dsPIC33CH512MP508 Family Silicon Errata and Data Sheet Clarification

The dsPIC33CH512MP508 family devices that you have received conform functionally to the current Device Data Sheet (DS70005371**E**), 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 dsPIC33CH512MP508 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 14, 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 dsPIC33CH512MP508 silicon revisions are shown in Table 1.

TABLE 1: SILICON DEVREV VALUES

Part Number	Device ID(1)	Revision ID for Silicon Revision				
		A0	A1	A2	A3	A4
Devices with CAN FD						
dsPIC33CH256MP505	0x7D42	0x0000	0x0001	0x0002	0x0003	0x0004
dsPIC33CH512MP505	0x7D52					
dsPIC33CH256MP506	0x7D43					
dsPIC33CH512MP506	0x7D53					
dsPIC33CH256MP508	0x7D44					
dsPIC33CH512MP508	0x7D54					

Note 1: The Device IDs (DEVID and DEVREV) are located at the last two implemented addresses of configuration memory space. They are shown in hexadecimal in the format "DEVID DEVREV".

TABLE 1: SILICON DEVREV VALUES (CONTINUED)

Part Number	Device ID[1]	Revision ID for Silicon Revision				
		A0	A1	A2	A3	A4
Devices with No CAN FD						
dsPIC33CH256MP205	0x7D02	0x0000	0x0001	0x0002	0x0003	0x0004
dsPIC33CH512MP205	0x7D12					
dsPIC33CH256MP206	0x7D03					
dsPIC33CH512MP206	0x7D13					
dsPIC33CH256MP208	0x7D04					
dsPIC33CH512MP208	0x7D14					

Note 1: The Device IDs (DEVID and DEVREV) are located at the last two implemented addresses of configuration memory space. They are shown in hexadecimal in the format “DEVID DEVREV”.

TABLE 2: SILICON ISSUE SUMMARY

Module	Feature	Item Number	Issue Summary	Affected Revisions				
				A0	A1	A2	A3	A4
I²C	Interrupt	1.	In Client mode, an incorrect interrupt is generated with DHEN = 1.	X	X	X	X	X
I²C	Error	2.	False bus collision error generated.	X	X			
I²C	Idle	3.	SFRs are reset in Idle mode.	X	X	X	X	X
I²C	SMBus 3.0	4.	When Configuration bit SMBEN (FDEVOPT[10]) = 1, the SMBus 3.0 VIH minimum specification may not be met.	X				
Oscillator	HS, XT	5.	Removed.	—	—	—	—	—
UART	FERR	6.	The FERR bit will not get set if one Stop bit is received.	X	X	X	X	X
UART	OERR	7.	The ninth byte received will not be available to be read.	X	X	X	X	X
UART	TXWRE	8.	TXWRE bit (UxSTAH[7]) cannot be cleared once it gets set.	X	X	X	X	X
UART	Address Detect	9.	When writing to UxP1 with UTXBRK = 1, content of P1 will not get transmitted.	X	X	X	X	X
UART	Address Detect	10.	In Address Detect mode, content of P1 is not transmitted on writing to P1 with UTXBRK = 1.	X	X	X	X	X
UART	Sleep	11.	When waking from Sleep with a UART reception, SLPEN needs to be set in addition to WAKE = 1.	X	X	X	X	X
UART	Smart Card	12.	The wait time interrupt flag is set when the last character transmitted has the bit, LAST = 0.	X	X	X	X	X
MBIST	MBISTDONE	13.	After executing a Reset, the MBISTDONE bit will always be set.	X	X	X	X	X
CPU	FLIM Instruction	14.	When the operands are of different signs, the FLIM instruction may not force the correct data limit.	X	X	X	X	X
CPU	MAXAB/MINAB/MINZAB Instructions	15.	When the operands are of different signs, the MAXAB, MINAB and MINZAB instructions may not output the correct value.	X	X	X	X	X
CPU	DIV.SD Instruction	16.	When using the signed 32-by-16-bit division instruction, DIV.SD, the Overflow bit is not getting set when an overflow occurs.	X	X	X	X	X
SCCP/MCCP	Clock Source	17.	Using FOSC as the clock source may cause synchronization issues.	X	X	X	X	X
I/O	POR	18.	Spike on I/O at POR.	X				
DMA	ADC Triggers	19.	DMA is triggered continuously from ADC.	X				
PWM	Time Base Capture	20.	The PWM Capture Status (CAP) flag will not set again under certain conditions.	X	X	X	X	X
I²C	I²C	21.	Instances of I²C/SMBus on silicon **A0-A2** may exhibit errors.	X	X	X		
Oscillator	VCO and AVCO Dividers	22.	Main and auxiliary PLL external VCO dividers can fail to output clock signal.	X	X			
Main Secondary Interface (MSI)	DMA Transfer	23.	DMA transfer of mailbox data.	X	X	X	X	X
Secondary CPU	REPEAT	24.	REPEAT loops interrupted by nested interrupts on the Secondary core may corrupt data and traps.	X	X	X	X	X
UART	UART	25.	TXWRE bit is not cleared when the transmitter is disabled.	X	X	X	X	X

TABLE 2: SILICON ISSUE SUMMARY (CONTINUED)

Module	Feature	Item Number	Issue Summary	Affected Revisions				
				A0	A1	A2	A3	A4
Secondary Core Program RAM	PRAM ECC	26.	Secondary core CPU may read from an unprogrammed PRAM location.	X	X	X	X	X
SCCP	Timer Interrupt	27.	In Capture mode, the CCP timer interrupt may not occur if the timer prescale is not 1:1.	X	X			
ADC	Differential-mode	28.	When using ADC in Differential-mode (DIFFx = 1) with Input Frequency (FSRC) above 50 MHz, the first result data may be incorrect.	X	X	X	X	X
Secondary CPU	Dual Partition PRAM	29.	Secondary core Partition 2 PRAM holes in Dual Partition mode.	X	X	X	X	

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 (**A4**).

1. Module: I²C

In Client mode with DHEN = 1 (Data Hold Enable), if software sends a NACK, a client interrupt is asserted at the ninth falling edge of the clock.

Work around

Software should ignore the client interrupt that is asserted after sending a NACK.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

2. Module: I²C

In Client mode, a false bus collision event is generated when the bus collision is enabled (SBCDE = 1) and a Stop bit is received.

Work around

None.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X				
Secondary	X	X				

3. Module: I²C

In Client mode, the SFRs are reset when the device is in Idle and the module is set for discontinue in Idle (I2CSIDL = 1).

Work around

None.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

4. Module: I²C

When selecting SMBus 3.0 operation using Configuration bit, SMBEN (FDEVOPT[10]), the Voltage Input High (V_{IH}) of the SMBus 3.0 specification minimum may not be met.

Work around

None.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X					
Secondary	X					

5. Module: Oscillator

This errata is no longer applicable to any silicon revisions of this product. See **Section 2.5 "External Oscillator Pins"** in the current device data sheet (DS70005371**E**) for guidance on oscillator design to avoid start-up related issues.

6. Module: UART

When the UART is operating with STSEL[1:0] = 2 (two Stop bits sent, two checked at receive) and STPMD = 0, the FERR bit will not get set if one Stop bit is received.

Work around

Use STPSEL = 3 instead of STSEL = 2. When operating with STSEL = 3 mode, the UART will be configured to send two Stop bits, but check one at receive.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

7. Module: UART

When the receive buffer overflows, the ninth byte received will get lost and cannot be read.

Work around

Do not allow the OERR bit to get set by reading the received data byte on each byte reception.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

8. Module: UART

Once the TX Write Transmit Error Status bit (TXWRE, UxSTAH[7]) gets set, the TXWRE bit cannot be cleared by a single clear instruction.

Work around

Use multiple clear instructions in a loop until the TXWRE bit gets cleared.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

9. Module: UART

In UART Address Detect mode, writing to UxP1 with UTXBRK = 1 should cause a break to be transmitted, followed by the content in P1x, but the content of P1x will not get transmitted.

Work around

After writing to P1x, wait for UTXBRK to get clear and then rewrite to P1x.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

10. Module: UART

In Address Detect mode, the content of P1x is not transmitted on writing to P1x with UTXBRK = 1.

Work around

Write P1x a second time after waiting for the break transmission to start.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

11. Module: UART

When waking from Sleep with a UART reception, SLPEN needs to be set in addition to WAKE = 1.

Work around

Set the SPLEN bit in addition to WAKE before entering Sleep.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

12. Module: UART

In Smart Card T = 1 mode, the wait time interrupt flag is set when the last character transmitted has the bit, LAST = 0.

Work around

Ignore WTC interrupt events on non-last bytes.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

13. Module: MBIST

After a Reset, the MBISTDONE status bit will be set regardless of a BIST test being executed. If a BIST is requested and executed, the MBISTDONE bit will set as expected.

Work around

None.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

14. Module: CPU

The FLIM instruction may incorrectly limit the data range when operating on signed operands of different sign values. If the operands are either all negative or all positive, the limit is correct.

Work around

None.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

15. Module: CPU

When operating on signed operands of different sign values, the output for MAXAB, MINAB and MINZAB instructions may be incorrect. If the operands are either all negative or all positive, the output is correct.

Work around

None.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

16. Module: CPU

When using the Signed 32-by-16-Bit Division instruction, DIV.SD, the Overflow bit may not always get set when an overflow occurs. This erratum only affects operations in which at least one of the following conditions is true:

- Dividend and divisor differ in sign,
- Dividend > 0x3FFFFFFF or
- Dividend < 0xC0000000

Work around

The application software must perform both the following actions to handle possible undetected overflow conditions:

a) The value of the dividend must always be constrained to be in the following range: 0xC0000000 ≤ Dividend ≤ 0x3FFFFFFF.

b) If the dividend and divisor differ in sign (e.g., dividend is negative and divisor is positive), then after executing the DIV.SD instruction or the compiler built-in function, __builtin_divsd(), inspect the sign of the resultant quotient. If the quotient is found to be a positive number, then treat it as an overflow condition.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

17. Module: SCCP/MCCP

When FOSC is selected as the clock source using the CLKSEL[2:0] bits (CCPxCON1L[10:8]), an unexpected operation may occur. For proper SCCP/MCCP input clock synchronization, do not use FOSC as the CCP clock source.

Work around

Use any of the other available clock sources in CLKSEL[2:0].

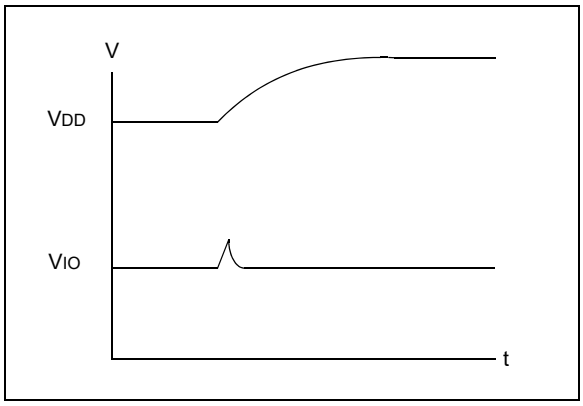
Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

18. Module: I/O

During a fast device power-up, when the VDD ramp is less than 4 mS, the I/O pins may drive up to 100 µA current for a duration of up to 10 µS (Figure 1-1).

FIGURE 1-1: I/O RAMP



Work around

1. Slow down the VDD ramp time (greater than 4 mS for VDD to ramp 0V to 3.3V).
2. Ensure the circuitry that is connected to the pins can endure this pulse.

   Example applications affected may include complementary power switches, where a transient current shoot-through might occur. High-voltage applications with complementary switches should power the high-voltage 200 µSec later than powering the dsPIC® device to avoid the current shoot-through. This behavior is specific to each device and not affected by aging.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X					
Secondary	X					

19. Module: DMA

The DMA receives multiple continuous triggers from the ADC until the trigger event from ADC is cleared. The OVRUNIF flag (DMAINTn[3]) will be set. When the OVRUNIF bit changes state, from '0' to '1', a DMA interrupt is generated.

Work around

Ignore the OVRUNIF bit and the first DMA interrupt. Clear the ADC trigger source, ANxRDY, with a DMA read of the ADC buffer, ADCBUFx, for the corresponding ADC channel.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X					
Secondary	X					

20. Module: PWM

When using a PWM Control Input (PCI) to trigger a time base capture, the Capture Status flag, CAP (PGxSTAT[5]), may not set again under certain conditions. When a subsequent PWM capture event occurs while, or just after, reading the current capture value from the PGxCAP register, the Capture Status Flag, CAP, will not set again.

Work around

Read the PWM Generator x Capture (PGxCAP) register as soon as possible to avoid the condition. Poll the CAP bit and read the PGxCAP value within the associated PWM Generator (1-8) interrupt or any of the six PWM Event (A-F) interrupts corresponding to the PCI event which triggered the time base capture.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

21. Module: I²C

Silicon Revision A0:

All instances of I²C/SMBus may exhibit errors and should not be used. When operating I²C/SMBus in a noisy environment, the I²C module may exhibit various errors. These errors may include, but are not limited to, corrupted data, unintended interrupts or the I²C bus getting hung up due to injected noise. Examples of system noise include, but are not limited to, PWM outputs or other pins toggled at high speed adjacent to the I²C pins. Both Host and Client I²C/SMBus modes may exhibit this issue.

Silicon Revision A1/A2:

The device revisions A1 and A2 have an improved I²C/SMBus module noise robustness due to improved filter design. The improved filter design includes a delay filter to increase the hysteresis on the I²C signals.

However, on a small subset of samples, when there is a high level of noise, I²C/SMBus may exhibit errors. Examples of system noise include any pins toggled at high frequency adjacent to the I²C pins, such as PWM outputs. The system noise can also be exaggerated in low temperature environments, below 0ºC. Errors may include, but are not limited to, corrupted data or address, bus collision, unintended flags set or the I²C bus getting hung up. Both Host and Client modes of all instances of I²C/SMBus may exhibit this issue.

Silicon Revision A3:

Revisions starting from A3 are unaffected by this issue.

Work around

Silicon Revision A0:

If I²C is required, use a software I²C implementation. An example I²C software library is available from Microchip:

https://www.microchip.com/en-us/software-library/dspic33c-i2c-softwarelibrary

Silicon Revision A1/A2:

The following guidance will help to mitigate the error in a noisy environment. If an error occurs, restarting the affected I²C/SMBus module will recover from the error condition.

Note: To successfully implement I²C/SMBus in an application, one or more of the following hardware or software workarounds may be required to be implemented. It is recommended to test the solution, including the workarounds as per the end application requirements.

Hardware:

- Decrease the value of pull-up resistors to ensure faster transitions of SDA/SCL signals.
- Add 10pF capacitors to SDA/SCL lines to filter out high-frequency noise.
- Limit I²C operating speed to 400 kHz or less.
- Reduce system noise during I²C/SMBus transactions as much as possible.

Software:

- Avoid placement of high-speed signals on pins adjacent to those used for I²C/SMBus.
- Monitor the I²C/SMBus status register to check for any errors during I²C/SMBus module execution to ensure no errors have occurred. If an error occurs, restarting the affected I²C/SMBus module will recover from the error condition.
- If I²C use is required beyond the conditions described above, use a software I²C implementation. An example I²C software library is available from Microchip:

https://www.microchip.com/en-us/software-library/dspic33c-i2c-softwarelibrary

Silicon Revision A3/A4:

No workaround is required for revision A3/A4.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X			
Secondary	X	X	X			

22. Module: Oscillator

At PLL start-up, the main and auxiliary PLL VCO dividers may occasionally halt and not provide a clock output. The VCO and AVCO dividers can be selected as clock sources for different peripheral modules, including the ADC, PWM, DAC, CAN FD, UART, etc.

All VCO and AVCO divider outputs, FVCO/2, FVCO/3, FVCO/4, FVCODIV, AFVCO/2, AFVCO/3, AFVCO/4 and AFVCODIV, are affected and may show the issue independently.

Any type of Reset may recover the VCO/AVCO divider clock outputs (Software Reset, WDT, $\overline{\text{MCLR}}$ or POR).

Work around 1

Use another clock source, such as the FOSC, PLL or APLL output (FPLLO and AFPLLO), instead of the VCO or AVCO dividers.

Work around 2

If the application requires the VCO/AVCO divider, peripheral activity should be verified within some time or the device should be reset.

The Watchdog Timer (WDT) or Timer1 may be used to establish the time-out period and reset the device. The following steps may be taken to implement this work around for any given peripheral and VCO/AVCO divider combination.

1) Set up the WDT or Timer1 time-out period.
2) Set up the VCO/AVCO divider source to be used by the peripheral.
3) Start the peripheral from this source.
4) Verify peripheral activity using an interrupt or other method and disable the time-out.
5) If the time-out expires, the device should be reset; WDT will reset the device without intervention, but Timer1 will require a SWR in the Timer1 Interrupt Service Routine.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X				
Secondary	X	X				

23. Module: Main Secondary Interface (MSI)

When transferring data between cores using the MSI mailbox with DMA, if the transmitting core is running more than two times the system clock frequency of the receiving core, the data transfer may not be processed correctly and the MSI may appear to be in a Freeze state.

An example of the application includes the DMA in the transmitting core that may load data to the MSI Mailbox register after the receiving core initiates the MSI interrupt to Acknowledge data reception, but prior to hardware clear of the DTRDY bit, causing the hardware to appear to be frozen in the state where DTRDY is set in the transmitting core and cleared in the receiving core.

Work around

Do not use DMA for MSI data transfer when the core sending data will be operating at more than two times the system clock frequency of the core receiving data. Instead, in the MSI ISR, clear the DTRDY bit and load the next data to the MSI buffer/FIFO directly.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

24. Module: Secondary CPU

While the Secondary CPU core is executing the instruction targeted by a REPEAT loop and two or more nestable interrupts occur in near simultaneous proximity, data corruption may occur within the lowest priority Interrupt Service Routine (ISR) and/or original REPEAT loop code. Specifically, when the CPU is vectoring to the lower priority ISR and a higher priority stimulus forces the CPU to vector to a different ISR, hardware will retarget the background REPEAT loop onto the first instruction of the lower priority ISR. Typically, ISRs start with a stack PUSH instruction, resulting in repeated stack pushes later when the lower priority ISR executes. This manifests as an Address Error Trap upon return from the low-priority ISR as the CPU attempts to use the repeated stack push data as the ISR's return address.

If the first instruction of the lower priority ISR is repeatable without harmful effects, such as a NOP, data corruption will still be apparent in the background code as its REPEAT loop will prematurely terminate.

Work around 1

Avoid using REPEAT instructions in projects built for the Secondary core. For compiled code, use MPLAB XC16, v1.61 or higher and specify -merrata=repeat_gie or -merrata=repeat_nstdis as an additional option for XC16 (Global Options). Alternatively, this option may be individually added to the command lines invoking xc16-gcc and xc16-ld.

-merrata=repeat_gie will disable interrupts before a REPEAT instruction is executed so that the REPEAT loop can execute uninterrupted. This is achieved by prefixing the REPEAT loop with instructions that save INTCON2, write GIE (INTCON2[15]) = 0 to globally disable all interrupts, then postfix with an instruction to restore INTCON2.

Make sure to not directly write GIE (INTCON2[15]) = 1 from within an interrupt context. Writing GIE = 1 from within an interrupt poses an issue in the following scenario:

1. The errata work around sequence described above is starting. GIE (INTCON2[15]) is being cleared, but there exists a two instruction cycle window where interrupts are not masked yet.
2. The ISR that contains the instruction to write GIE (INTCON2[15]) = 1 is triggered.
3. Upon return from the ISR, the REPEAT loop outside the interrupt is allowed to run but now with interrupts enabled.
4. Two more interrupts of different priority levels occur in proximity while still executing the original REPEAT loop, leading to the hardware errata condition.

To make sure that the work around is effective, all ISR handlers must save, clear and restore GIE (INTCON2[15]) rather than directly write to it.

-merrata=repeat_nstdis will disable interrupt nesting before a REPEAT instruction is executed. This is achieved by prefixing the REPEAT loop with instructions that save INTCON1, write NSTDIS (INTCON1[15]) = 1 to globally disable all interrupts, then postfix with an instruction to restore INTCON1. Toolchain libraries will continue to use GIE (INTCON2[15]) global interrupt masking instead of relying on NSTDIS (INTCON1[15]) to protect divide loops.

Note: Blocking interrupt nesting typically adds no latency to interrupt processing but increases worst-case interrupt latency for higher priority ISRs. A low-priority interrupt triggered immediately before a high-priority stimulus adds the entire execution time of the low-priority ISR to the worst-case response latency for the higher priority ISR.

Work around 2

Ensure that all REPEAT instructions only execute in a context where back-to-back interrupts of nestable priority are impossible, such as within IPL6 and IPL7 ISRs, or anywhere all enabled interrupts are known to be configured to the same priority level. Also, if the application implements periodic interrupts corresponding to internal/synchronously timed events, it may be possible to find or wait for an execution window where a REPEAT loop can deterministically complete without two nested interrupts able to clobber it. A PWRSAV call to enter Idle mode may help find synchronized, safe windows as code flow will halt, then resume in response to the next interrupt.

As compiled code can have REPEAT loops hidden within them, this work around should only be attempted on a per source file basis with Work around 1 applied for all other files that cannot be carefully controlled or which do not require REPEAT loops. It is additionally suggested that the full project disassembly listing be searched for REPEAT instructions and that proper interrupt masking, nest disabling or contextual state and timing conditions for safe REPEAT execution have been met.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main						
Secondary	X	X	X	X	X	

25. Module: UART

The TXWRE bit is not cleared when the UART transmitter is disabled.

Work around

To clear the TXWRE bit, the user can first flush the FIFO via writing TXBE = 1, then the TXWRE bit can be cleared.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

26. Module: Secondary Core Program RAM

After a Reset, the secondary core's Program RAM (PRAM) is uninitialized, and the ECC checksums for all addresses are random. Any read access to these uninitialized locations — by the CPU or DMA — may generate an ECC error trap.

To avoid these errors, the entire PRAM must be initialized by writing to it, which corrects the ECC checksum values. The "`_wipe_secondary()`" function performs the PRAM initialization.

Work around

The "`_wipe_secondary(1);`" routine is available in MPLAB XC16 v1.70 or higher, and the MPLAB IDE device family pack, dsPIC33CH-MP 1.9.207 or higher, is required. This routine will load the PRAM with a known value and valid ECC contents. The Main core application should call this routine at least once each time the device is powered on or after Reset, before loading the PRAM with the secondary core user application image.

MPLAB XC-DSC compiler version 3.30 includes calling the "`_wipe_secondary()`" routine by default. If using XC-DSC v3.30 and above, the routine does not need to be called in the Main core application.

The time needed to run the "`_wipe_secondary(1);`" routine is Main core execution speed-dependent but will be similar to loading a full-size image to a PRAM of a given size.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main						
Secondary	X	X	X	X	X	

27. Module: SCCP

In Capture mode, the CCPx timer interrupt, _CCTxInterrupt, may not occur if the Timer Prescale Value, TMRPS[1:0], is not configured for 1:1 operation.

Work around

If the _CCTxInterrupt is needed in Capture mode, maintain TMRPS[1:0] = 00 for 1:1 timer prescale.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X				
Secondary	X	X				

28. Module: ADC

When using ADC in Differential-mode (DIFFx = 1) with Input Frequency (FSRC) above 50 MHz, the first result data may be incorrect. The Single-Ended Channel mode is unaffected by this errata.

Work around

Use a slower input frequency of 50 MHz or less for the ADC initialization to write to the ADMODxL/H registers. After completion of the first data conversion of each channel in Differential-mode, the Input Frequency, FSRC, can be increased to the maximum specified frequency in **Section 24.0 "Electrical Characteristics"**.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X	X	
Secondary	X	X	X	X	X	

29. Module: Secondary CPU

When the Secondary core is configured to use the dual partition program memory map, Partition 2 contains inaccessible memory ranges where Program RAM should reside. The impacted address ranges are:

P2ACTIV	Program Bus Address		Access Result
	start	end	
0	0x401000	0x401FFF	Read: undefined (VFSLV failure, ECC exception or illegal opcode reset) Write: may corrupt existing Partition 2 contents at other addresses
0	0x404000	0x404FFF	
1	0x001000	0x001FFF	
1	0x004000	0x004FFF	

Primary CPU LDSLV/VFSLV execution, Secondary CPU instruction fetching, NVMCON initiated double-word programming operations and PSV/EDS/table reads are all affected when targeting the above address ranges. Single Partition mode and Partition 1 in Dual Partition mode are unaffected.

Work around

Do not store code/PSV constants, execute, read or program the impacted address ranges. To maintain symmetry between Partition 1 and Partition 2 that does not depend on the runtime P2ACTIV state, this work around should be applied to all dual partition projects built against the Secondary core, regardless of which partition they are programmed into.

Allocate reserved program space objects over the affected memory ranges by adding the following code to one (any) .c file in the project:

```
__prog__ const char __attribute__((section(".errataPgmHole1"), space(prog), address(0x000FFC), noload, keep, unused))
erratapgmHole1[0x1004];
__prog__ const char __attribute__((section(".errataPgmHole2"), space(prog), address(0x003FFC), noload, keep, unused))
erratapgmHole2[0x1004];
```

The reserved keep-out ranges are 0x4 addresses larger than the physically inaccessible memory ranges to ensure that instructions are not placed or executed immediately preceding the memory holes. Instruction prefetching extends the effective read address one Flash double word forward of the Program Counter. Apparent program space utilization will increase by 12 KB. However, the `noload` attribute causes null data to be stored for the objects, suppressing any data records from existing in .hex and assembly .s source image files that target the affected addresses (no increase to primary core Flash utilization). Code and PSV constants will automatically flow around these reserved spaces during linking, such that all 24KB of remaining PRAM can still be used despite being fragmented.

Affected Silicon Revisions

Core	A0	A1	A2	A3	A4	
Main	X	X	X	X		
Secondary	X	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 (DS70005371**E**):

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 (10/2018)

Initial version of this document; issued for revision A0.

Rev B Document (7/2019)

Removes original silicon errata issues 6 (PWM) and 18 (CPU). The issues are no longer relevant and were removed.

Updates silicon errata issue 18 (I/O)

Adds silicon errata issues 19 (DMA) and 20 (PWM).

Rev C Document (9/2019)

Updates device data sheet reference to the current revision D.

Rev D Document (2/2020)

Adds silicon issue 21 (I^2C).

Rev E Document (6/2020)

Adds silicon issues 22 (Oscillator) and 23 (Main Secondary Interface (MSI)).

Adds data sheet clarification 1 (Electrical Characteristics).

Removes silicon issue 5 (Oscillator) since it is no longer applicable.

Rev F Document (7/2020)

Adds silicon revision A1.

Updates the wording in silicon issue 21 (I^2C).

Adds data sheet clarification 2 (Functional Safety and Qualification Support) and 3 (Guidelines for Getting Started with 16-Bit Digital Signal Controllers).

Rev G Document (10/2020)

Updates silicon issue 22 (Oscillator).

Adds silicon issue 24 (Secondary CPU).

Updates data sheet clarification 2 (Functional Safety and Qualification Support).

Adds data sheet clarification 4 (dsPIC® Core Naming Convention).

Rev H Document (12/2020)

Adds silicon issue 25 (UART) and 26 (Secondary Core Program RAM).

Adds data sheet clarifications 5 (SPI), 6 (Oscillator), 7 (Electrical Characteristics), 8 (ADC), 9 (DAC), 10 (PTG), 11 (PPS), 12 (Electrical Characteristics), 13 (Main Core Memory Organization), 14 (Special Features), 15 (Special Features) and 16 (Secondary Core Memory Organization).

Rev J Document (4/2021)

Updates silicon issue 26 (Secondary Core Program RAM).

Adds silicon issue 27 (SCCP).

Adds data sheet clarification 17 (Electrical Characteristics).

Rev K Document (7/2021)

Adds silicon revision A2.

Rev L Document (9/2021)

Updates silicon issue 26 (Secondary Core Program RAM).

Adds silicon issue 28 (ADC).

Rev M Document (9/2023)

Removes all data sheet clarifications since corrections and clarifications have been included in the latest data sheet revision (DS70005371**E**).

Rev N Document (4/2024)

Updates the first work around in silicon issue 24 (Secondary CPU) and adds silicon issue 29 (Secondary CPU).

Rev P Document (8/2024)

Adds silicon revision A3.

Updates silicon errata issue 21 (I^2C).

Rev Q Document (9/2024)

Updates silicon errata issue 21 (I^2C).

Rev R Document (10/2025)

Updates silicon errata issue 26 (Secondary Core Program RAM).

Rev S Document (12/2025)

Adds silicon revision A4.

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/legalinformation/microchip-trademarks.

ISBN: 979-8-3371-2391-2

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.