Adding DMA Support for Peripherals on PIC32CZ CA and SAM E70/S70/V7x MCUs using MPLAB Harmony v3 and MCC



TB3370

www.microchip.com Product Page Links

Introduction

In modern computing systems, efficient data transfer mechanisms are crucial for maintaining high performance and responsiveness, especially in applications that handle large volumes of data. Direct Memory Access (DMA) is one such mechanism that is pivotal in optimizing data movement. The DMA significantly reduces processing bottlenecks by allowing hardware subsystems to access the system independently of the CPU. Additionally, the use of a dedicated DMA controller ensures that data transfers are managed seamlessly further enhancing system efficiency and performance.

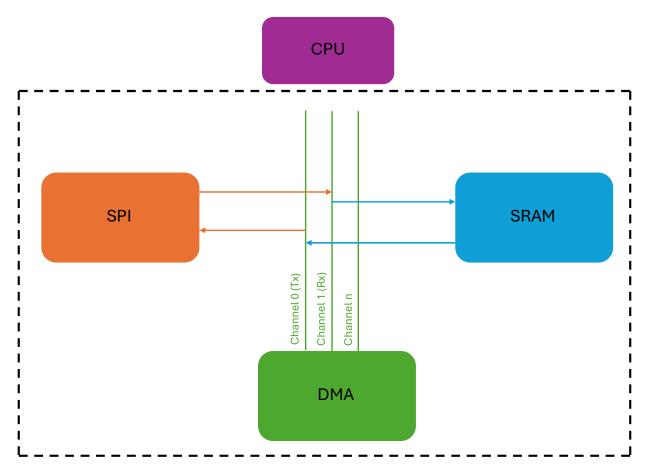
This document describes how to configure and use DMA with peripherals, such as the SPI on PIC32CZ CA and SAM E/S/V family of devices using MPLAB[®] Harmony v3 and MCC.

1. What is DMA?

Direct Memory Access (DMA) is an essential feature allowing specific hardware subsystems to access the system memory (RAM) independently of the central processing unit (CPU). This capability significantly enhances the efficiency and performance of data transfers, particularly in applications requiring the rapid movement of large data volumes, such as multimedia, networking, and storage systems.

The DMA process is managed by a dedicated hardware component known as the DMA controller, which can be integrated into the CPU or exist as a separate chip. The controller typically includes multiple channels, each capable of handling separate data transfer operations, and utilizes registers to store memory addresses, transfer counts, and control information. The primary advantages of the DMA include improved efficiency, faster data transfer rates, and reduced CPU overhead, making it indispensable in applications like audio and video streaming and real-time embedded systems.

Figure 1-1. DMA Block Diagram



1.1 Working Principle of DMA

The DMA transfer can be started only when a DMA transfer request is detected. The transfer requests may be software, peripheral, or an event. The DMA operation begins with the CPU initializing the DMA controller, followed by a peripheral device requesting a transfer. The DMA controller then arbitrates between multiple requests, if necessary, and takes control of the system bus to perform the data transfer directly between memory and the peripheral device, thereby freeing the CPU to perform other tasks. Upon completion, the DMA controller notifies the CPU through an interrupt.



1.2 Types of Data Transfers in DMA

The DMA data transfers are essential for optimizing system performance by enabling efficient communication between different components. These data transfers are categorized based on the source and destination of the data. It is categorized into the following four categories:

- Peripheral-to-Memory transfer
- Memory-to-Peripheral transfer
- Peripheral-to-Peripheral transfer
- Memory-to-Memory transfer

Table 1-1. Types of Data Transfers in DMA

Туреѕ	Source Location	Destination Location
Peripheral-to-Memory	Peripheral	Memory
Memory-to-Peripheral	Memory	Peripheral
Peripheral-to-Peripheral	Peripheral	Peripheral
Memory-to-Memory	Memory	Memory

Note: The SAM E/S/V family of devices does not support Peripheral-to-Peripheral type of data transfer.

The DMA transfer mode can also be categorized based on the size of the data being transferred by the components:

- **Beat transfer**: Size of one data bus transfer.
- **Block transfer**: Amount of data one transfer descriptor can transfer.
- **Burst transfer**: Back-to-back DMA transfer without CPU intervention.
- DMA transaction: Complete transfer of all data in a linked list of descriptors.
- **Cycle Stealing**: The DMA controller interrupts the CPU after every cycle to transfer data.



2. Creating the Application Using MPLAB Harmony v3 and MCC

The following software and hardware tools are used for this demonstration:

- MPLAB X IDE v6.20
- MPLAB Code Configurator Plugin v5.5.1
- MPLAB XC32 Compiler v4.45
- csp v3.20.0
- PIC32CZ CA90 Curiosity Ultra Development Board
- PIC32CZ CA80 Curiosity Ultra Development Board
- SAM E70 Xplained Ultra Evaluation Kit
- SAM V71 Xplained Ultra Evaluation Kit

Note: The updated versions of the above listed tools can also be used to create the application, and users are not restricted to use the older versions. Also, only one of the above mentioned boards is needed for creating the application.

2.1 Creating Demo Application Using PIC32CZ CA90 Curiosity Ultra Development Board

To create an MPLAB Harmony v3-based project, follow these steps:

- 1. From the **Start** menu launch MPLAB X IDE.
- 2. On the File menu, click New Project or click on the New Project icon.
- 3. The **New Project** window will be displayed. From the **Steps** navigation pane, click **Choose Project**.
- 4. In the right **Choose Project** property page:
 - a. Categories select Microchip Embedded.
 - b. Projects select **Application Project(s)**.
- 5. Click Next.

Figure 2-1. New Project Window

Steps		Choose Project	
1. Choose 2	Project	Q, Filter:	
		Categories: Microchip Embedded Other Embedded Generic	Projects: Application Project() Probuit (Hex, Loadable Image) Project User Makefile Project Ubrary Project
		Description:	
		Creates a new application project. It u	uses an IDE-generated makefile to build your project.



 Click Select Device and in the right Select Device property page, for Device select PIC32CZ8110CA90208 for creating the project on the PIC32CZ CA90 Curiosity Ultra Development Board (the selected device will be reflected under the Target Device).

🛿 New Project		>
Steps	Select Device	
1. Choose Project		
Select Device Select Header Select Plugin Board	Family: All Families	
 Select Compiler Select Project Name and Folder 	Device: PIC32C28110CA90208 V	
7. (Optional) Add Project	Tool: No Tool 🗸 🗋 Show All	
MPLAB X IDE		
< Back	Add Another Project Next > Finish (Cancel Help

- 7. Click Next.
- 8. Click Select Compiler, and in the right Select Compiler property page, under Compiler Toolchains click and expand XC32, and then select XC32 (v4.45).

Figure 2-3. Select Compiler

Steps	Select Compiler	×
Choose Project Select Device Select Needer Select Yangen Board Select Yangen Board Select Compiler Select Compiler Select Yangen Name and Folder (Optional) Add Project	Complet Toolchains Complet Toolchains TC32 (Deveload Latest) TC32 (4-45) [c1 Program Files (Mcrochip (xc32))/4-45(bin)] -Xc32 (V4-40) [c1 Program Files (Mcrochip (xc32))/4-45(bin)] C-ARM	
MPLAB X IDE		
< Back	Add Another Project Next > Einish Cance	el Help

- 9. Click Next.
- 10. Click **Select Project Name and Folder** and in the right **Select Project Name and Folder** property page:
 - a. **Project Name**: Enter *pic32cz_ca90_cult* (Indicates the name of the project that will be shown in MPLAB X IDE to set the project's name).



- b. **Project Location**: Enter *C*:*microchip**h3**tech_brief**dma_support**firmware* (Indicates the path to the root folder of the new project. All project files will be placed in this folder. The project location can be any valid path).
- c. **Project Folder**: Read-only content (Automatically updates when users make changes to the above entries).

Figure 2-4.	Project Name	e and Folder Setting	ζS
-------------	--------------	----------------------	----

eps	Select Project Name	and Folder		
Choose Project Select Device Select Header	Project Name:	pic32cz_ca90_cult	1	
Select Plugin Board Select Compiler	Project Location:	C:\microchip\h3\tech_brief\dma_support\firmware	Browse	
Select Project Name and Folder	Project Folder:	tech_brief\dma_support\firmware\pic32cz_ca90_cult.X		
(Optional) Add Project	Overwrite existin	ig project.	de la companya de la	
	Also delete source	ies.		
	ᠵ Set as main proje	ect		
	🖵 🛃 Open MCC	on Finish		
	Use project locat	ion as the project folder		
MPLAB	Encoding: IS	0-8859-1		
XIDE				
~				

- 11. Click **Finish** to launch MCC.
- 12. The MCC plugin will open in a new window as shown in the following figure:

Figure 2-5. MPLAB Code Configurator Window

File Edit View Navigate Source Refactor Production Debug Team Tools Window Help	Q · Search (Ctrl+I)
🕙 🚰 🚆 🖏 👘 🐢 🕼 Kanak 🤍 📓 + 📓 + 📓 + 📓 + 💺 + 🍢 + 🍡 🖏 📓 + 🧊 + 🚵 + 🏹 + 🍩 💷 🤓 Kanak Heranak P (enversion	
	Configuration Options ×
MCC v5.5.1 X □ ■ ● Plugins ∨ Profiles: Main ∨ View: Root	
Project ResourceGenerate importExport@ * Ubrainet * Ubrainet * Harmory * System CMSIS Pack Perspharal Library	System Device & Project Configuration Configuration Ports Ports Ports Configuration Configuration
e ● System Project Resources	Homenuples (MVIC) Group (MVIC) Group (MVIC) WOT Broc
Device Resources Content Manager	
▼ Libraries	
• Hannovy Project Graph	Configuration Options
Board Support Packages (BSPs)	
► Core	
Drivers	
► © Libraries	
Peripherals	
► System Services	
Third Party Libraries	
► Tools	
Device Resources	
Output x Hews Hotifications [HCC]	
CiG2cz_cs90_ctilt - Dashboard x Navigator DBG x Kits x Scripting x MPLA8@ Code Configurator x	
<pre>%% %pr2xr_ca00_ch Loading script file C:\Users\177539\.mchp_packs\Microchip\PfC3222-CA90_DFP\1.5.162\CA90\scripts\properties_device.py %% %pr2xr_ca00_ch Loading script file C:\Users\177539\.mchp_packs\Microchip\PfC3222-CA90_DFP\1.5.162\CA90\scripts\properties_device.py %% %pr2xr_ca00_ch Loading script file C:\Users\177539\.mchp_packs\Microchip\PfC3222-CA90_DFP\1.5.162\CA90\scripts\properties_device.py %% %pr2xr_ca00_ch %% %pr2xr_ca00_c</pre>	
Contracting C	
л	3



2.2 Creating Demo Application Using SAM E70 Xplained Ultra Evaluation Kit

To create an MPLAB Harmony v3-based project, follow these steps:

- 1. From the **Start** menu launch MPLAB X IDE.
- 2. On the File menu, click New Project or click on the New Project icon.
- 3. The **New Project** window will be displayed. From the **Steps** navigation pane, click **Choose Project**.
- 4. In the right **Choose Project** property page:
 - a. Categories select Microchip Embedded.
 - b. Projects select **Application Project(s)**.
- 5. Click Next.

Figure 2-6. New Project Window

Steps	Choose Project	
L. Choose Project	Q Filter:	
	Categories: Projects: Microchip Embedded Other Embedded Generic User Makefile Project User Makefile Project	able Image) Project
	Description:	
	Creates a new application project. It uses an IDE-generated makefile	to build your project.

6. Click **Select Device**, and in the right **Select Device** property page, for **Device** select **ATSAME70Q21B** for creating the project on the SAM E70 Xplained Ultra Evaluation Kit (the device entry will be reflected under the Target Device).



Figure 2-7. Device Selection

🔀 New Project					×
Steps	Select Device	•			
1. Choose Project 2. Select Device					
Select Device Select Header Select Plugin Board	Family:	All Families	~		
5. Select Compiler 6. Select Project Name and	Device:	ATSAME70Q21B	~		
Folder 7. (Optional) Add Project	Tool:	No Tool	→ Show	w All	
MPLAB					
XIDE					
< Back Add A	nother Project	Next >	Finish	Cancel	Help

- 7. Click Next.
- 8. Click **Select Compiler**, and in the right **Select Compiler** property page, under Compiler Toolchains click and expand **XC32** and then select **XC32** (v4.45).

Figure 2-8. Selecting Compiler

Steps	Select Compiler	×
Choose Project Select Device Select Header Select Young Doord Select Project Name and Folder (Optional) Add Project	Compiler Toolchains Compiler Toolchains CC22 (v4.45) [C: [Program Files (Mcrochip (vc32 (v4.45)bin]] - Xc32 (v4.40) [C: [Program Files (Mcrochip (vc32 (v4.40)bin]] - Xc32 (v4.40) [C: [Program Files (Mcrochip (vc32 (v4.40)bin]]	
MPLAB X IDE		
< gack A	Idd Another Project Next > Enish Cancel	Help

- 9. Click Next.
- 10. Click **Select Project Name and Folder**, and in the right **Select Project Name and Folder** property page:
 - a. **Project Name**: Enter *sam_e70_xult* (Indicates the name of the project that will be shown in MPLAB X IDE to set the project's name).
 - b. **Project Location**: Enter *C:\microchip\h3\tech_brief\dma_support\firmware* (Indicates the path to the root folder of the new project. All project files will be placed in this folder. The project location can be any valid path).



c. **Project Folder**: Read-only content (Automatically updates when users make changes to the above entries).

Figure 2-9. Project Name and Folder Settings

teps	Select Project Name	e and Folder
Choose Project		
Select Device	Project Name:	sam_e70_xult
Select Header Select Plugin Board		
Select Compiler	Project Location:	C:\microchip\h3\tech_brief\dma_support\firmware Browse
Select Project Name and	Project Folder:	\h3\tech_brief\dma_support\firmware\sam_e70_xult.X
Folder (Optional) Add Project		
(optional) Hour roject	Overwrite existin	ng project.
	Also delete source	ces.
	🔽 Set as main proje	ect
	- Open MCC	an Einich
	Use project locat	tion as the project folder
MPLAR		
INT LAD	Encoding: IS	O-8859-1 🗸
XIDE		
14 25		
< Back Add A	nother Project	Next > Finish Cancel Help

- 11. Click **Finish** to launch the MCC.
- 12. The MCC plugin will open in a new window as shown in the following figure:

Figure 2-10. MPLAB Code Configurator Window

File Edit View Navigate Source Refactor Production Debug Team			Q Search (Ctrl+I)
🕾 🚰 📲 🦏 🦃 🦿 default 🤍 🕍 -	🛍 + 🐻 + 🕨 + 🏪 + 🏪 + 🍓 👪 + 🔯 + 📾 + 🛒 + 🥮 💷 🤓 🚾 🚾 How do I? (cryword(s)		
Projects Files Resource Management (MCC) ×	Kit Window 🗴 Start Page 🗴 🛒 MPLAB X Store 🗴 Project Graph 🗴		Configuration Options ×
MCC v5.5.1	🗶 🗔 🖩 🖻 🚔 🔶 Plugins: 🔍 Profiles: Main 🗸 Vent	Root v	- •
Project Resourc Gener Impo Exp 😮 🌐			⊡-System
▼ Libraries	Device Family Pack (DFP) System CMSIS Pack EFC		Device & Project Configuration
 Harmony 	Device Family Pack (DFP) System CMSIS Pack EFC Peripheral Library		Cortex-M7 Configuration
▼ System	MEMORY		Clock (PMC)
😗 🗢 System			Ports (PIO)
			Interrupts (NVIC)
			E DMA (XDMAC)
Project Resources			B-RSWDT
riojeet neoduleeo			⊕-WDT
Device Resources			
Libraries	Project Graph		Configuration Options
	Floject Glaph		Configuration Options
 Harmony 			
 Board Support Packages (BSPs) 			
Core Drivers			
Libraries			
 Peripherals 			
System Services			
Third Party Libraries Tools			
► 100IS			
Device Resources			
Device nesources	Variables Output × News Notifications [HCC]		
	EDBG × Kits × Scripting × MPLAB@ Code Configurator × Configuration Loading Error × EDBG-sam_e70_xult × sam_e70_xult (Buld, Load,) × Debug	gger Console ×	
	Loading script file C:\Users\I77539\.mchp packs\Microchip\SAME70 DFP\4.11.257\same70b\\scripts\dap cortex-m7.	.pv	
sam_e70_xult - Dashboa × RX_DMA_EventHandler(X 🖬	Loading script file C:\Users\I77539\.mchp packs\Microchip\SAME70 DFP\4.11.257\same70b\\scripts\properties_dev		
Sg sam_e70_xult Project Type: Application - Configuration: default	Loading script file C:\Program Files\Microchip\MPLABX\v6.20\packs\Microchip\SAME70_DFP\4.10.248\same70b\\scri		
😪 🗄 🧼 Device	Loading script file C:\Program Files\Microchip\MPLABX\v6.20\packs\Microchip\SAME70_DFP\4.10.248\same70b\\scri	ipts\properties_devic	e.py
ATSAME70Q218 Gecksum: Blank, no code loaded	December 18 2024 10:31:45		
CRC32: Hex file unavailable	Begin comm session Loading script file C:\Users\177539\.mchp packs\Microchip\SAME70 DPP\4.11.2		
Packs Packs	Loading script file C:\Users\I77539\.mchp packs\Microchip\SAME70 DFP\4.11.257\same70b\\scripts\properties dev		
-			
ED6G (1.6.762)			
Gompler Toolchain With the state of the state o			
			(4)



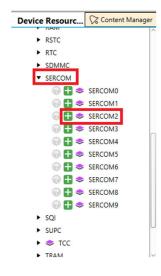
3. Adding and Configuring MPLAB Harmony Components

3.1 PIC32CZ CA90 Curiosity Ultra Development Board

To add and configure MPLAB Harmony components using MCC, follow these steps:

- 1. In the MCC window, under Device Resources, click and expand the list of options *Harmony* > *Peripherals* > *SERCOM.*
- 2. Click **SERCOM2** and observe that the SERCOM2 Peripheral Library block is added in the Project Graph section.

Figure 3-1. SERCOM Module



3. Click **SERCOM2** and in the **Configuration Options** property page, click and expand **SERCOM2** and configure SERCOM2 module as SPI Master as shown below.

Figure 3-2. SERCOM2 Configuration

Profiles: Main View: Root View:	
Homes. Main V View. Root V = -	
□ SERCOM2	
Pack (DFP) System C Select SERCOM Operation Mode SPI Master	
Enable Interrupts ?	
Enable operation in Standby mode	
SERCOM2 SPI Data Out Pad DO on PAD[0], SCK on PAD[1] and S	
Peripheral Library SPI Data In Pad Selection SERCOM PAD3 is used as data input	\sim
SPI Data Order MSB is transmitted first V	
SPI Speed in Hz 1,000,000 文	
UART SPI Data Character Size 8-bits character V	
	edge and change
- SPI Clock Polarity SCK is low when ide 🗸	
- Enable SPI Master Hardware Slave Select <table-cell></table-cell>	
— SPI Receiver Enable 🛛 🗹	
****SPI Transfer Mode 0 is Selected***	



4. From the **Plugins** drop-down list, select **Pin Configuration**.

Figure 3-3. Plugins - Pin Configuration

	NVIC Configuration Event Configurator	
EVSYS Peripheral Library	Device DMA Configuration MPU Configurator Pin Configuration Clock Configuration	System C
		SERCOM2 Peripheral Library I2C
		SPI UART

5. Click **Pin Settings** and then sort entries by selecting **Ports** from the Order list.

Figure 3-4. Selecting Ports from the Order Menu

ow Start Page 🗙 🛒 N	IPLAB X Store ×	Project Gra	oh x	Pin Diagran	n ×	Pin Table	×	Pin Settings	×
Order: Pins 🗸	Table View	🔽 Easy Vi	ew						
Pins Pin Num	Custom	Name	Fu	nction	Mod	de	Dire	ection	Latch

6. Configure the pins as shown below:

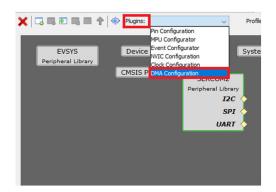
Figure 3-5. Configuration of Pins

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Open Drain	Slew Ra	te
V15	PB31		Available	Digital	High Impedance $$	Low				FAST	\sim
P15	PC00		Available	/ Digital	High Impedance $$	Low				FAST	\sim
T17	PC01		Available	Digital	High Impedance $$	Low				FAST	\sim
T18	PC02		Available	Digital	High Impedance $$	Low				FAST	\sim
R17	PC03		Available	/ Digital	High Impedance $$	Low				FAST	\sim
R 18	PC04		Available	Digital	High Impedance $$	Low				FAST	\sim
N15	PC05		Available	Digital	High Impedance $$	Low				FAST	\sim
N17	PC06		Available	Digital	High Impedance $$	Low				FAST	\sim
N18	PC07		Available	Digital	High Impedance 🗸	Low				FAST	\sim
M13	PC08		SERCOM2_PAD0	 Digital 	High Impedance \lor	n/a				FAST	\sim
M18	PC09		SERCOM2_PAD1	Digital	High Impedance \lor	n/a				FAST	\sim
H17	PC10		SERCOM2_PAD2	/ Digital	High Impedance \lor	n/a				FAST	\sim
H15	PC11		SERCOM2_PAD3	 Digital 	High Impedance \lor	n/a				FAST	\sim
F18	PC12		Available	 Digital 	High Impedance \lor	Low				FAST	\sim
F17	PC13		Available	Digital	High Impedance $$	Low				FAST	\sim
E17	PC14		Available	/ Digital	High Impedance $$	Low				FAST	~

7. In the **Plugins** drop-down list, select **DMA Configuration**.



Figure 3-6. Plugins - DMA Configuration



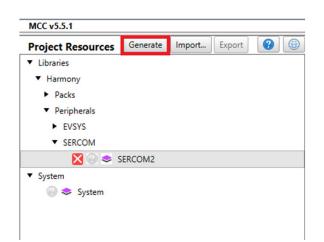
8. In the DMA Configuration, click **Add Channel** to add DMAC Channel 0 and choose the trigger as SERCOM2_Transmit. Repeat the same steps, but for DMAC Channel 1 choose SERCOM2_Receive as the trigger.

Figure 3-7. DMA Configuration

Activ	e Channels List	Use Linked List Mode		
Channel Number	Trigger		OMA Channel 0 Settings	
DMAC Channel 0	SERCOM2_Transmit V	Trigger Action (Cell Auto Start Enable)	One Cell Transfer Per DMA Start Trigger	~
DMAC Channel 1	SERCOM2_Receive V	Read Address Sequence	Incrementing Address+1 with Transfers of Byte Operands	~
	Remove Selected Channel	Write Address Sequence	Fixed Byte Address (Single Byte Address with Enable Based u	~
Add Channel	Remove Selected Channel	Cell Transfer Size	1	\$
		Channel Priority Level	Priority Level 1	~

9. Click **Generate** to generate the code.

Figure 3-8. Generating the Code



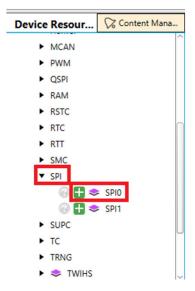


3.2 SAM E70 Xplained Ultra Evaluation Kit

To add and configure MPLAB Harmony components using MCC, follow these steps:

- 1. In the MCC window, under Device Resources, click and expand the list of options *Harmony* > *Peripherals* > *SPI.*
- 2. Click **SPIO** and observe that the SPIO Peripheral Library block is added in the Project Graph section.

Figure 3-9. SPI Module



3. Click SPI0, and to configure SPI0 module, click and expand SP10 and then configure it as shown below.

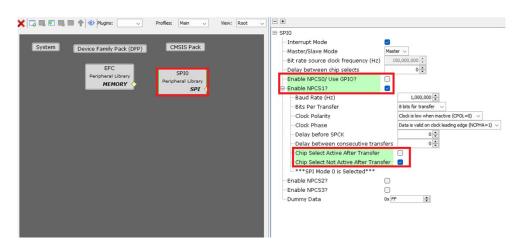
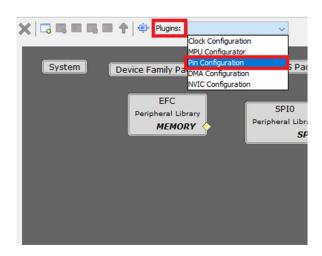


Figure 3-10. SPIO Configuration

4. From the **Plugins** drop-down list, select **Pin Configuration**.



Figure 3-11. Plugins - Pin Configuration



5. From the **Order** drop-down list, select **Ports** to sort the entries.

Figure 3-12. Selecting Ports from the Order Menu

Order:		~	Table View	🛃 Easy \	liew						
Pin Num	Pins Ports		Custom N	lame	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength

6. From the **Plugins** drop-down list, select **DMA Configuration**. In the **DMA Configuration** property page, click **Add Channel** to add DMAC Channel 0, and set the trigger to SPI0_Transmit. Repeat the same steps, but for DMAC Channel 1, set the trigger to SPI0_Receive.

Figure 3-13. Plugins - DMA Configuration

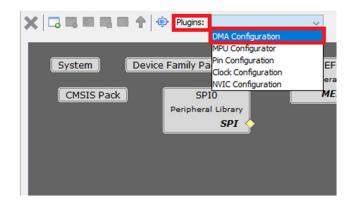




Figure 3-14. DMA Configuration

Active	Channels List	Use Linked List Mode		
Channel Number	Trigger	DM	IA Channel 0 Settings	
DMAC Channel 0	SPI0_Transmit 🗸	Enable Interrupt		
DMAC Channel 1	SPI0_Receive V	Source Addressing Mode	Increment Address After Every Transfer	~
binke endiner i	Jing Receive	Destination Addressing Mode	Fixed Address Mode	~
Add Channel	Remove Selected Channel	DMA Interface Bus To Read Source Data	DMA Interface Bus 0	~
		DMA Interface Bus To Write Destination Data	DMA Interface Bus 1	~
		Data Width	8-Bits	~
		Data Transfers Per DMA Request	1 Transfer Per Request	~
		Burst Size For Memory To Memory Transfer	1 Transfer Per Burst	~

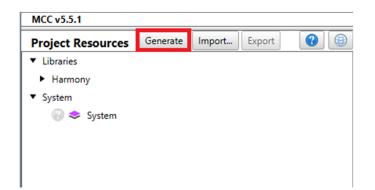
7. Configure the pins as shown below.

Figure 3-15. Configuration of Pins

Order:	Ports	V Table View												
Pin Number	Pin ID	Custom Name	Function		Direction	Latch	Open Drain	PIO Inter	rupt	Pull Up	Pull Down	Glitch/Debou Filter	ince	Drive
74	PD17		Available	~	In	n/a		Disabled	~			Disabled	~	Low
69	PD18		Available	~	In	n/a		Disabled	~			Disabled	~	Low
67	PD 19		Available	~	In	n/a		Disabled	~		0	Disabled	~	Low
65	PD20		SPI0_MISO	\sim	n/a	n/a		Disabled	~			Disabled	~	Low
63	PD21		SPI0_MOSI	~	n/a	n/a		Disabled	~			Disabled	~	Low
60	PD22		SPI0_SPCK	~	n/a	n/a		Disabled	~		0	Disabled	~	Low
57	PD23		Available	~	In	n/a		Disabled	~			Disabled	~	Low
55	PD24		Available	\sim	In	n/a		Disabled	~			Disabled	~	Low
52	PD25		SPI0_NPCS1	\sim	n/a	n/a		Disabled	~			Disabled	~	Low
53	PD26		Available	\sim	In	n/a		Disabled	~			Disabled	~	Low
47	PD27		Available	~	In	n/a		Disabled	~			Disabled	~	Low
71	PD28		Available	~	In	n/a		Disabled	~			Disabled	~	Low
108	PD29		Available	~	In	n/a		Disabled	~			Disabled	~	Low
34	PD30		Available	~	In	n/a		Disabled	~			Disabled	~	Low
2	PD31		Available	\sim	In	n/a		Disabled	~			Disabled	~	Low
4	PEO		Available	~	In	n/a		Disabled	~			Disabled	~	Low
6	PE1		Available	~	In	n/a		Disabled	~			Disabled	~	Low
7	PE2		Available	\sim	In	n/a		Disabled	~			Disabled	~	Low
10	PE3		Available	~	In	n/a		Disabled	~			Disabled	~	Low
27	PE4		Available	~	In	n/a		Disabled	~			Disabled	~	Low
28	PE5		Available	~	In	n/a		Disabled	~			Disabled	~	Low
3	VDDOUT			~	In	n/a	0		~				~	Low
5	VDDIN			~	In	n/a		-	~				~	Low

8. Click **Generate** to generate the code.

Figure 3-16. Generating the Project





4. Adding Application Logic to the Project

4.1 PIC32CZ CA90 Curiosity Ultra Development Board

To develop and run the application, follow these steps:

1. Click **Projects** and then under Source files open the main.c file of the project and add the required variables outside the main() function.

```
uint8_t__attribute__ ((aligned(32))) Tx[10] = {0x1F, 0x2F, 0x3F, 0x4F, 0x5F, 0x6F, 0x7F,
0x8F, 0x9F, 0xAF};
uint8_t__attribute__ ((aligned(32)))Rx[10];
/* transfer done flag */
volatile bool Rx_transfer_done = false;
volatile bool Tx_transfer_done = false;
```

2. Add the DMA Event Handler for both Tx and Rx outside the ${\tt main}$ () function.

```
/* This is called after transfer is done */
void Tx_DMA_EventHandler(DMA_TRANSFER_EVENT event, uintptr_t context)
{
    if (event == DMA_TRANSFER_EVENT_BLOCK_TRANSFER_COMPLETE)
    {
        Tx_transfer_done = true;
    }
}
void Rx_DMA_EventHandler(DMA_TRANSFER_EVENT event, uintptr_t context)
{
    if (event == DMA_TRANSFER_EVENT_BLOCK_TRANSFER_COMPLETE)
    {
        Rx_transfer_done = true;
    }
}
```

Figure 4-1. Adding Macros, Variables, and Event Handlers





3. Add the DMA Callback register function, cache invalidate function, and DMA Channel transfer function.

```
DMA_ChannelCallbackRegister(DMA_CHANNEL_0, Tx_DMA_EventHandler, (uintptr_t)NULL);
DMA_ChannelCallbackRegister(DMA_CHANNEL_1, Rx_DMA_EventHandler, (uintptr_t)NULL);
DCACHE_INVALIDATE_BY_ADDR((uint32_t *)Tx, 10);
DCACHE_INVALIDATE_BY_ADDR((uint32_t *)Rx, 10);
DMA_ChannelTransfer(DMA_CHANNEL_1, (void *)&SERCOM2_REGS->SPIM.SERCOM_DATA, Rx,
sizeof(Rx));
DMA_ChannelTransfer(DMA_CHANNEL_0, Tx, (void *)&SERCOM2_REGS->SPIM.SERCOM_DATA, sizeof(Tx));
```

Note: Cache invalidation is crucial for maintaining data consistency and accuracy, especially in dynamic systems where data frequently changes. By invalidating the cache, it ensures that the next time the data is requested, it will be fetched from the original source rather than the cache.

Figure 4-2. Application Logic

61		int	main (void)
62	Ę	{	
63			/* Initialize all modules */
64			SYS Initialize (NULL);
65			
66			DMA ChannelCallbackRegister(DMA CHANNEL 0, TX DMA EventHandler, (uintptr t)NULL);
67			DMA ChannelCallbackRegister(DMA CHANNEL 1, RX DMA EventHandler, (uintptr t)NULL);
68			
69			DCACHE INVALIDATE BY ADDR((uint32 t *)tx, 10);
70			DCACHE INVALIDATE BY ADDR((uint32 t *)rx, 10);
71			
72			DMA ChannelTransfer(DMA CHANNEL 1, (void *)&SERCOM2 REGS->SPIM.SERCOM DATA, rx, sizeof(rx))
73			DMA ChannelTransfer(DMA CHANNEL 0, tx, (void *)&SERCOM2 REGS-SSPIM.SERCOM DATA, fx, Sizeof(tx))
74			DEA_CHAINETTIAISTET(DEA_CHARNEL_0, CX, (VOId -)&SERCOR2_REGS-SEFER.SERCOR_DATA, SIZEOT(CX))
75			while (true)
76			white (true)
77			/* Maintain state machines of all polled MPLAB Harmony modules. */
78			SYS_Tasks ();
79			3
80			
81			/* Execution should not come here during normal operation */
82			
83			return (EXIT_FAILURE);
84	L	3	

4.2 SAM E70 Xplained Ultra Evaluation Kit

To develop and run the application, follow these steps:

1. Open the main.c file of the project and add the required variables outside the main () function.

```
uint8_t __attribute__ ((aligned(32))) Tx[10] = {0x1F, 0x2F, 0x3F, 0x4F, 0x5F, 0x6F, 0x7F,
0x8F, 0x9F, 0xAF};
uint8_t __attribute__ ((aligned(32)))Rx[10];
/* transfer done flag */
volatile bool Rx_transfer_done = false;
volatile bool Tx_transfer_done = false;
```

2. Add the DMA Event Handler for both Tx and Rx outside of the main () function.

```
/* This is called after transfer is done */
void Tx_DMA_EventHandler(XDMAC_TRANSFER_EVENT event, uintptr_t context)
{
    if (event == XDMAC_TRANSFER_COMPLETE)
    {
        Tx_transfer_done = true;
    }
}
void Rx_DMA_EventHandler(XDMAC_TRANSFER_EVENT event, uintptr_t context)
{
    if (event == XDMAC_TRANSFER_COMPLETE)
    {
        Rx_transfer_done = true;
    }
}
```



Figure 4-3. Adding Macros, Variables, and Event Handlers



3. Add the DMA Callback register function, cache invalidate function, and DMA Channel transfer function.

```
XDMAC_ChannelCallbackRegister(XDMAC_CHANNEL_0, Tx_DMA_EventHandler, (uintptr_t)NULL);
XDMAC_ChannelCallbackRegister(XDMAC_CHANNEL_1, Rx_DMA_EventHandler, (uintptr_t)NULL);
DCACHE_INVALIDATE_BY_ADDR((uint32_t *)Tx, 10);
DCACHE_INVALIDATE_BY_ADDR((uint32_t *)Rx, 10);
XDMAC_ChannelTransfer(XDMAC_CHANNEL_1, (void *)&SPI0_REGS->SPI_RDR, Rx, sizeof(Rx));
XDMAC_ChannelTransfer(XDMAC_CHANNEL_0, Tx, (void *)&SPI0_REGS->SPI_TDR, sizeof(Tx));
```

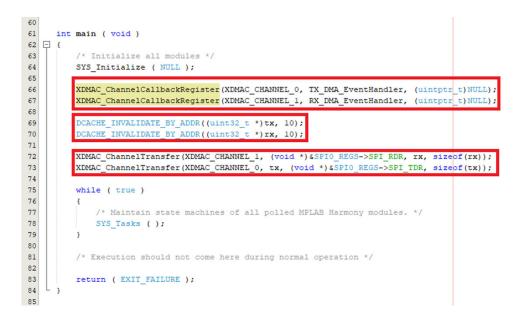


Figure 4-4. Application Logic



5. Building and Debugging the Application

1. The PIC32CZ CA90 Curiosity Ultra Development Board supports debugging using an Embedded Debugger (EDBG). Connect the Type-A male to micro-B USB cable to the micro-B USB port on the PIC32CZ CA90 Curiosity Ultra Development Board and connect the Type-A male end to the PC. Additionally, connect an external power supply (6.5V-14V) to power up the board.

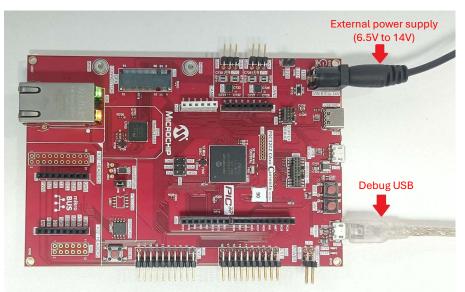


Figure 5-1. Hardware - PIC32CZ CA90 Curiosity Ultra Development Board

2. Short the MISO (PC11 – Pin 17) and MOSI (PC08 – Pin 16) pins present in EXT1 in the PIC32CZ CA90 Curiosity Ultra Development Board using a wire.

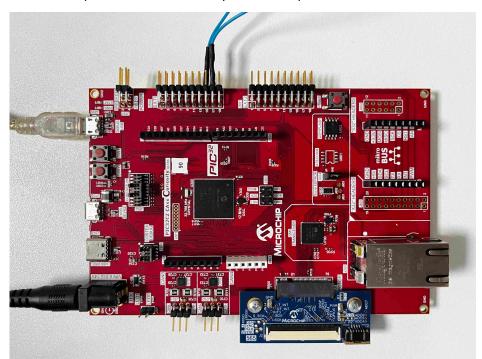


Figure 5-2. Hardware Setup - PIC32CZ CA90 Curiosity Ultra Development Board



3. Connect the Type-A male to the micro-B USB cable to the micro-B debug USB port to power and debug the SAM E70 Xplained Ultra Evaluation Kit.

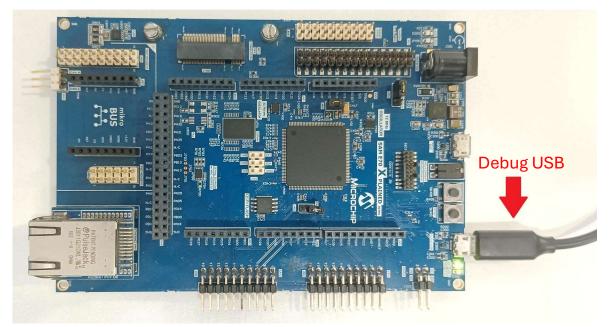


Figure 5-3. Hardware - SAM E70 Xplained Ultra Evaluation Kit

4. Short the MISO (PD20 – Pin 17) and MOSI (PD21 – Pin 16) pins present in EXT1 in the SAM E70 Xplained Ultra Evaluation Kit using a wire.



Figure 5-4. Hardware Setup - SAM E70 Xplained Ultra Evaluation Kit



5. Select the **Tx** variable in the code and right-click on the selected text, then select **New Watch**. **Note:** Follow the same for **Rx**.

Figure 5-5. Adding a New Watch



- 6. In the MPLAB X IDE Project Properties window perform these actions.
 - a. Under the left Categories section, select **Conf: [default]**, and in the right **Configuration** properties page, select the Connected Hardware Tool and Compiler Toolchain as shown below.

Figure 5-6. Project Properties - PIC32CZ CA90 Curiosity Ultra Development Board

Family:	Device:
All Famílies 🗸 🗸	PIC32CZ8110CA90208 ~
Connected Hardware Tool:	Supported Debug Header:
No Tool Show All	None
No Tool	
Simulator	
PIC32CZ CA90 Curiosity Ultra -SN: 020061504RYN000045	
5.4.0	1
5.8.0	
E-PIC32CZ-CA90_DFP	
- 3.152	1
XC32 (v4.40) [C: \Program Files \Microchip \xc32 \v4.40 \bin]	
	All Families Connected Hardware Tool: No Tool Show All Simulator PIC322CCA90 Curricety Litrar -54: 02006150-#RYN000045 PIC322CCA90 Curricety Litrar -54: 02006150-#RYN000045 PIC322CCA90 Curricety Litrar -54: 02006150-#RYN000045 PIC322CCA90 DFP PIC322CCA90 Curricety Litrar -54: 02006150-#RYN000045 Complet Toolcham: XC322 (V4-430 [C4)*Forgam Fries/Morochig/vc32(V4-430 [bn] XC322 (V4-40) [C1*Forgam Fries/Morochig/vc32(V4-40 [bn]



7. For SAM E70 Xplained Ultra Evaluation Kit, the hardware tool needs to be changed as follows.

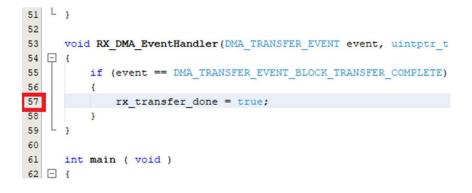
Figure 5-7. Project Properties - SAM E70 Xplained Ultra Evaluation Kit

gories:	Configuration	
 General 	Family:	Device:
File Inclusion/Exclusion	All Families V	PIC32CZ8110CA90208 ~
<u>Conf: [default]</u> O PKoB4	Connected Hardware Tool:	Supported Debug Header:
O Loading	No Tool	None
- O Libraries	No Tool	
Building XC32 (Global Options)	Simulator	
→ ○ XC32 (Global Options)	SAM E70 Xplained Ultra-SN: ATML3186071800001067	
- • xc32-gcc	5.4.0	1
- • xc32-g++	5.8.0	
- • xc32-ld	PIC32CZ-CA90_DFP	
— 🍳 xc32-ar	1.3.152	1
- O Analysis	Compiler Toolchain:	200
	Completer rook dans	
	XC32 (v4.45) [C:\Program Files\Microchip\xc32\v4.45\bin]	
	XC32 (v4.40) [C: Program Files Microchip (xc32 (v4.40)pin]	
	⊕ ARM	
Manage Configurations		
Manage Network Tools		

Note: The following steps are applicable for both the boards (the PIC32CZ CA90 Curiosity Ultra Development Board and the SAM E70 Xplained Ultra Evaluation Kit).

- 8. Click **Apply** and then click **OK**.
- 9. Click on the highlighted position as shown below to put a breakpoint. **Note:** Breakpoint is used in the reception transfer handler to visualize the received data.

Figure 5-8. Adding the Breakpoint in Reception Complete









10. Click on the **Debug main Project** button.

Figure 5-10. Debug Main Project

	Window						
SRC -	-	D	• 🖳	- 🚡	• 🖓	-	🔟 - 📄 - 🛒 - 🤹

Note: If the Variable window does not appear, go to *Window > Debugging > Variables* to open the Variable window.

Figure 5-11. Window Menu details

<u>Iools W</u> ir	ndow Help		
- 🐘 🚥	MCC Content Manager		- K- K) 🖄 🖄 🗿 🕗
- and	MPLAB® Code Configurator v5	i.	
Start P	Kit Window		×
/ 🖀 🛛 🖷	Projects	Ctrl+1	2 2 😐 💷 🖉
if (et 🥮	MPLAB Discover		FER_COMPLETE)
1	Files	Ctrl+2	
r> @	Classes	Ctrl+9	
1	Fa <u>v</u> orites	Ctrl+3	
-	Services	Ctrl+5	
main (%	Dashboard		
/* Ini 🛇	<u>N</u> avigator	Ctrl+7	
SYS Ir	Action Items	Ctrl+6	
	Tas <u>k</u> s	Ctrl+Shift+6	б
DMA_Cr	Output	Ctrl+4	TX_DMA_EventHandler, (uintptr_
DMA_CF	Editor	Ctrl+0	RX_DMA_EventHandler, (uintptr_
DCACHI	De <u>b</u> ugging		> Output >
DCACHE	<u>W</u> eb		> 😔 Variables Alt+Shift+1
	IDE <u>T</u> ools		Watches Alt+Shift+2
DMA_CF	Target Memory Views		Call Stack Alt+Shift+3
DMA_Ch	Simulator		Resknoints Alt+Shift+5

Figure 5-12. Variables Window

2	Name	Туре	Address	Value	1
	🗉 🛃 💮 rx; file:	/src/main. uint8_t[10]	0x20020080	"\u001f/?O_o \u008f	\ []
	🕀 🛃 💮 tx; file:	./src/main.uint8_t[10]	0x200200E0	"\u001f/?O_o \u008f	۱
Ð	Enter new	watch>			
		done; file: bool	0x2002008B	0x00	



11. The transmitted data (Tx) matches with the received data (Rx).

Figure 5-13. Output

1	Name	Туре		Address	Value		
E	🔽 💮 rx; file:.	./src uint8_t[10])	0x20020080		"\u001f/?O_o	\u
	<pre> rx[0] </pre>	uint8_t)	0x20020080		US; 0x1f	
	♦ rx[1]	uint8_t		0x20020081		'/; 0x2f	
	♦ rx[2]	uint8_t		0x20020082		'?'; 0x3f	
	♦ rx[3]	uint8_t		0x20020083		'O'; 0x4f	
		uint8_t		0x20020084		'_'; 0x5f	
	🔷 rx[5]	uint8_t		0x20020085		'o'; 0x6f	
	🛞 rx[6]	uint8_t		0x20020086		'?'; 0x7f	
		uint8_t		0x20020087		'?'; 0x8f	
	♦ rx[8]	uint8_t		0x20020088		'?'; 0x9f	
	🔷 rx[9]	uint8_t		0x20020089		'?'; 0xaf	
E	🛛 🔄 💮 tx; file:.	./src uint8_t[10])	0x200200E0		"\u001f/?O_o	\u
	🔷 tx[0]	uint8_t)	0x200200E0		US; 0x1f	
	🔷 tx[1]	uint8_t		0x200200E1		'/; 0x2f	
	🔷 tx[2]	uint8_t		0x200200E2		'?'; 0x3f	
	🔷 tx[3]	uint8_t		0x200200E3		'O'; 0x4f	
	🔷 tx[4]	uint8_t		0x200200E4		'_'; 0x5f	
	🔷 tx[5]	uint8_t		0x200200E5		'o'; 0x6f	
	🔷 tx[6]	uint8_t		0x200200E6		'?'; 0x7f	
	🔷 tx[7]	uint8_t		0x200200E7		'?'; 0x8f	
	🚸 tx[8]	uint8_t		0x200200E8		'?'; 0x9f	
	🔷 tx[9]	uint8_t		0x200200E9		'?'; 0xaf	



6. References

- PIC32CZ CA90 Curiosity Ultra Development Board
- SAM E70 Xplained Ultra Evaluation Kit
- PIC32CZ CA80/CA90 Curiosity Ultra User Guide (DS70005522)
- SAM E70 Xplained Ultra User Guide (DS70005389)
- Demonstrating Application Development with PIC32CZ CA90 Curiosity Ultra Evaluation Board
- Getting Started Extended Application on PIC32CZ CA90 Curiosity Ultra Development Board
- Create Your First Project with SAM E70 using MPLAB[®] Harmony v3
- Getting Started with MPLAB[®] Harmony v3 Drivers and System Services on SAM E70/S70/V70/V71 MCUs
- For additional information about 32-bit Microcontroller Collaterals and Solutions, refer to: 32-bit Microcontroller Collateral and Solutions Reference Guide (*DS70005534*)
- For additional information on MPLAB[®] Harmony v3, refer to the Microchip web site: https://www.microchip.com/en-us/tools-resources/configure/mplab-harmony and https:// developerhelp.microchip.com/xwiki/bin/view/software-tools/harmony/
- For more information on various applications, refer to: github.com/Microchip-MPLAB-Harmony/reference_apps
- For other relevant information, refer to the Microchip web site: www.microchip.com/



7. Revision History

7.1 Revision A - January 2025

This is the initial release of this document.



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: 979-8-3371-0498-0

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 products are 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.



Product Page Links

ATSAME51G18A, ATSAME51G19A, ATSAME51J18A, ATSAME51J19A, ATSAME51J20A, ATSAME51N19A, ATSAME51N20A, ATSAME53J18A, ATSAME53J19A, ATSAME53J20A, ATSAMS70J19, ATSAMS70J20, ATSAMS70J21, ATSAMS70N19, ATSAMS70N20, ATSAMS70N21, ATSAMS70Q19, ATSAMS70Q20, ATSAMS70Q21, ATSAMV70J19, ATSAMV70J20, ATSAMV70N19, ATSAMV70N20, ATSAMV70Q19, ATSAMV70Q20, ATSAMV71J19, ATSAMV71J20, ATSAMV71J21, ATSAMV71N19, PIC32CZ2051CA70064, PIC32CZ2051CA70100, PIC32CZ2051CA70144, PIC32CZ2051CA80100, PIC32CZ2051CA80144, PIC32CZ2051CA80176, PIC32CZ2051CA80208, PIC32CZ2051CA90100, PIC32CZ2051CA90144, PIC32CZ2051CA90176

