

Getting Started with Dual Core

Author: Naveen Raj
Microchip Technology Inc.

DUAL CORE OVERVIEW

The dsPIC33C family is the first dual core device from Microchip. The dsPIC33C devices contain extensive Digital Signal Processor (DSP) functionality with a high-performance 16-bit MCU architecture.

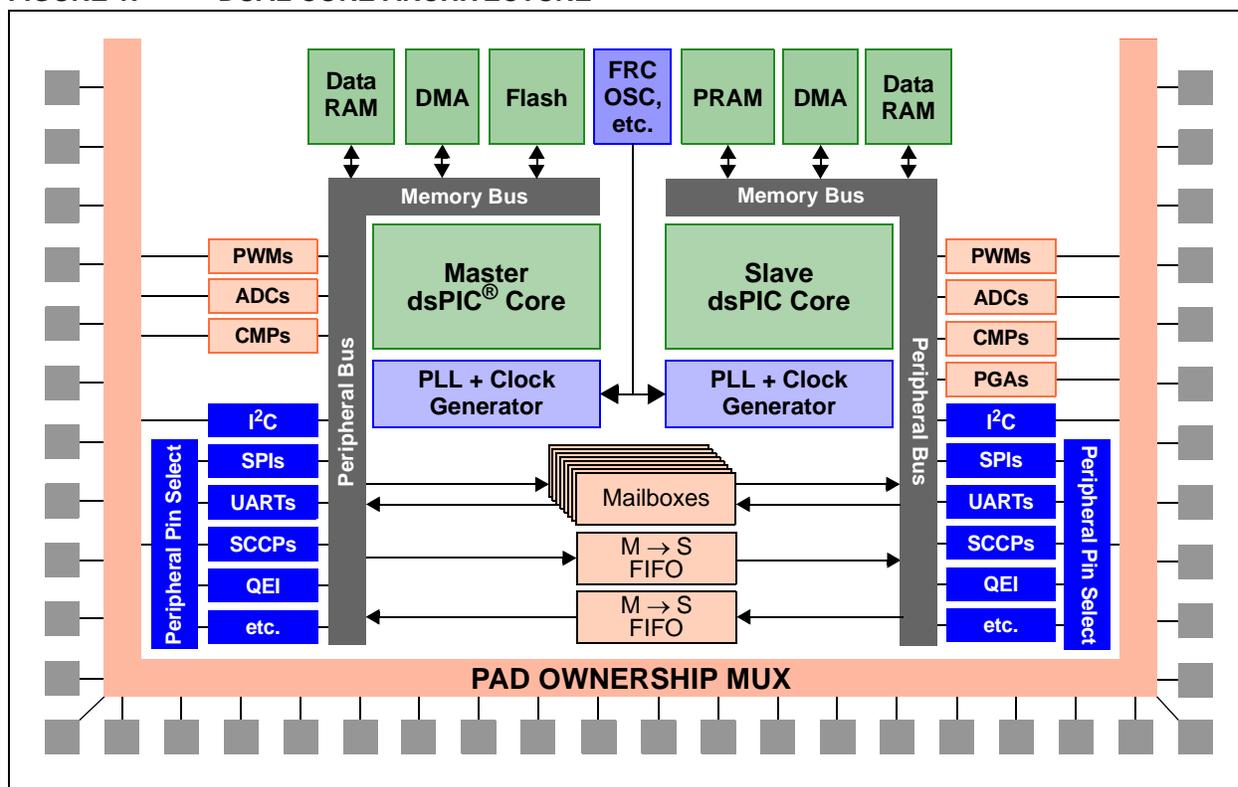
The Master core and Slave core can operate independently, and can be programmed and debugged separately during the application development. Both processor (Master core and Slave core) subsystems have their own interrupt controllers, clock generators,

Debug support, port logic, I/O MUXes and PPS. The device is equivalent to having two complete dsPIC® DSCs on a single die.

This application note assumes that the user is familiar with developing code on a single core dsPIC device. The discussion here will be pertaining to how to develop and debug code in a dual core environment using the Microchip MPLAB® X IDE. This document will discuss different debugging features and how to use the tool to get into those Debugging modes.

Figure 1 shows how the dual core is implemented and how each core has its own peripherals, RAM and program memory, etc.

FIGURE 1: DUAL CORE ARCHITECTURE



AN2721

Program Memory

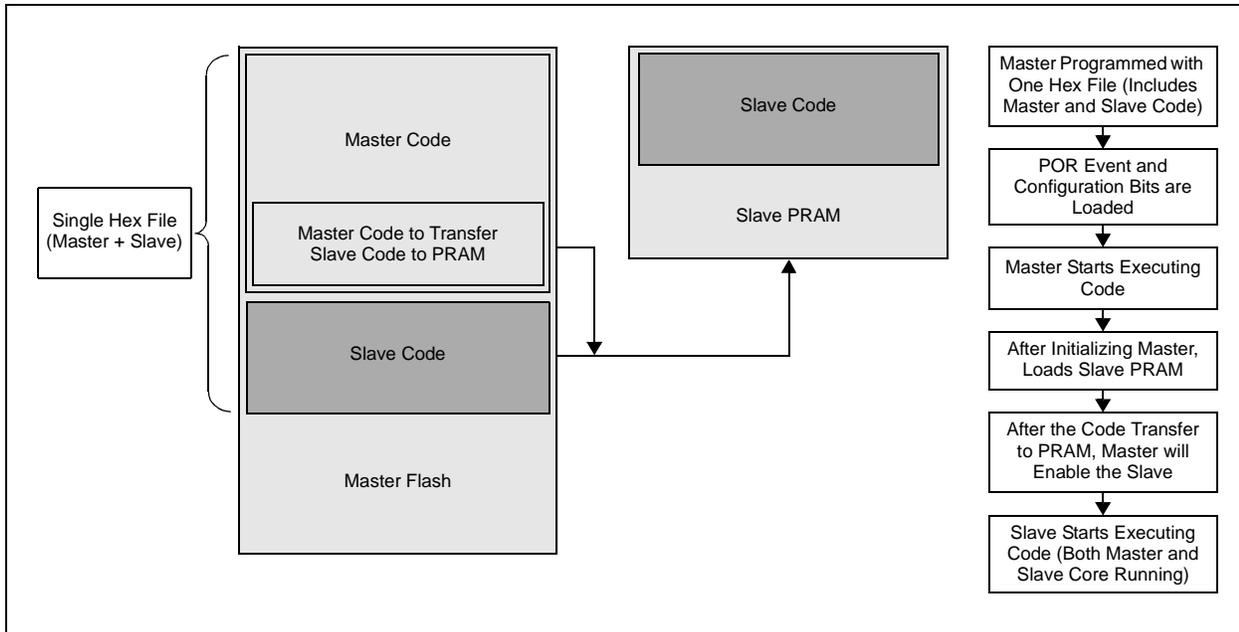
The Master core and Slave core each have their own program memory. On the dsPIC33CH device implementation, the Master program memory is Flash and the Slave program memory is PRAM. In the application, the code for Master core and Slave core is stored as a single hex file. This single hex file resides in the Master Flash.

Note: The use of Master and Slave terminology throughout this document relates to Master core and Slave core, respectively.

On a POR, the Master must transfer the Slave code from Flash to the Slave PRAM. On a power loss, the Slave PRAM will be lost, so on each power-up, the Master must repeat this process of transferring the Slave code from Master Flash to Slave PRAM. The Reset behavior of the cores is dependent on the Configuration bits, S1MSRE and S1SSRE. Please refer to the device data sheet for more details.

The process of transferring the Slave code from the Flash to the PRAM is done by software, so the Master core has full control of when to start the transfer. Once the Master core has completed the code transfer to the Slave, the Master core has control of when to enable the Slave core. This is also a part of the Master software, so the Master software can decide when to enable the Slave core (SLVEN = 1).

FIGURE 2: MASTER CORE TO SLAVE CORE CODE TRANSFER



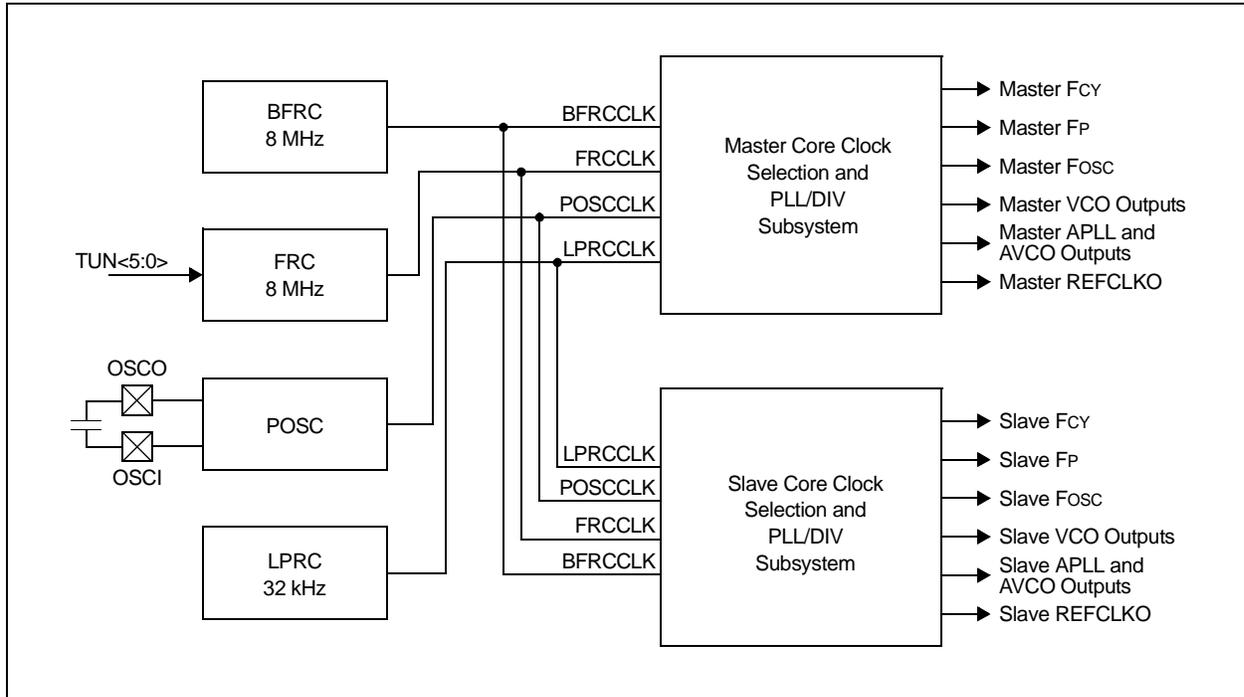
Data RAM

The Master core and Slave core each have their own data RAM; RAM size is dependent on the device. Please check the specific device data sheet for the available RAM on each core.

Oscillator

The oscillator has shared and individual modules across the two cores. FRC, BFRC (Backup FRC), LPRC and the Primary Oscillator are shared between the two cores.

FIGURE 3: OSCILLATOR RESOURCE ALLOCATION IN DUAL CORE



Each core has its own clock multiplexers, dividers, PLL and APLL modules. There are multiple options for clock selection, and the Master core and Slave core can run independently on different clock sources and different frequencies as needed.

The MSI module also has some features that can be used to interrupt the other core, as well as monitor the Reset state of the other cores. The feature for the Master core to start the Slave core is also a part of the MSI.

Configuration Bits

The Configuration bits are a part of the Master Flash. Both Master core and Slave core Configuration bits are in the Master Flash, so on a POR, the Configuration bits are loaded from the Master Flash.

I/O Ports and PPS

I/O ports are shared between the Master core and Slave core; the input function of the I/O ports goes to both Master core and Slave core. The output can be controlled by giving the pin ownership to either Master core or Slave core. The output pin ownership is controlled by the Configuration bits. By assigning the Configuration bits correctly, the user has full control of deciding which I/O port output is assigned to which core.

MSI Module

The Master Slave Interface (MSI) is an integral part of the dual core architecture. Using the MSI module, the Master core and the Slave core can transfer data between the two cores. The MSI module can work with DMA, so the data transfer between the two cores can take place with minimal CPU intervention.

The Master core and Slave core have their own Peripheral Pin Select (PPS) module, so through software, the Master core and Slave core can control which pins are assigned to their respective peripherals.

PROGRAMMING AND DEBUGGING ON THE DUAL CORE

During code development, the dual core has multiple features to help program and debug the silicon. All the different modes of programming, as well as debugging, are discussed in this section.

The various modes of debugging and programming options are:

1. Master Core Only Modes: Programming and debugging.
2. Slave Core Only Modes: Programming and debugging.

3. Dual Debugging: Both cores are debugged simultaneously.
4. Normal Mode: Master core programs the Slave core (final production code).

All of these modes for operation provide a lot of flexibility for the user in the development phase. The main advantage is the ability to develop projects on each core independently. This provides the edge for the user to have totally independent teams working on the Master project and the Slave project (Table 1).

TABLE 1: MODES OF DEBUG/PROGRAM OPERATION

Modes	Master Core	Slave Core	MCLR Pins		PGC/PGD Pins		# of Tools
			Master	Slave	Master	Slave	
Master Core Only ⁽¹⁾	Program/Debug	N/A	MCLR	N/A	PGC1/PGD1 or PGC2/PGD2 or PGC3/PGD3	N/A	1
Slave Core Only ⁽¹⁾	Program the Stub ⁽²⁾	Program/Debug	MCLR	MCLR	PGC1/PGD1 or PGC2/PGD2 or PGC3/PGD3	PGC1/PGD1 or PGC2/PGD2 or PGC3/PGD3	1
Dual Debug	Debug	Debug	MCLR	S1MCLR1 or S1MCLR2 or S1MCLR3	PGC1/PGD1 or PGC2/PGD2 or PGC3/PGD3	S1PGC1/S1PGD1 or S1PGC2/S1PGD2 or S1PGC3/S1PGD3 ⁽³⁾	2
Normal Mode	Program/Debug ⁽⁴⁾	Program Only	MCLR		PGC1/PGD1 or PGC2/PGD2 or PGC3/PGD3		1

Note 1: Programming or debugging.

2: Master stub can be a small code or a full Master project. The main reason for requiring a stub for the Master core is to have Configuration bits programmed. The Slave core and Master core Configuration bits reside in Master Flash.

3: The only time S1MCLR_x and S1PGC_x/S1PGD_x pins are used is when both the Master core and Slave core are debugged simultaneously (Dual Debug mode).

4: In Final Production mode, the Slave hex code resides in the Master Flash and the Master code loads the Slave code. The Master starts the Slave core after the Slave is programmed.

MASTER CORE ONLY MODE

Master Core Only Programming and Debugging

This mode is the same as programming single core dsPIC devices. The project can be created and debugged, ignoring the Slave code. Assign the I/O ports ownership and other Configuration bits that are required for the Master code to run. A Slave project or code is not required to run the Master Core Only mode.

The device is programmed using the (ICSP™) $\overline{\text{MCLR}}$ pin and the PGCx/PGDx pins are selected using the Configuration bits.

Select the device variant, for example, the dsPIC33CH128MP508 device and create a new project. The project can be compiled, programmed or debugged like any other existing Microchip single core device.

SLAVE ONLY MODES

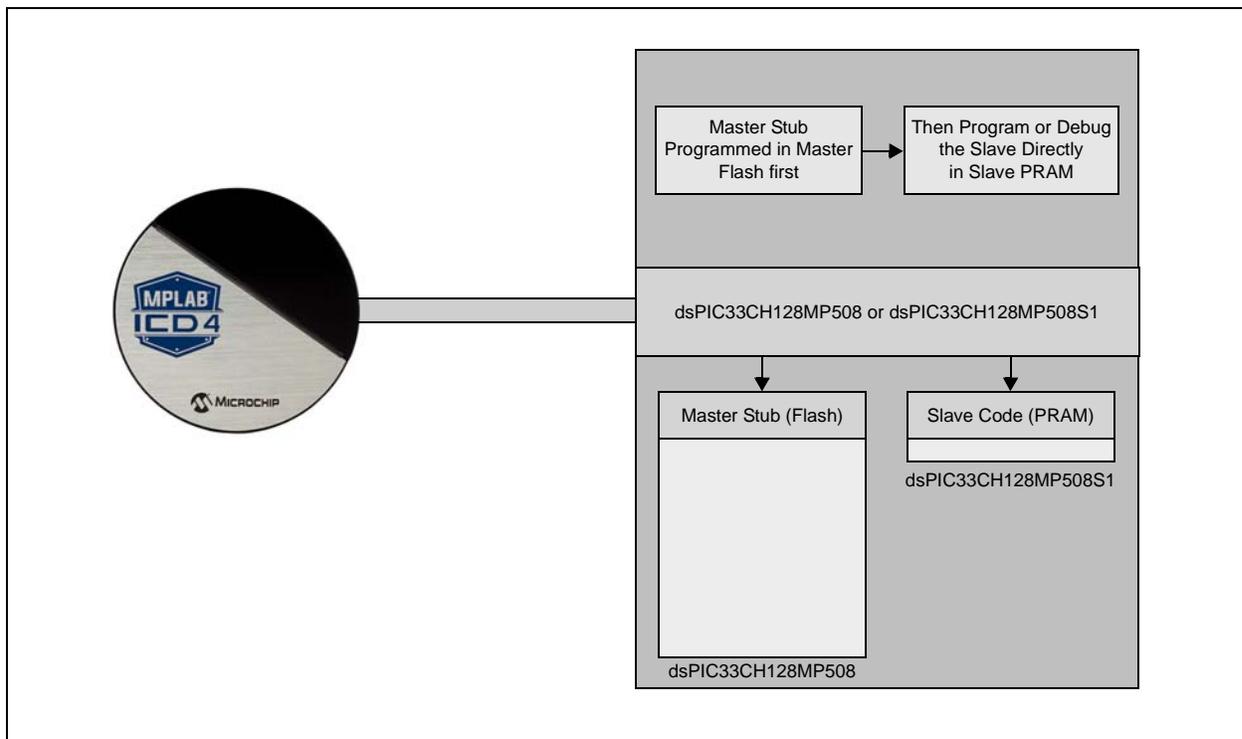
These modes use the dual core features. In these modes, the Master core has to be programmed. The Master code can be full Master code or small Master code with a "while(1)" (called Master stub).

The idea of the Master code or a Master stub is to program the Master core with a small code, which has the code to enable the Slave core. The Slave core is enabled with the code, `SLVEN = 1`. There is also an unlock sequence required to enable the Slave core. Please refer to the specific device data sheet for more information.

The XC16 provides a function which includes the unlock sequence. It is recommended to include, `#include <libpic30.h>`, and use function, `_start_slave();`, to start the Slave core.

The Master stub also has the Configuration bits for the Master and Slave core, since Configuration bits are stored in Master Flash.

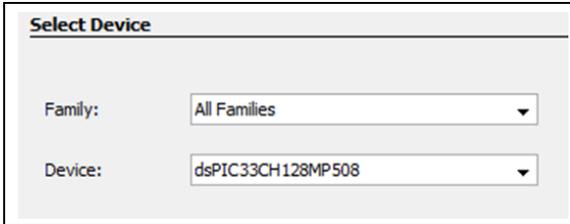
FIGURE 4: SLAVE ONLY PROGRAMMING/DEBUGGING



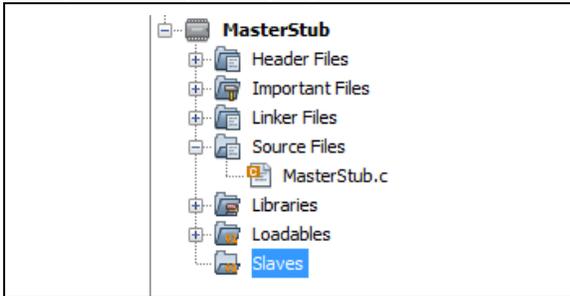
Slave Only Programming

Create the Master project like a single core device and perform the following steps:

1. Create a project for the Master stub selecting the Master device.



2. Navigate to the Slave folder and right click to open the Project Property window.



3. Uncheck the Debug check box (since the Slave core is not being debugged).
4. Program the Master code with the Master stub code.

Creating the Slave Project

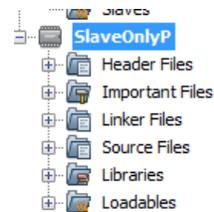
Once the Master project is programmed, start a new project for the Slave. Then, perform the following steps:

1. Select the Slave device (the Slave project should be the Slave device of the Master project. For example, if the Master project selected is dsPIC33CH128MP508, the Slave project selected should be dsPIC33CH128MP508S1. If the Master stub project is dsPIC33CH64MP508, then the Slave project selected should be dsPIC33CH64MP508S1.



For example, if a new Slave project, **SlaveOnlyP**, is selected as shown below, right click the Slave and set the Slave as the **Main Project**.

2. Start programming the Slave directly.



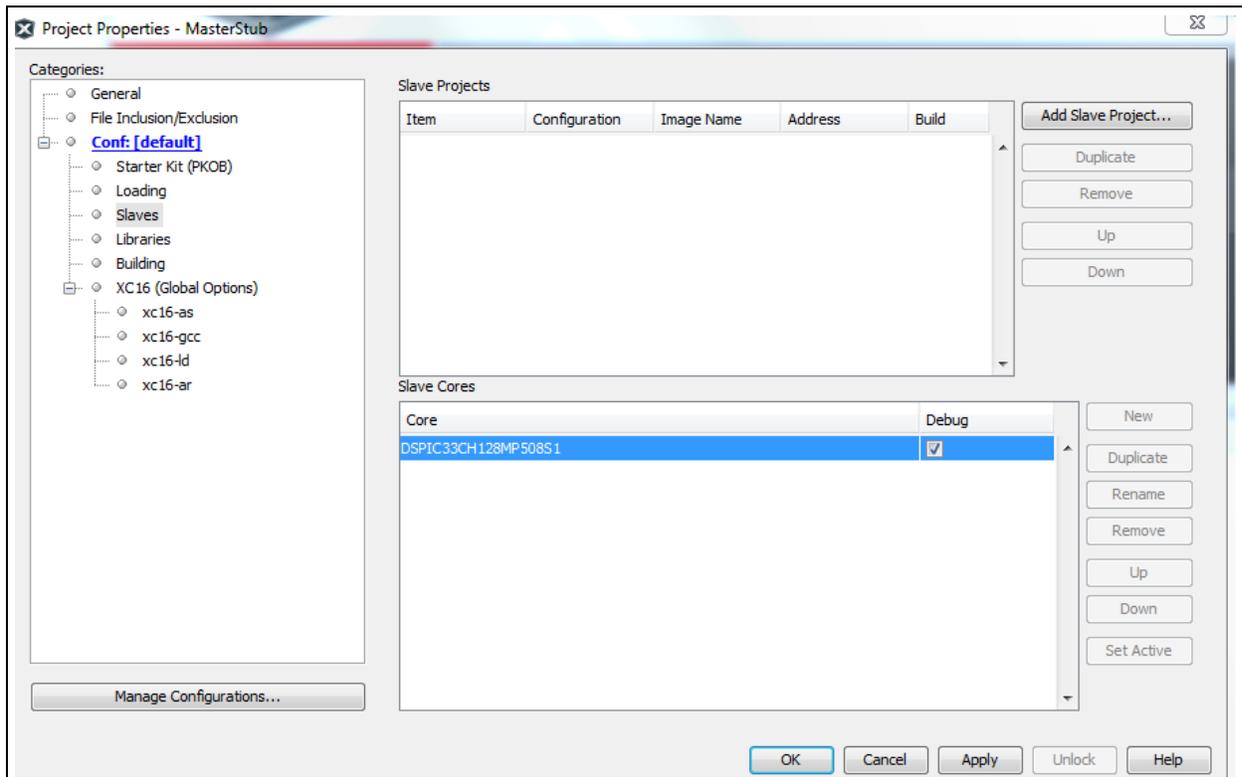
Note: The Master stub must be programmed prior to programming the Slave project. Without programming the Master stub, the Slave should not be programmed.

The Master and Slave can be programmed using the same ICSP™ pins (PGCx/PGDx).

Slave Only Debugging

In this mode, the Master stub has to be programmed first, as explained in “[Slave Only Programming](#)”. The main difference is the Debug check box selection option.

In the Master stub, go to the Slave folder and right click to open the window below.



Once the window is open, click the Debug check box to enable the Debug bit. This makes the Slave core ready for debugging.

In the Master code, there are two options:

- Enable the Slave, `SLVEN = 1` (`"_start_slave();"`). This ensures the Slave core is enabled and ready.
- Isolate the Slave from the Master code using the Configuration bit, `S1ISOLAT` (`S1ISOLAT = ON`). This will isolate the Slave from Master code and the Slave can start, irrespective of the `SLVEN = 1` or `0`. The `S1ISOLAT` bit is available only when the Slave is being debugged.

Creating the Slave Project

Once the Master stub project is programmed, start a new project for the Slave core. Select the Slave device. The Slave project should be the Slave device of the Master project.



The same ICSP pins (`MCLR` and `PGCx/PGDx`) will be used for programming the Master stub project as well the Slave project. Depending on the project, the tool will use the ICSP/`MCLR` pins to switch between the Master and Slave core.

Once the project is created and the Slave code is compiled, the Slave project can be debugged like any other project by clicking the **Debug**  button.

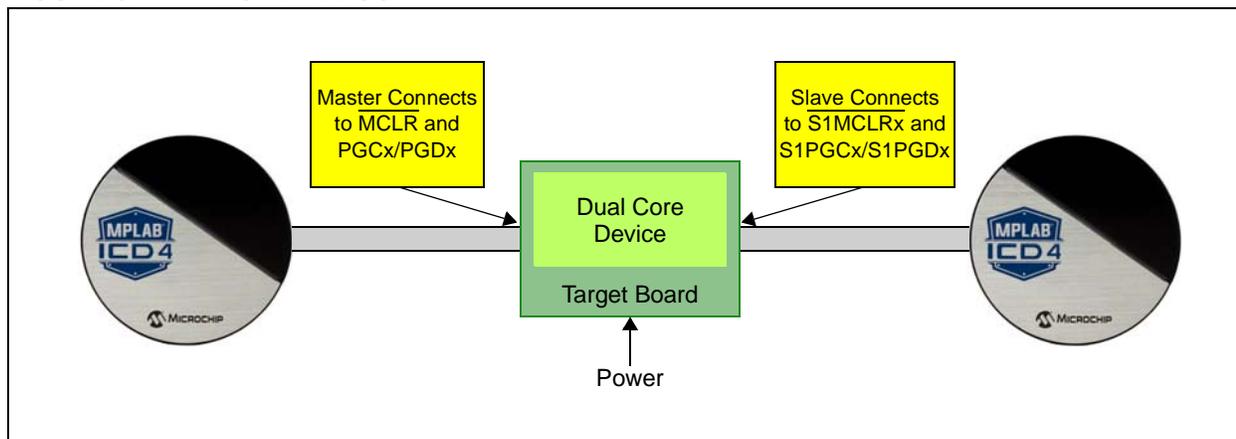
AN2721

DUAL DEBUG MODE

In this mode, both the Master and the Slave core are in Debug mode. The Master and Slave are debugged at the same time and two debug tools are required ([Figure 5](#)).

S1MCLR_x/S1PGC_x/S1PGD_x are only used in Dual Debug mode to debug the Slave project.

FIGURE 5: DUAL DEBUG



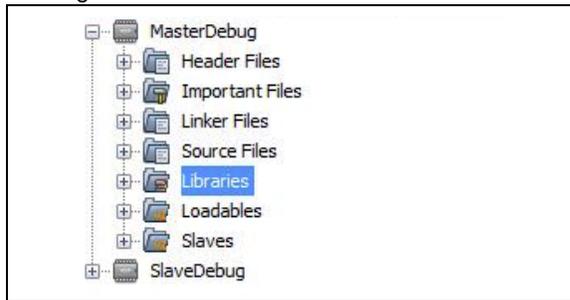
There are two distinct ways of debugging the Master and Slave simultaneously:

- Using a single MPLAB X IDE window.
- Using two MPLAB X IDE windows, one for Master and one for Slave.

Dual Debug with Single Window

Create a Master project by performing the following steps:

1. Right click the Slaves subfolder.

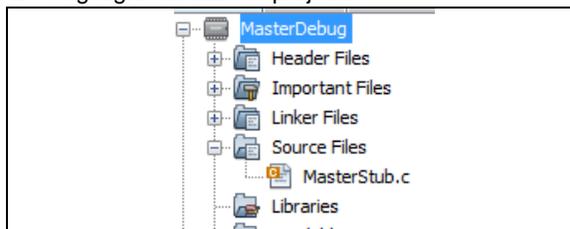


2. Select the Slave Debug check box to enable the debug bit.

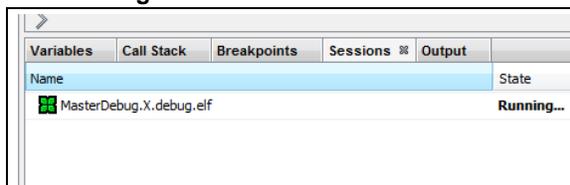


3. Make sure neither the Slave nor Master is set as the main project. If the Master or Slave project is selected as a main project, right click the specific project and click **Unset as the Main Project**.

4. Highlight the Master project.

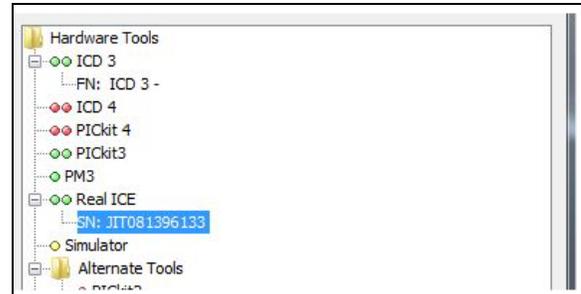


5. Press the **Debug** button to debug the Master project.
6. Once the Master is debugging, go to **Windows, Debugging** and select **Sessions**. This opens a Sessions window with the Master shown as **Running**.

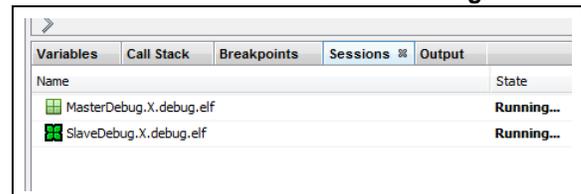


Note: The Master code should have the code enable the Slave for the Slave to run (SLVEN = 1). If not, the S1ISOLAT bit should be ON.

7. Now for the Slave core, create a Slave project and click the Slave project to highlight the project.
8. Select the project properties and select the second tool for debugging the Slave (in the example shown below, the ICD 3 is for the Master core and REAL ICE™ is for the Slave core).



9. Apply and click the **Debug** button.
10. Once the Slave core is debugging, go to **Windows, Debugging** and click on **Sessions**. This opens a Sessions window with the Master core and Slave core shown as **Running**.



11. Double click the Master or Slave session to highlight the window above the selection. All tool buttons on the window that are relevant to the session are highlighted.

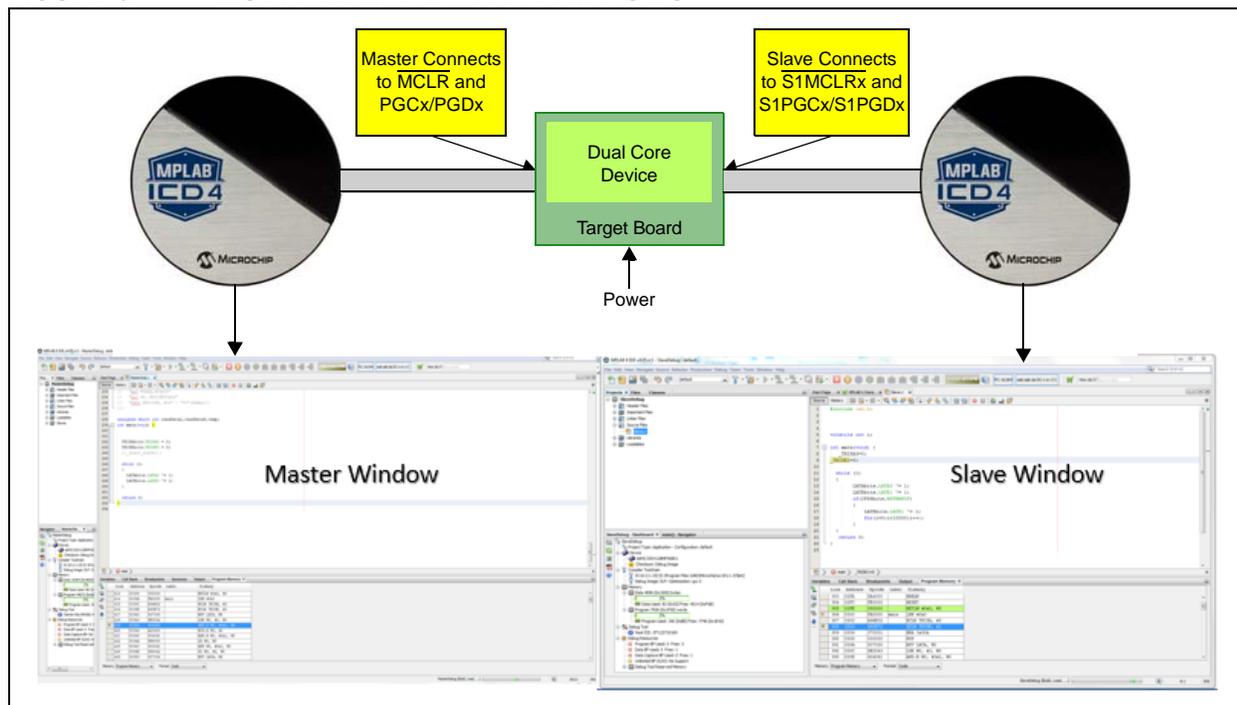


AN2721

Dual Debug with Multiple MPLAB X IDE Windows

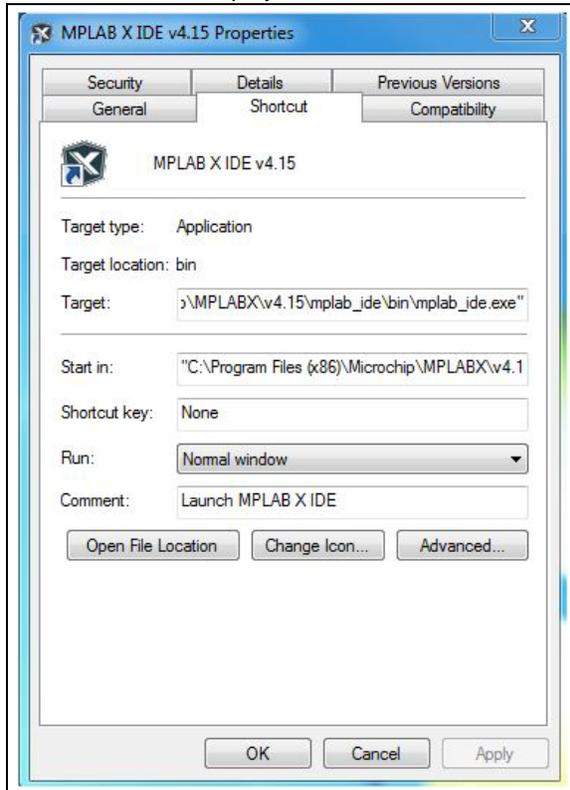
The hardware connection is the same as before with the single window. The only difference is the Master core will be in one MPLAB X IDE window, while the Slave core will be in another MPLAB X IDE window.

FIGURE 6: MULTIPLE MPLAB® X IDE WINDOWS



Create a dual debug project in multiple MPLAB X IDE windows by performing the following steps:

1. Create a Master core debug project. Make sure to enable the Slave core debug.
2. Right click the MPLAB X IDE icon and select **Properties**.
3. Go to Target location and add '`< - -userdir> < Slave project path>`' after the "`...\mplab_ide_.exe`". This provides the path to where the Slave project needs to be created.



4. Double-click the MPLAB icon and create a new Slave project. This opens a new MPLAB X IDE window for the Slave project. Using the second debug tool, the Slave core can be debugged.

Breakpoints in Dual Debug Mode

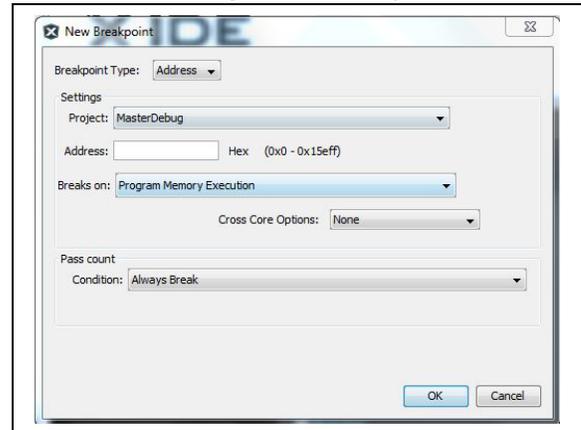
One of the key features of the Debug modes in the dual core is to create cross core breakpoints and interrupts. There are multiple features related to cross core breakpoints in the dual core device.

On a breakpoint, the user can opt for:

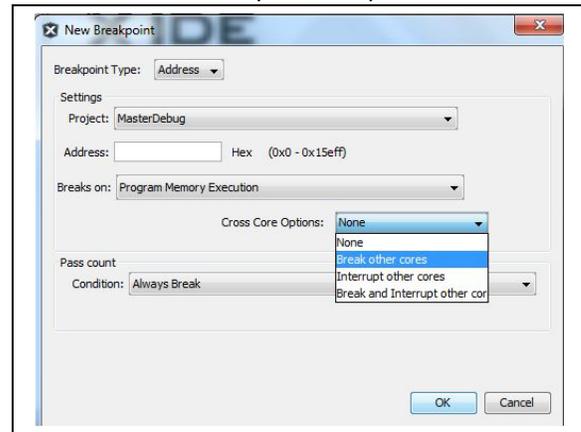
- Break other cores
- Interrupt other cores
- Break and interrupt the other core

Set a cross core breakpoint by performing the following steps:

1. Select the **Debug** menu and click **New Breakpoint**.
2. The **New Breakpoint** window opens.



3. Select the Address breakpoint and select which project the breakpoint is going to be. The decision to break the Master project or the Slave project can be done here.
4. Select the address where the code needs to break. In this example, the Master project is set to break at address, "0x360", and then decide which cross core option is required.



Since the main project is Master debug, the cross core which will break or interrupt will be the Slave.

Select what will appear in the **Cross Core Options** by selecting one of the following:

- **Break other cores:** When the Master reaches address 0x360, the Slave will also stop.
- **Interrupt other cores:** When the Master code reaches 0x360 and stops, the Slave will receive an interrupt (MSTBRKIF).
- **Break and interrupt the other core:** When the Master reaches address 0x360 and stops, the Slave will stop as well (MSTBRKIF interrupt flag will also get set).

For example, if the **Break other cores** is selected, when the Master stops at 0x360, the Slave will also stop.

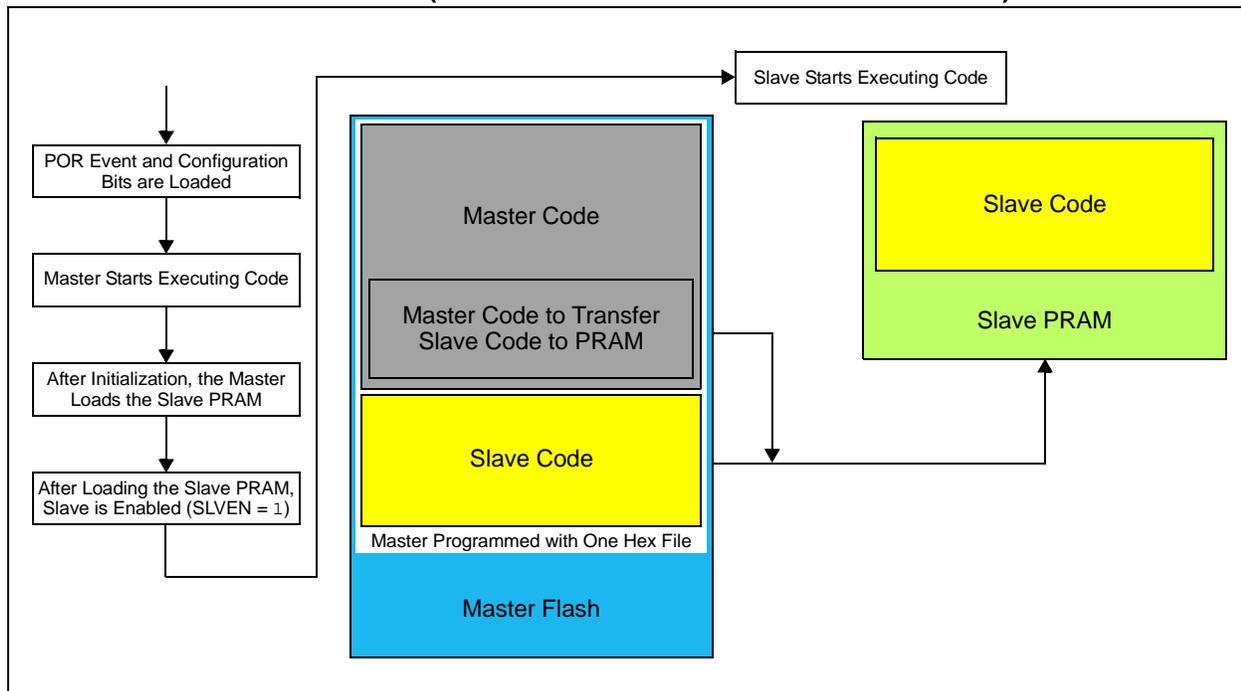
AN2721

NORMAL MODE (MASTER CORE PROGRAMS THE SLAVE CORE – FINAL PRODUCTION CODE)

This Normal mode is the final production code. The Slave code resides in the Master core. During power-up, the Master core has to transfer the code to the Slave core.

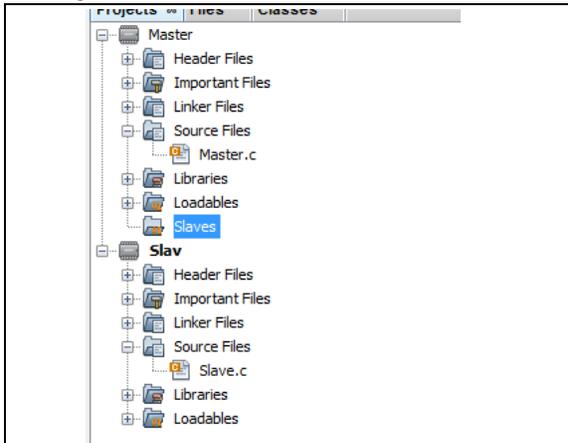
The Slave project is linked to the Master project, and when the Master project is built, the Slave project is also built at the same time. The Slave code is then placed in the Master Flash along with the Master code. When the Master code is running, the Master transfers the Slave code from Master Flash to the Slave PRAM (Figure 7).

FIGURE 7: NORMAL MODE (MASTER AND SLAVE IN A SINGLE HEX FILE)

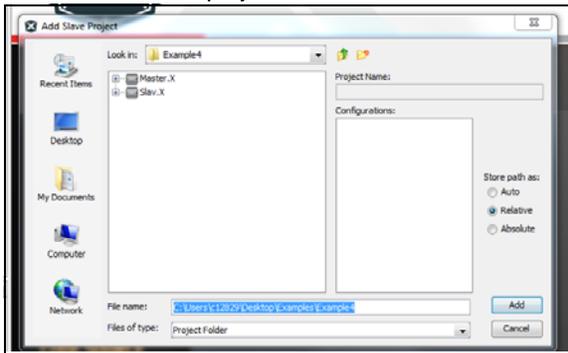


Create a Normal mode project:

1. Create a Master project (Master) and a Slave Project (Slav).
2. Right click the **Slaves** folder under the Master.

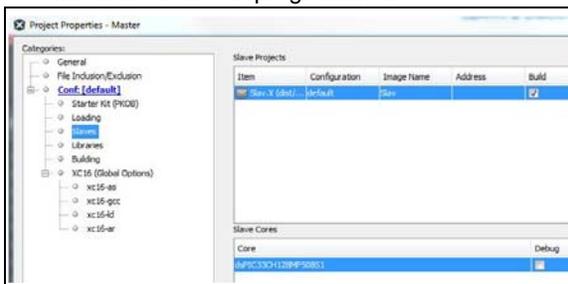


3. Add the Slave project as shown below.



4. Once the Slave project is added to the Master project, right click and open **Properties**. Once the Slave code is added and lined to the Master project, there will be only one active project. When the Master project is compiled, the Slave project will also be compiled.

Each time when the project is built, any changes made to the Master or Slave source code will be compiled to get a hex file. Only one hex file is generated and is placed in the Master Flash when the device is programmed.



Note: In the MPLAB X IDE project properties, make sure to leave the Debug check box unchecked. In this project, once the Master project is built, the Slave and the Master will get built and there will be only one single hex file that needs to be programmed in the Master Flash.

In this mode of operation, the Master can run in Debug mode and the Slave cannot run in Debug mode. When the Master core is stopped or single-stepped, the Slave core will continue to run as long as the Slave code is transferred and the Slave is enabled.

Slave Image Option

SLAVE IMAGE LOCATION

While linking the Slave project to the Master project, MPLAB X IDE provides the option for the user to assign the address for the Slave image. This is required only when the user needs to place the Slave code in any specific address location in the Master Flash. If the address location is left blank, the compiler will decide where to place the Slave image in the Master Flash.

The following shows an example of how the Slave image is assigned at address 0x1000.

Item	Configuration	Image Name	Address	Build
Slav.X (dist/...	default	Slav	0x1000	<input checked="" type="checkbox"/>

By adding address 0x1000, the Slave image will be kept at location 0x1000 in the Master core (if available and free).

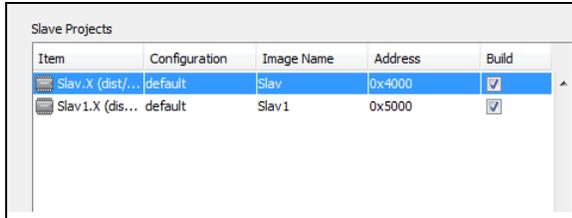
AN2721

Multiple Slave Images

KEEPING MULTIPLE IMAGES OF THE SLAVE IN MASTER FLASH

The Slave image location feature can be used to store multiple Slave code images if needed. This allows the user to change the Slave code by loading the desired Slave image.

In the Slave folder, multiple images of the Slave project can be added. For example, there are two Slave projects shown below.



Item	Configuration	Image Name	Address	Build
Slav.X (dist/...)	default	Slav	0x4000	<input checked="" type="checkbox"/>
Slav1.X (dis...)	default	Slav1	0x5000	<input checked="" type="checkbox"/>

It is possible to build both projects while building the Master project.

In the Master project, the user has the flexibility to choose which project image to load to the Slave.

```
#include "Slav.h"
#include "Slav1.h"
int main(void) {
if (condition1)
{
_program_slave(1,0,Slav); // load image Slav
}
else
{
_program_slave(1,0,Slav1); // load image Slav1
}
_start_slave();

while(1)
}
```

Depending on Condition 1, the Master can load the Slave PRAM (Slav image or Slav1 image).

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.

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, KEELoC® 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.

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

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Helder, JukeBlox, KeeLoq, Klear, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KlearNet, KlearNet logo, memBrain, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICKit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, 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.

© 2018, Microchip Technology Incorporated, All Rights Reserved.
ISBN: 978-1-5224-3287-6



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://www.microchip.com/support>
Web Address:
www.microchip.com

Atlanta

Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX

Tel: 512-257-3370

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

Novi, MI
Tel: 248-848-4000

Houston, TX

Tel: 281-894-5983

Indianapolis

Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles

Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC

Tel: 919-844-7510

New York, NY

Tel: 631-435-6000

San Jose, CA

Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto

Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880-3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

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

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7289-7561

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820