dsPIC33AK512MPS512 Family Silicon Errata and Data Sheet Clarification

The dsPIC33AK512MPS512 family devices that you have received conform functionally to the current device data sheet (DS70005591**C**), 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 dsPIC33AK512MPS512 family 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 (**A2**).

Data sheet clarifications and corrections start on Page 10, 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 website (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 silicon revisions of the dsPIC33AK512MPS512 family are shown in Table 1.

TABLE 1: SILICON DEVREV VALUES

Part Number	Device ID(1)	Revision ID for Silicon Revision(1)		Part Number	Device ID(1)	Revision ID for Silicon Revision(1)	
		A1	A2			A1	A2
dsPIC33AK512MPS512	0xA77C	01h	02h	dsPIC33AK256MPS206	0xA719	01h	02h
dsPIC33AK512MPS510	0xA77B			dsPIC33AK256MPS205	0xA718		
dsPIC33AK512MPS508	0xA77A			dsPIC33AK512MC510	0xA763		
dsPIC33AK512MPS506	0xA779			dsPIC33AK512MC508	0xA762		
dsPIC33AK512MPS505	0xA778			dsPIC33AK512MC506	0xA761		
dsPIC33AK256MPS512	0xA75C			dsPIC33AK512MC505	0xA760		
dsPIC33AK256MPS510	0xA75B			dsPIC33AK256MC510	0xA743		
dsPIC33AK256MPS508	0xA75A			dsPIC33AK256MC508	0xA742		
dsPIC33AK256MPS506	0xA759			dsPIC33AK256MC506	0xA741		
dsPIC33AK256MPS505	0xA758			dsPIC33AK256MC505	0xA740		
dsPIC33AK512MPS212	0xA73C			dsPIC33AK512MC210	0xA723		
dsPIC33AK512MPS210	0xA73B			dsPIC33AK512MC208	0xA722		
dsPIC33AK512MPS208	0xA73A			dsPIC33AK512MC206	0xA721		
dsPIC33AK512MPS206	0xA739			dsPIC33AK512MC205	0xA720		
dsPIC33AK512MPS205	0xA738			dsPIC33AK256MC210	0xA703		
dsPIC33AK256MPS212	0xA71C			dsPIC33AK256MC208	0xA702		
dsPIC33AK256MPS210	0xA71B			dsPIC33AK256MC206	0xA701		
dsPIC33AK256MPS208	0xA71A			dsPIC33AK256MC205	0xA700		

Note 1: Refer to the *"dsPIC33AK512MPS512 Family Flash Programming Specification"* for detailed information on Device and Revision IDs for your specific device.

TABLE 2: SILICON ISSUE SUMMARY

Module	Feature	Item Number	Issue Summary	Affected Revisions(1)	
				A1	A2
CPU	PCTRAP	1.	PCTRAP captures the trap origination address of a math error trap even when disabled.	X	X
DMA	Bus Read Error	2.	DMA Bus Read Error Fault bit not set.	X	X
PWM	PCI	3.	PCI active is not terminated in latched edge detection modes.	X	X
PWM	Complementary Mode	4.	Software Override priority is not maintained over current limit PCI (CLPCI).	X	X
PWM	Complementary Mode	5.	Dead time is not applied properly when the PCI active signal is deasserted for a short time of 1-2 PWM clock cycles.	X	X
PWM	PCI	6.	PCI SR latch does not reset in RESET Dominant mode.	X	X
PWM	EOC	7.	In Retriggerable mode, EOC event is not blocked when the retrigger event is close to EOC.	X	X
PWM	LEB	8.	LEB active signal asserts for 2 clocks less than programmed.	X	X
PWM	Time Base Capture	9.	Manual time base capture fails after first capture event.	X	X
PWM	TRIGx	10.	When TRIGx = EOC and CAHALF is changed by software, a trigger event can occur in the opposite phase.	X	X
PWM	PCI	11.	In Complementary Output mode, when PSYNC is enabled in PCI Edge Detect mode, PCI override can be applied twice.	X	X
ITC	CVD Array	12.	Make-before-break issue with CVD/cap array.	X	X
ITC	RX Guard Pins	13.	Guard settings in ITCRECx registers control the wrong pins.	X	X
QEI	Index Event	14.	Index event is missed when clock divider (INTDIV[2:0]) is configured for greater than 1.	X	X
ITC	Index Counter	15.	Position counter is loaded with 0x00000001 on an index event for PIMOD[2:0] = 0b001.	X	X
QEI	Buffer Overflow	16.	Address not causing the buffer overflow in Address Detect mode.	X	X
CCP	Output	17.	When ASDGM = 1, the CCP generates an output with shutdown gate input at 0.	X	X
CCP	Capture	18.	When CCSEL= 1 and IGSM= 1, CCP will capture the timer value without an edge.	X	X
PTG	Software Trigger	19.	PTGSWT bit is not cleared in hardware.	X	X
PWM	ADC Trigger Events	20.	In LLC mode, when TRIGy is equal to EOC and with CAPTREN set, ADC triggers are not generated when ADC trigger source is TRIGy.	X	X
DAC	DACOUT Buffers	21.	Both DAC output buffers are non-linear near the supply rails.	X	
CPU	Trap	22.	Address error trap generated on legal indirect+offset move.	X	
GPIO	IO Current Limit	23.	Source/Sink capability of IO pins is less than electrical specifications.	X	
SPI	Ignore Overflow	24.	SPI receiver does not continue to run on overflow with Ignore Overflow bit set.	X	X
SILICON	Silicon Temperature Variants	25.	Silicon only supports Industrial temp (-40°C to 85°C) and not Extended temp (-40°C to 125°C).	X	
CPU	Interrupt Vectoring	26.	Interrupt vectoring and simultaneous PBU/cache invalidation may stall the CPU indefinitely.	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 (**A2**).

1. Module: CPU

In any trap event, the trap origination address register (PCTRAP) is loaded with the value of the Program Counter (PC) associated with the instruction that caused the trap. However, it is not expected to capture the trap origination address for math error traps when they are disabled.

Work around

None. Alternatively, enable math error traps, ignore the PCTRAP value and clear PCTRAP before exiting the math error trap handler. This re-enables the PCTRAP to capture the trap origination address of any other traps.

Affected Silicon Revisions

A1	A2						
X	X						

2. Module: DMA

To initiate a read error trap, it is necessary to activate the Read Error Trap Enable (RETEN) bit. Upon setting the RETEN bit, a read error trap is triggered, which concurrently sets the DMA Bus Read Error Fault bit (DMASTATx.DMAFLT2) to high. Conversely, if the RETEN bit is not enabled, the DMA Bus Read Error Fault bit (DMASTATx.DMAFLT2) will not be set. Therefore, to guarantee the detection of bus read errors in the DMA through the Bus Read Fault bit, RETEN needs to be enabled, although this action may result in the generation of undesired traps.

Work around

Set the RETEN bit explicitly in the application software. When the trap is generated, ignore it and exit from the respective ISR.

Affected Silicon Revisions

A1	A2						
X	X						

3. Module: PWM

PCI Active signal is not terminated immediately when PCI synchronization is enabled, termination synchronization is disabled, and latch is operated in set dominant mode.

Work around

When using PCI Latched modes, do not use PCI source select synchronization (PSYNC) when TERM synchronization is disabled (TSYNCDIS = 1).

Affected Silicon Revisions

A1	A2						
X	X						

4. Module: PWM

The Software Override priority is not maintained over the current limit PCI (CLPCI).

Work around

Use another PCI instance with a higher priority than the current limit, such as the Fault PCI.

Affected Silicon Revisions

A1	A2						
X	X						

5. Module: PWM

Dead time is not applied properly when the PCI active signal is deasserted for a short time of 1-2 PWM clock cycles.

Work around

1. Ensure the PCI active signal does not have a deassertion period of less than 4 PWM clock cycles.
2. Use EOC synchronization or Latched modes to control the timing of PCI deassertion.

Affected Silicon Revisions

A1	A2						
X	X						

6. Module: PWM

When the PCI input is high, the PCI Active signal is terminated even when there is no termination signal.

Work around

None. Only the PCI active signal is affected and the PWM outputs remain halted.

Affected Silicon Revisions

A1	A2						
X	X						

7. Module: PWM

In Retriggerable mode, the EOC event is not blocked when the retrigger event is close to EOC.

Work around

Ensure that the retrigger event is 64 PWM clock cycles before EOC.

Affected Silicon Revisions

A1	A2						
X	X						

8. Module: PWM

The LEB active signal is asserted for 2 clocks less than the programmed value.

Work around

Add 2 to calculated LEB register value before programming.

Affected Silicon Revisions

A1	A2						
X	X						

9. Module: PWM

Manual time base capture fails after the first capture event.

Work around

Use hardware capture for the first capture event. Select the PCI input as the capture source and generate a capture event. Subsequent captures can be triggered in software.

Affected Silicon Revisions

A1	A2						
X	X						

10. Module: PWM

A trigger event is generated at the wrong phase when the trigger value is programmed to be equal to EOC, and the trigger compare event time (CAHALF) is changed on the fly.

Work around

The minimum difference between EOC and trigger values should be at least 64.

Affected Silicon Revisions

A1	A2						
X	X						

11. Module: PWM

In Complementary Output mode, when PSYNC is enabled in PCI Edge Detect mode, PCI override can be applied twice.

Work around

1. Ensure that the PCI active signal does not have a deassert period of less than 4 PWM clock cycles.
2. Use EOC synchronization or Latched modes to control the timing of PCI deassertion.

Affected Silicon Revisions

A1	A2						
X	X						

12. Module: ITC

The charge and discharge switches may be make-before-break and not completely open before the balance switch of the CVD array is closed.

Work around

Use additional sequence steps, where all switches are open, should be added to acquisition sequence.

Affected Silicon Revisions

A1	A2						
X	X						

13. Module: ITC

The guard settings in the ITCRECx registers control the incorrect pins.

Work around

Do not use active guards in the ITCRECx register.

Affected Silicon Revisions

A1	A2						
X	X						

14. Module: QEI

When the QEI is configured with Position Counter Initialization Mode bits (PIMOD[2:0]) equal to 0b001, 0b010, 0b011, 0b100 or 0b111, any index event initializes the position counter. However, if the clock divider value configured by the INTDIV[2:0] bits is greater than 1, and if the index event occurs between two clock edges, the index event will be missed.

Work around

The index pulse width should be sufficiently high enough so that the event is captured by the QEI clock.

Affected Silicon Revisions

A1	A2						
X	X						

15. Module: ITC

When QEI is configured in Quadrature mode (CCM[1:0] = 0b00) with PIMOD[2:0] = 0b001, any index event is supposed to reset the position counter. However, an index event will not reset the position counter; instead, it will be loaded with 0x00000001.

Work around

The user application can enable an interrupt on the index event and load the position counter with zero. The index event should be sufficiently large enough to allow the software to reset the position counter before the next count-up or count-down pulse.

Affected Silicon Revisions

A1	A2						
X	X						

16. Module: QEI

In Address Detect mode, the address byte does not cause the buffer overflow if an address is received when the RX buffer is full.

Work around

Read the data from the RX buffer before it becomes full.

Affected Silicon Revisions

A1	A2						
X	X						

17. Module: CCP

When ASDGM = 1 and shutdown gate input is low (OCFx = 0), the CCP output is still generated.

Work around

Read the auto shutdown fault RP pin through the LAT register bits, and if the LAT value reads as zero, use the CCP Software Shutdown/Gate Control bit to turn off the CCP.

Affected Silicon Revisions

A1	A2						
X	X						

18. Module: CCP

When CCP is configured for input capture and One-Shot mode (IGSM = 1), the CCP will capture the timer value without an edge.

Work around

Once CCP is turned ON, and before enabling gate capture, re-arm the capture module using Input Capture Gate Arm bit.

Affected Silicon Revisions

A1	A2						
X	X						

19. Module: PTG

When executing back-to-back “wait for software trigger” step commands, the PTGSWT bit is not cleared by hardware.

Work around

Manually clear the PTGSWT bit before setting it again between the two PTGSW trigger commands.

Affected Silicon Revisions

A1	A2						
X	X						

20. Module: PWM

In LLC mode, when TRIGy is equal to EOC and with CAPTREN set, ADC triggers are not generated when the ADC trigger source is TRIGy.

Work around

Program initial TRIGy value with 50% of EOC value.

Affected Silicon Revisions

A1	A2						
X	X						

21. Module: DAC

Both DAC output buffers are non-linear near the supply rails. As a result, the DAC's electrical specifications can be greatly affected when using the DAC output buffers.

Work around

Use the UREF module to route DAC1-DAC7 signals to the UREF output pin. The following is an example of how to route the DAC signal through the UREF:

```
// Input to UREF is set to DAC1
UREFCONbits.INSEL = 6;

// Output from UREF is set to UREF pin
UREFCONbits.OUTSEL = 1;

// Turn UREF module on
UREFCONbits.ON = 1;
```

Affected Silicon Revisions

A1	A2						
X							

22. Module: CPU

An address error trap is generated in Indirect Register Offset Addressing mode (represented as [Ws + Wb] within instructions) when the value of Ws (Source) used is beyond 24 bits wide (size of the address path). The issue may occur even when the effective address (Ws + Wb) is a valid address.

Work around

Either of the following workarounds can be used to overcome the issue:

1. Define the value of Ws within the 24-bit address range.
2. When using the XC-DSC V3.20 or newer C Compiler for dsPIC® DSCs, enable the compiler option: `-merrata=base_offset (Project Properties>XC_DSC>xc-dsc-gcc>Additonal options)`.

Affected Silicon Revisions

A1	A2						
X							

23. Module: GPIO

When a GPIO is configured as an output and driven low, if an external signal sinks more than 8.7mA of current into a 4x pin, or 11.6mA on an 8x pin, the pin may become damaged.

Similarly, when a GPIO is configured as an output and driven high, if an external signal sources more than 5.8mA of current from a 4x pin, the pin may become damaged. 8x pins are not affected by this condition.

Work around

If an I/O is configured as an output and driven high or low, limit the current to or from the pin with a series resistor, or avoid direct pin contention scenarios.

Affected Silicon Revisions

A1	A2						
X							

24. Module: SPI

In the event of an overflow, the SPI reception is suspended when the Ignore Overflow bit is activated, and the SPIROV bit is not cleared.

Work around

Clear the SPIROV bit during the reception of the next byte that triggers the overflow. Additionally, SPIBUF must be read at least once.

Affected Silicon Revisions

A1	A2						
X	X						

25. Module: SILICON

Silicon only supports Industrial temp (-40°C to 85°C) and not Extended temp (-40°C to 125°C).

Affected Silicon Revisions

A1	A2						
X							

Work around

Use Silicon Rev A2 or later, which supports all three temperature ranges: Industrial (-40°C to 85°C), Extended (-40°C to 125°C) and High (-40°C to 150°C).

26. Module: CPU

If a Prefetch Branch Unit (PBU) cache invalidation request is issued and immediately followed by the CPU reading an IVT entry from Flash for exception processing, the CPU may enter a perpetually stalled state. The CPU will only recover in response to a hardware reset (Watchdog Timeout, MCLR, POR, etc.). Cache invalidation requests capable of triggering this erratum have three possible sources:

1. Manual cache invalidation: Firmware or DMA writes a '1' to the CHECON[CHEINV] bit.
2. RTSP operations: Firmware or DMA writes a '1' to the NVMCON[WR] bit while CHECON[CHECOH] == 1 and NVMADR < 0xC00000.
3. BOOTSWP instruction executed.

Work around

Either of the following workarounds can be used to overcome the issue.

Work around 1

Place the IVT in RAM instead of Flash. This option is recommended for best performance, as all interrupts will have 2-5 clocks less latency, and up to 3 clocks of latency jitter will be removed each time the CPU fetches an IVT entry.

Example 1

```
// Copies IVT from flash into RAM and begins using the RAM copy.
// Avoids cache invalidation errata. Call once at startup/initialization.
void SetIVTInRAM(void)
{
    static __attribute__((aligned(0x40))) void * ram_ivt[__IVT_NUM];
// __IVT_NUM requires XC-DSC v3.20 or newer

// If undefined, change to highest implemented vector number

    if(IVTBASE != (uint32_t)ram_ivt)
    {
        __builtin_memcpy(ram_ivt, (const void*)IVTBASE, sizeof(ram_ivt));
        uint32_t PACCON1Save = PACCON1;
        PACCON1 |= 0x00000000UL;
// BMXIRAMLWR/BMXIRAMHWR/IVTBASEWR must be set
         while((PACCON1 & 0x00000000UL) != 0x00000000UL);
// Infinite loop if lock bits set.
// SetIVTInRAM() MUST be called before any code that locks BMXIRAMLWR/BMXIRAMHWR/IVTBASE.
       int originalIPLMask = __builtin_write_DISICTL(7);
        BMXIRAML = (uint32_t)ram_ivt;
// Specify ram_ivt[]'s address range as read-only + executable
        BMXIRAMH = (uint32_t)ram_ivt + sizeof(ram_ivt);
       IVTBASE = (uint32_t)ram_ivt;
        PACCON1 = PACCON1Save;
        __builtin_write_DISICTL(originalIPLMask);
    }
}
```

Work around 2

Temporarily block all IVT vector fetches while triggering cache invalidation events. This option suspends interrupts but continues to allow traps to be detected and immediately serviced by a common handler routine.

Example 2

```
void __attribute__((interrupt, nocontent)) IVTReadFailException(void)
{
    // Common trap handler code - will be reentrant if multiple dissimilar traps
    // of increasing priority take place.
    //
    // Individual flag bits in INTCON1 and INTCON3 to INTCON5 may be polled to
    // determine the nature of this trap. Only flags that are serviced should be
    // cleared if this function is allowed to return so as to preserve status
    // for recursive instances.
}

// Cache invalidation errata workaround
void CacheInvalidationEvent(void)
{
    VFA = (uint32_t)IVTReadFailException;
// Fallback address to vector to when the IVT can't be read via the CPU-I bus
    uint32_t PACCON1Save = PACCON1;
    PACCON1bits.IVTBASEWR = 1;            // Must have IVTBASE write access
    while(PACCON1bits.IVTBASELK);         // Infinite loop if locked. IVTBASELK must never be set
when calling CacheInvalidationEvent()
    uint32_t IVTBASESave = IVTBASE;
    int DISICTLSave = __builtin_write_DISICTL(7); // Mask interrupts <= IPL7
    IVTBASE = 0;
// Set illegal IVT base address in SFR space: CPU will jump to [VFA] if a trap occurs
     NVMCONbits.WR = 1; /* and/or */ CHECONbits.CHEINV = 1; /* and/or BOOTSWP */
    (void)CHECON;                         // Dummy SFR read to ensure the prior SFR write
instruction has exited CPU pipeline
    IVTBASE = IVTBASESave;                // Reenable IVT usage
    PACCON1 = PACCON1Save;
    __builtin_write_DISICTL(DISICTLSave);// Reenable interrupts
}
```

Since this workaround blocks interrupts, and RTSP code often sets NVMCON[WR] many times before executing from any of the erased/reprogrammed address ranges, it is suggested that the CHECON[CHECOH] bit be cleared at Reset/code initialization. In such cases, this workaround would only need to be applied to manual cache invalidation and BOOTSWP events, not to NVMCONbits.WR = 1 operations. After all RTSP operations are complete, one manually triggered cache invalidation can be performed if there is any chance that stale instructions remain in the cache from before starting the RTSP session.

Example 3

```
int main(void)
{
    CHECONbits.CHECOH = 0;
// Disable automatic cache invalidation each time NVMCONbits.WR is set - do manual invalidation only

    // Do all RTSP/bootloader operations

    CacheInvalidationEvent(); // Set CHECONbits.CHEINV = 1 in here, not NVMCONbits.WR

    // Safe to begin executing from the reprogrammed flash regions now
```

Affected Silicon Revisions

A1	A2						
X	X						

Data Sheet Clarifications

The following typographic corrections and clarifications are to be noted for the latest version of the device data sheet (DS70005591**C**).

Note:	Corrections are shown in **bold**. Where possible, the original bold text formatting has been removed for clarity.

None.

APPENDIX A: DOCUMENT REVISION HISTORY

Rev A Document (05/2025)

Initial version of this document; issued for silicon revision A1.

Rev B Document (05/2025)

Updated silicon issue 25 (SILICON).

Rev C Document (10/2025)

Added silicon revision A2.

Rev D Document (2/2026)

Added silicon issue 26 (CPU).

Updated Example 1 and Example 2.

Added workaround to silicon issue 25 (SILICON).

Microchip Information

Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at https://www.microchip.com/en-us/about/legal-information/microchip-trademarks.

ISBN: 979-8-3371-2383-7

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications 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. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". 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 ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

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.

Microchip Devices Code Protection Feature

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

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.