulator capable of sourcing a minimum of 190 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 the device is operating at less than 10 MIPS, the application may postscale the system clock or switch to the internal FRC oscillator.

10. Module: PLL

When the x4 PLL mode of operation is selected, the specified input frequency range of 4-10 MHz is not fully supported.

When device V_{DD} is 2.5-3.0V, the x4 PLL input frequency must be in the range of 4-5 MHz. When device V_{DD} is 3.0-3.6V, the x4 PLL input frequency must be in the range of 4-6 MHz for both industrial and extended temperature ranges.

Work around

1. Use x8 PLL or x16 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 x4 PLL clock rate.

11. 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 with priority higher than Interrupt 1 occurs.

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 9. 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 10. A macro may also be used to perform this task, as shown in Example 11.

EXAMPLE 9:

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

EXAMPLE 10:

```
.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 11:

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

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

12. Module: Using OSC2/RC15 pin for Digital I/O

The port pin, RC15, is multiplexed with the primary oscillator pin, OSC2. When pin RC15 is required for digital input/output, specific bits in the Oscillator Configuration register, FOSC, may be set up as follows:

- FOS<2:0> (FOSC<10:8>) bits configured for LP, LPRC, FRC, ECIO, ERCIO or ECIO w/PLL 4x/8x/16x
- FPR<4:0> (FOSC<4:0>) bits may be configured for ECIO w/PLL 4x/8x/16x

For this revision of silicon, if the RC15 digital I/O port function is desired, the FPR<4:0> bits in the FOSC Configuration register may not be set up for FRC w/PLL 4x/8x/16x modes.

Work around

None. In future revisions of silicon, port pin RC15 may also be configured for digital I/O when the FPR<4:0> bits in the FOSC Configuration register are set up for FRC w/PLL 4x/8x/16x modes.

APPENDIX A: REVISION HISTORY

Revision A (9/2004)

Original version of the document.

Revision B (11/2004)

Added silicon issues 3 (PSV Operations Using SR) and 4 (Early Termination of Nested DO Loops).

Revision C (12/2004)

Added silicon issues 5 (I^{2}C – Read Operations on I2CCON SFR), 6 (I^{2}C – Write Operations on I2CTRN SFR) and 7 (UART – Write Operations on U1MODE and U2MODE SFRs).

Revision D (3/2005)

Added silicon issues 8 (Data Converter Interface – Idle), 9 (High I_{DD} During Row Erase of Program Flash Memory), 10 (PLL) and 11 (Interrupt Controller – Sequential Interrupts).

Revision E (4/2005)

Added silicon issue 12 (Using OSC2/RC15 pin for Digital I/O).

NOTES:

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
Alpharetta, GA
Tel: 770-640-0034
Fax: 770-640-0307

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

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: 91-11-5160-8631
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: 45-4450-2828
Fax: 45-4485-2829

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

03/01/05