

## PIC16(L)F1938/1939 Silicon Errata and Data Sheet Clarification

The PIC16(L)F1938/1939 family devices that you have received conform functionally to the current Device Data Sheet (DS40001574C), except for the anomalies described in this document.

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


The errata described in this document will be addressed in future revisions of the PIC16(L)F1938/1939 silicon.

**Note:** This document summarizes all silicon errata issues from all revisions of silicon, previous as well as current. Only the issues indicated in the last column of [Table 2](#) apply to the current silicon revision (**A3**).

Data Sheet clarifications and corrections start on [page 9](#), following the discussion of silicon issues.

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

For example, to identify the silicon revision level using MPLAB IDE in conjunction with a hardware debugger:

1. Using the appropriate interface, connect the device to the hardware debugger.
2. Open an MPLAB IDE project.
3. Configure the MPLAB IDE project for the appropriate device and hardware debugger.
4. Based on the version of MPLAB IDE you are using, do one of the following:
  - a) For MPLAB IDE 8, select *Programmer > Reconnect*.
  - b) For MPLAB X IDE, select *Window > Dashboard* and click the **Refresh Debug Tool Status** icon (  ).
5. Depending on the development tool used, the part number *and* Device Revision ID value appear in the **Output** window.

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

The DEVREV values for the various PIC16(L)F1938/1939 silicon revisions are shown in [Table 1](#).

**TABLE 1: SILICON DEVREV VALUES**

Part Number	DEVICE ID<13:0> <sup>(1,2)</sup>			
	DEV<8:0>	Revision ID for Silicon Revision		
		A1	A2	A3
PIC16F1938	10 0011 101	0 0001	0 0010	0 0011
PIC16F1939	10 0011 110	0 0001	0 0010	0 0011
PIC16LF1938	10 0100 101	0 0001	0 0010	0 0011
PIC16LF1939	10 0100 110	0 0001	0 0010	0 0011

**Note 1:** The Device ID is located in the configuration memory at address 8006h.

**2:** Refer to the “PIC16F193X/LF193X/PIC16F194X/LF194X/PIC16LF190X Memory Programming Specification” (DS41397) for detailed information on Device and Revision IDs for your specific device.

# PIC16(L)F1938/1939

**TABLE 2: SILICON ISSUE SUMMARY**

Module	Feature	Item Number	Issue Summary	Affected Revisions <sup>(1)</sup>		
				A1	A2	A3
ADC	ADC Conversion	1.1	ADC Conversion may not Complete.	X		
Enhanced Capture Compare PWM (ECCP)	Enhanced PWM	2.1	PWM 0% Duty Cycle Direction Change.	X		
Enhanced Capture Compare PWM (ECCP)	Enhanced PWM	2.2	PWM 0% Duty Cycle Port Steering.	X		
Timer	Timer1 Gate Toggle mode	3.1	T1 Gate flip-flop does not Clear.	X		
In-Circuit Serial Programming™ (ICSP™)	Low-Voltage Programming	4.1	Bulk Erase not Available with LVP.	X		
Oscillator	Clock Switching	5.1	Clock switching can cause a single corrupted instruction.	X	X	
Oscillator	Oscillator Start-up Timer (OSTS) bit	5.2	OSTS bit remains set.	X	X	
Oscillator	Oscillator Start-up Timer (OSTS) bit	5.3	OSTS bit remains clear.	X	X	X
Enhanced Universal Synchronous Asynchronous Receiver (EUSART)	Auto-Baud Detect	6.1	Auto-Baud Detect may store incorrect count value in the SPBRG registers.	X	X	
BOR	Wake-up from Sleep	7.1	Device resets on wake-up from Sleep (PIC16LF1938/1939 devices only).	X	X	
MSSP (Master Synchronous Serial Port)	SPI Master mode	8.1	Buffer Full (BF) bit or MSSP Interrupt Flag (SSPIF) bit becomes set half SCK cycle too early.	X	X	X

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

## Silicon Errata Issues

**Note:** This document summarizes all silicon errata issues from all revisions of silicon, previous as well as current. Only the issues indicated by the shaded column in the following tables apply to the current silicon revision (**A3**).

### 1. Module: ADC

#### 1.1 Analog-to-Digital Conversion

An ADC conversion may not complete under these conditions:

1. When FOSC is greater than 8 MHz and it is the clock source used for the ADC converter.
2. The ADC is operating from its dedicated internal FRC oscillator and the device is not in Sleep mode (any FOSC frequency).

When this occurs, the ADC Interrupt Flag (ADIF) does not get set, the GO/DONE bit does not get cleared, and the conversion result does not get loaded into the ADRESH and ADRESL result registers.

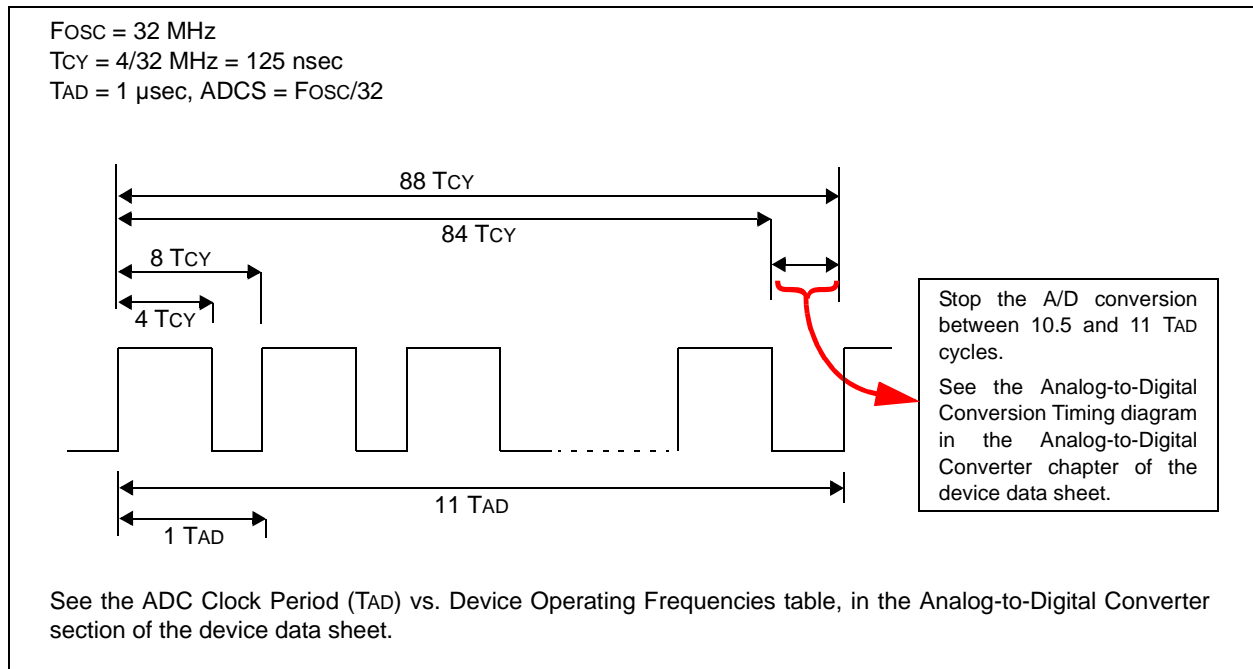
### Work around

Method 1: Select the system clock, FOSC, as the ADC clock source and reduce the FOSC frequency to 8 MHz or less when performing ADC conversions.

Method 2: Select the dedicated FRC oscillator as the ADC conversion clock source and perform all conversions with the device in Sleep.

Method 3: This method is provided if the application cannot use Sleep mode and requires continuous operation at frequencies above 8 MHz. This method requires early termination of an ADC conversion. Provide a fixed time delay in software to stop the A-to-D conversion manually, after all 10 bits are converted, but before the conversion would complete automatically. The conversion is stopped by clearing the GO/DONE bit in software. The GO/DONE bit must be cleared during the last 1/2 TAD cycle, before the conversion would have completed automatically. Refer to [Figure 1](#) for details.

**FIGURE 1: INSTRUCTION CYCLE DELAY CALCULATION EXAMPLE**



# PIC16(L)F1938/1939

In [Figure 1](#), 88 instruction cycles (T<sub>cy</sub>) will be required to complete the full conversion. Each TAD cycle consists of 8 T<sub>cy</sub> periods. A fixed delay is provided to stop the A/D conversion after 86 instruction cycles and terminate the conversion at the correct time as shown in the figure above.

**Note:** The exact delay time will depend on the TAD divisor (ADCS) selection. The T<sub>cy</sub> counts shown in the timing diagram above apply to this example only. Refer to [Table 3](#) for the required delay counts for other configurations.

## EXAMPLE 1: CODE EXAMPLE OF INSTRUCTION CYCLE DELAY

```
BSF    ADCON0, ADGO    ; Start ADC conversion
                        ; Provide 86
                        ; instruction cycle
                        ; delay here
BCF    ADCON0, ADGO    ; Terminate the
                        ; conversion manually
MOVF   ADRESH, W      ; Read conversion
                        ; result
```

For other combinations of FOSC, TAD values and Instruction cycle delay counts, refer to [Table 3](#).

**TABLE 3: INSTRUCTION CYCLE DELAY COUNTS BY TAD SELECTION**

TAD	Instruction Cycle Delay Counts
Fosc/64	172
Fosc/32	86
Fosc/16	43

### Affected Silicon Revisions

A1	A2	A3					
X							

## 2. Module: Enhanced Capture Compare PWM (ECCP)

### 2.1 Enhanced PWM

When the PWM is configured for Full-Bridge mode and the duty cycle is set to 0%, writing the P<sub>xM</sub><1:0> bits to change the direction has no effect on P<sub>x</sub>A and P<sub>x</sub>C outputs.

#### Work around

Increase the duty cycle to a value greater than 0% before changing directions.

#### Affected Silicon Revisions

A1	A2	A3					
X							

### 2.2 Enhanced PWM

In PWM mode, when the duty cycle is set to 0% and the STR<sub>x</sub>SYNC bit is set, writing the STR<sub>x</sub>A, STR<sub>x</sub>B, STR<sub>x</sub>C and the STR<sub>x</sub>D bits to enable/disable steering to port pins has no effect on the outputs.

#### Work around

Increase the duty cycle to a value greater than 0% before enabling/disabling steering to port pins.

#### Affected Silicon Revisions

A1	A2	A3					
X							

## 3. Module: Timer

### 3.1 Timer1 Gate Toggle mode

When Timer1 Gate Toggle mode is enabled, it is possible to measure the full-cycle length of a Timer1 gate signal. To perform this function, the Timer1 gate source is routed through a flip-flop that changes state on every incrementing edge of the gate signal. Timer1 Gate Toggle mode is enabled by setting the T1GTM bit of the T1GCON register. When working properly, clearing either the T1GTM bit or the TMR1ON bit would also clear the output value of this flip-flop, and hold it clear. This is done in order to control which edge is being measured. The issue that exists is that clearing the TMR1ON bit does not clear the output value of the flip-flop and hold it clear.

#### Work around

Clear the T1GTM bit in the T1GCON register to clear and hold clear the output value of the flip-flop.

#### Affected Silicon Revisions

A1	A2	A3						
X								

## 4. Module: In-Circuit Serial Programming™ (ICSP™)

### 4.1 Bulk Erase Feature not available with Low-Voltage Programming mode

A bulk erase of the program Flash memory or data memory cannot be executed in Low-Voltage Programming mode.

#### Work around

Method 1: If ICSP Low-Voltage Programming mode is required, use row erases to erase the program memory, as described in the Program/Verify mode section of the Programming Specification. Data memory must be over-written with the desired values.

Method 2: Use ICSP High-Voltage Programming mode if a bulk erase is required.

**Note:** Only the bulk erase feature will erase program or data memory if code or data protection is enabled. Method 2 must be used if code or data protection is enabled.

#### Affected Silicon Revisions

A1	A2	A3						
X								

## 5. Module: Oscillator

### 5.1 Clock Switching

When switching clock sources between INTOSC clock source and an external clock source, one corrupted instruction may be executed after the switch occurs.

This issue does not affect Two-Speed Start-up or the Fail-Safe Clock Monitor operation.

#### Work around

When switching from an external oscillator clock source, first switch to 16 MHz HFINTOSC. Once running at 16 MHz HFINTOSC, configure IRCF to run at desired internal oscillator frequency.

When switching from an internal oscillator (INTOSC) to an external oscillator clock source, first switch to HFINTOSC High-Power mode (8 MHz or 16 MHz). Once running from HFINTOSC, switch to the external oscillator clock source.

#### Affected Silicon Revisions

A1	A2	A3						
X	X							

### 5.2 Oscillator Start-up Timer (OSTS) bit

During the Two-Speed Start-up sequence, the OST is enabled to count 1024 clock cycles. After the count is reached, the OSTS bit is set, the system clock is held low until the next falling edge of the external crystal (LP, XT or HS mode), before switching to the external clock source.

When an external oscillator is configured as the primary clock and Fail-Safe Clock mode is enabled (FCMEN = 1), any of the following conditions will result in the Oscillator Start-up Timer (OST) failing to restart:

- MCLR Reset
- Wake from Sleep
- Clock change from INTOSC to Primary Clock

This anomaly will manifest itself as a clock failure condition for external oscillators which take longer than the clock failure time-out period to start.

#### Work around

None.

#### Affected Silicon Revisions

A1	A2	A3						
X	X							

# PIC16(L)F1938/1939

## 5.3 Oscillator Start-Up Timer (OSTS) bit

When using an external crystal in HS mode and the 4xPLL is enabled, once the Two-Speed Start-up sequence has expired, the OSTS bit remains clear incorrectly indicating that the microcontroller is operating from the internal oscillator. This issue only occurs when the 4xPLL is enabled; when the 4xPLL is disabled, the OSTS bit operates as expected.

### Work around

Use the PLL ready flag (PLLRF bit of the OSCSTAT register) to determine that the microcontroller is operating from the external crystal. The PLLRF bit will only be set when the clock source used by the PLL is ready; therefore, the PLLRF bit will indicate when both the external crystal and the PLL are ready.

### Affected Silicon Revisions

A1	A2	A3					
X	X	X					

## 6. Module: Enhanced Universal Synchronous Asynchronous Receiver (EUSART)

### 6.1 Auto-Baud Detect

When using automatic baud detection (ABDEN), on occasion, an incorrect count value can be stored at the end of auto-baud detection in the SPBRGH:SPBRGL (SPBRG) registers. The SPBRG value may be off by several counts. This condition happens sporadically when the device clock frequency drifts to a frequency where the SPBRG value oscillates between two different values. The issue is present regardless of the baud rate Configuration bit settings.

### Work around

When using auto-baud, it is a good practice to always verify the obtained value of SPBRG, to ensure it remains within the application specifications. Two recommended methods are shown below.

For additional auto-baud information, see Technical Brief TB3069, "Use of Auto-Baud for Reception of LIN Serial Communications Devices: Mid-Range and Enhanced Mid-Range".

## EXAMPLE 2: METHOD 1 – EUSART AUTO-BAUD DETECT WORK AROUND

```
#define SPBRG_16BIT    *((*int)&SPBRG;           // define location for 16-bit SPBRG value

const int DEFAULT_BAUD = 0x0067;              // Default Auto-Baud value
const int TOL = 0x05;                          // Baud Rate % tolerance
const int MIN_BAUD = DEFAULT_BAUD - TOL;       // Minimum Auto-Baud Limit
const int MAX_BAUD = DEFAULT_BAUD + TOL;       // Maximum Auto-Baud Limit
.
.
.
ABDEN = 1;                                     // Start Auto-Baud
while (ABDEN);                                 // Wait until Auto-Baud completes

if((SPBRG_16BIT > MAX_BAUD) || (SPBRG_16BIT < MIN_BAUD))
{
    SPBRG_16BIT = DEFAULT_BAUD;                // Compare if value is within limits
                                              // if out of spec, use DEFAULT_BAUD
}
.
.
.
                                              // if in spec, continue using the
                                              // Auto-Baud value in SPBRG
```

**Note:** In firmware, define default, minimum and maximum auto-baud (SPBRG) values according to the application requirements. For example, if the application runs at 9600 baud at 16 MHz then, the default SPBRG value would be (assuming 16-bit/Asynchronous mode) 0x67. The minimum and maximum allowed values can be calculated based on the application. In this example, a  $\pm 5\%$  tolerance is required, so tolerance is  $0x67 * 5\% = 0x05$ .

## EXAMPLE 3: METHOD 2 – EUSART AUTO-BAUD DETECT WORK AROUND

```

#define SPBRG_16BIT    *((*int)&SPBRG;                // define location for 16-bit SPBRG value

const int DEFAULT_BAUD = 0x0067;                    // Default Auto-Baud value
const int TOL = 0x05;                               // Baud Rate % tolerance
const int MIN_BAUD = DEFAULT_BAUD - TOL;            // Minimum Auto-Baud Limit
const int MAX_BAUD = DEFAULT_BAUD + TOL;            // Maximum Auto-Baud Limit

int Average_Baud;                                    // Define Average_Baud variable
int Integrator;                                     // Define Integrator variable
.
.
.
Average_Baud = DEFAULT_BAUD;                        // Set initial average Baud rate
Integrator = DEFAULT_BAUD*15;                       // The running 16 count average
.
.
.
ABDEN = 1;                                          // Start Auto-Baud
while (ABDEN);                                     // Wait until Auto-Baud completes

Integrator+ = SPBRG_16BIT;
Average_Baud = Integrator/16;
if((SPBRG_16BIT > MAX_BAUD)|| (SPBRG_16BIT < MIN_BAUD))
{
    SPBRG_16BIT = Average_Baud;                    // Check if value is within limits
                                                    // If out of spec, use previous average
}
else
{
    Integrator+ = SPBRG_16BIT;                      // If in spec, calculate the running
    Average_Baud = Integrator/16;                  // average but continue using the
    Integrator- = Average_Baud;                    // Auto-Baud value in SPBRG
}
.
.
.

```

**Note:** Similar to Method 1, define default, minimum and maximum auto-baud (SPBRG) values. In firmware, compute a running average of SPBRG. If the new SPBRG value falls outside the minimum or maximum limits, then use the current running average value (Average\_Baud), otherwise use the auto-baud SPBRG value and calculate a new running average. For example, if the application runs at 9600 baud at 16 MHz then, the default SPBRG value would be (assuming 16-bit/Asynchronous mode) 0x67. The minimum and maximum allowed values can be calculated based on the application. In this example, a  $\pm 5\%$  tolerance is required, so tolerance is  $0x67 \times 5\% = 0x05$ .

### Affected Silicon Revisions

A1	A2	A3						
X	X							

# PIC16(L)F1938/1939

## 7. Module: BOR

### 7.1 BOR Reset

This issue affects only the PIC16LF1938/1939 devices. These devices may undergo a BOR Reset when waking-up from Sleep and BOR is re-enabled. A BOR Reset may also occur the moment the software BOR is enabled.

Under certain voltage and temperature conditions and when either SBODEN or BOR\_NSLEEP is selected, the devices may occasionally reset when waking-up from Sleep or BOR is enabled.

#### Work around

- Method 1: In applications where BOR use is not critical, turn off the BOR in the Configuration Word.
- Method 2: Set the FVREN bit of the FVRCON register. Maintain this bit on at all times.
- Method 3: When BOR module is needed only during run-time, use the software-enabled BOR by setting the SBODEN option on the Configuration Word. BOR should be turned off by software before Sleep, then follow the below sequence for turning BOR on after wake-up:
- Wake-up event occurs;
  - Turn on FVR (FVREN bit of the FVRCON register);
  - Wait until FVRRDY bit is set;
  - Wait 15  $\mu$ s after the FVR Ready bit is set;
  - Manually turn on the BOR.
- Method 4: Use the software-enabled BOR as described in Method 3, but use the following sequence:
- Switch to internal 32 kHz oscillator immediately before Sleep;
  - Upon wake-up, turn on FVR (FVREN bit of the FVRCON register);
  - Manually turn on the BOR;
  - Switch the clock back to the preferred clock source.

**Note:** When using the software BOR follow the steps in Methods 3 or 4 above when enabling BOR for the first time during program execution.

#### Affected Silicon Revisions

A1	A2	A3					
X	X						

## 8. Module: MSSP (Master Synchronous Serial Port)

### 8.1 SPI Master mode

When the MSSP is used in SPI Master mode and the CKE bit is clear (CKE = 0), the Buffer Full (BF) bit and the MSSP Interrupt Flag (SSPIF) bit becomes set half an SCK cycle early. If the user software immediately reacts to either of the bits being set, a write collision may occur as indicated by the WCOL bit being set.

#### Work around

To avoid a write collision one of the following methods should be used:

- Method 1: Add a software delay of one SCK period after detecting the completed transfer (the BF bit or SSPIF bit becomes set) and prior to writing to the SSPBUF register. Verify the WCOL bit is clear after writing to SSPBUF. If the WCOL bit is set, clear the bit in software and rewrite the SSPBUF register.
- Method 2: As part of the MSSP initialization procedure, set the CKE bit (CKE = 1).

#### Affected Silicon Revisions

A1	A2	A3					
X	X	X					



## Data Sheet Clarifications

The following typographic corrections and clarifications are to be noted for the latest version of the device data sheet (DS40001574C):

<b>Note:</b> Corrections are shown in <b>bold</b> . Where possible, the original bold text formatting has been removed for clarity.
-------------------------------------------------------------------------------------------------------------------------------------

None.

# PIC16(L)F1938/1939

---

## APPENDIX A: DOCUMENT REVISION HISTORY

### **Rev A Document (05/2010)**

Initial release of this document.

### **Rev B Document (07/2010)**

Revised Module 1.1; Added Modules 3.2 and 4.1;  
Other minor corrections.

### **Rev C Document (12/2010)**

Updated errata to new format; Added Revision A2.

### **Rev D Document (02/2012)**

Updated Table 1; Added Module 5 and 6; Other minor  
corrections.

Data Sheet Clarifications: Deleted Modules 1 and 2.

### **Rev E Document (01/2013)**

Added MPLAB X IDE; Added Silicon Revision A3;  
Added module 7, BOR; Other minor corrections.

### **Rev F Document (12/2014)**

Added module 8, MSSP; Other minor corrections.

### **Rev G Document (12/2015)**

Added module 5.4. Removed errata items for Timer0  
Gate Source and HFINTOSC Ready/Stable bit that had  
been erroneously added to the document in a previous  
version.

---

**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 unless otherwise stated.

#### **Trademarks**

The Microchip name and logo, the Microchip logo, dsPIC, FlashFlex, flexPWR, JukeBlox, KEELOQ, KEELOQ logo, Klear, LANCheck, MediaLB, MOST, MOST logo, MPLAB, OptoLyzer, PIC, PICSTART, PIC<sup>32</sup> logo, RightTouch, SpyNIC, SST, SST Logo, SuperFlash and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

The Embedded Control Solutions Company and mTouch are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, BodyCom, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, ECAN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, KlearNet, KlearNet logo, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICKIT, PICtail, RightTouch logo, REAL ICE, SQI, Serial Quad I/O, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, 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.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

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

© 2010-2015, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-0094-3

**QUALITY MANAGEMENT SYSTEM**  
**CERTIFIED BY DNV**  
**== ISO/TS 16949 ==**

*Microchip received ISO/TS-16949:2009 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.*