CAE專家教室 PCM1001系列 課程

PIC32CXSG ARM Cortex-M4F Microcontroller & MCC Harmony

PCM1001 v1.00

A Leading Provider of Smart, Connected and Secure Embedded Control Solutions

CAE Taiwan
Microchip Technology Inc.

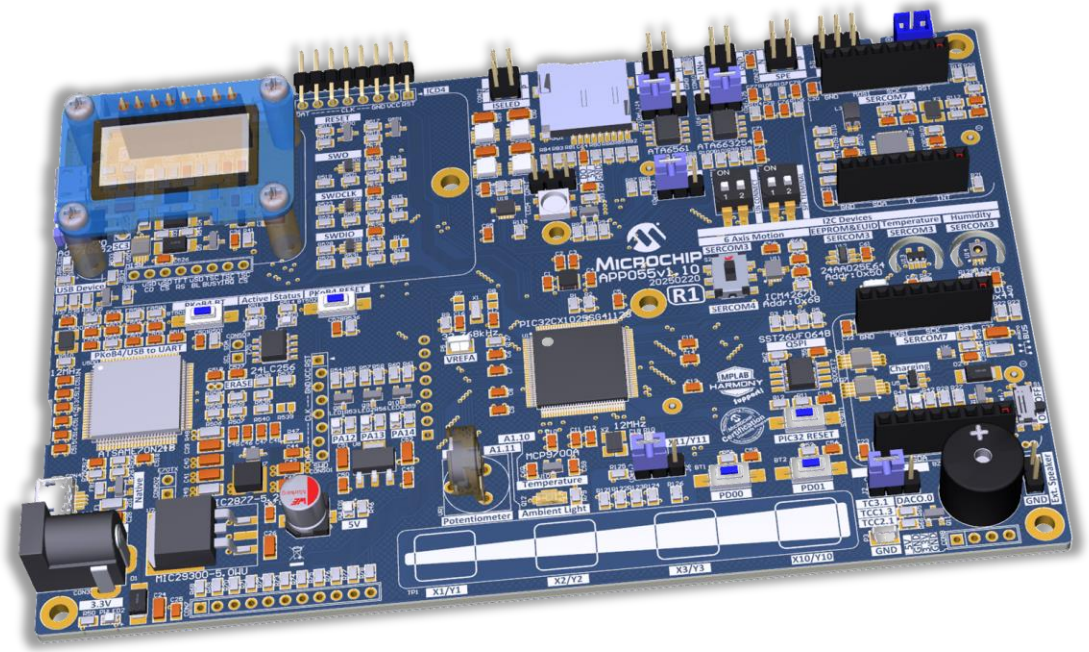
December 2025

Index (1/2)

PIC32CX-SG EVB
APP055 v1.10$_{R1}$ Introduction

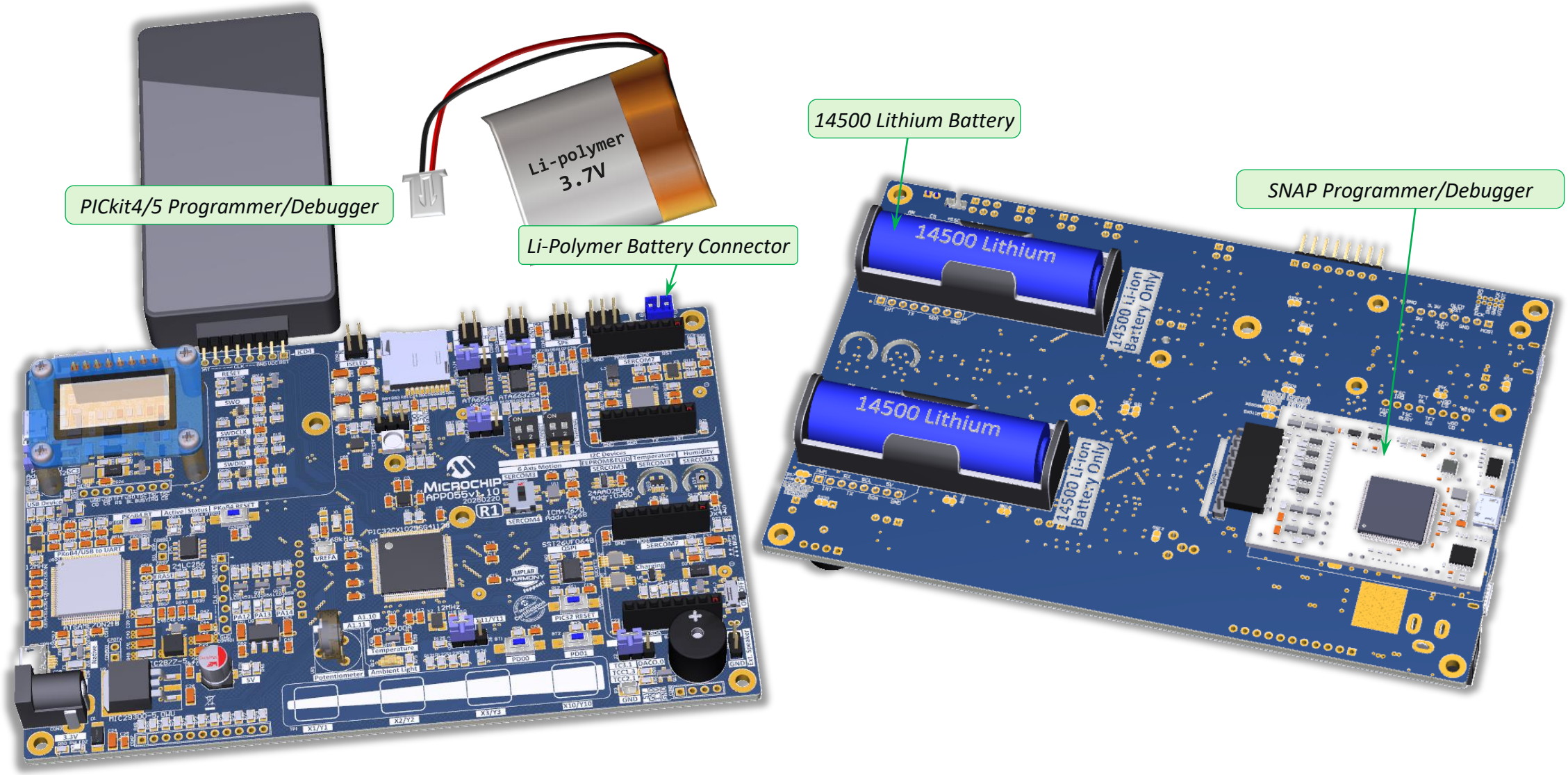
APP055 v1.10$_{R1}$ Features

- PIC32CX1025SG41128 32Bit Arm Cortex M4F Microcontroller with FPU.
- Built-in PKoB4 Programmer/Debugger with USB CDC Virtual COM.
- Multiple Power Source and support Li-on battery with battery charger.
- ISELED, Single Pair Ethernet, CAN-FD, LIN support ready.
- Capacitive QTouch® Slider & Buttons with Driven Shield.
- mikroBUS™ Click Board Socket ready.
- Digital IO : 2 x Buttons and 3 x LEDs and WS2812B.
- Analog : Potentiometer, Temperature Sensor and Ambient Light Sensor.
- I2C : Temperature, EEPROM, 6 Axis Motion and Humidity Sensor.
- SPI : OLED (128 x 64, SSD1306 Compliant)
- QSPI : Flash Memory.
- SDHC : MicroSD Socket.
- PWM, DAC : Buzzer.

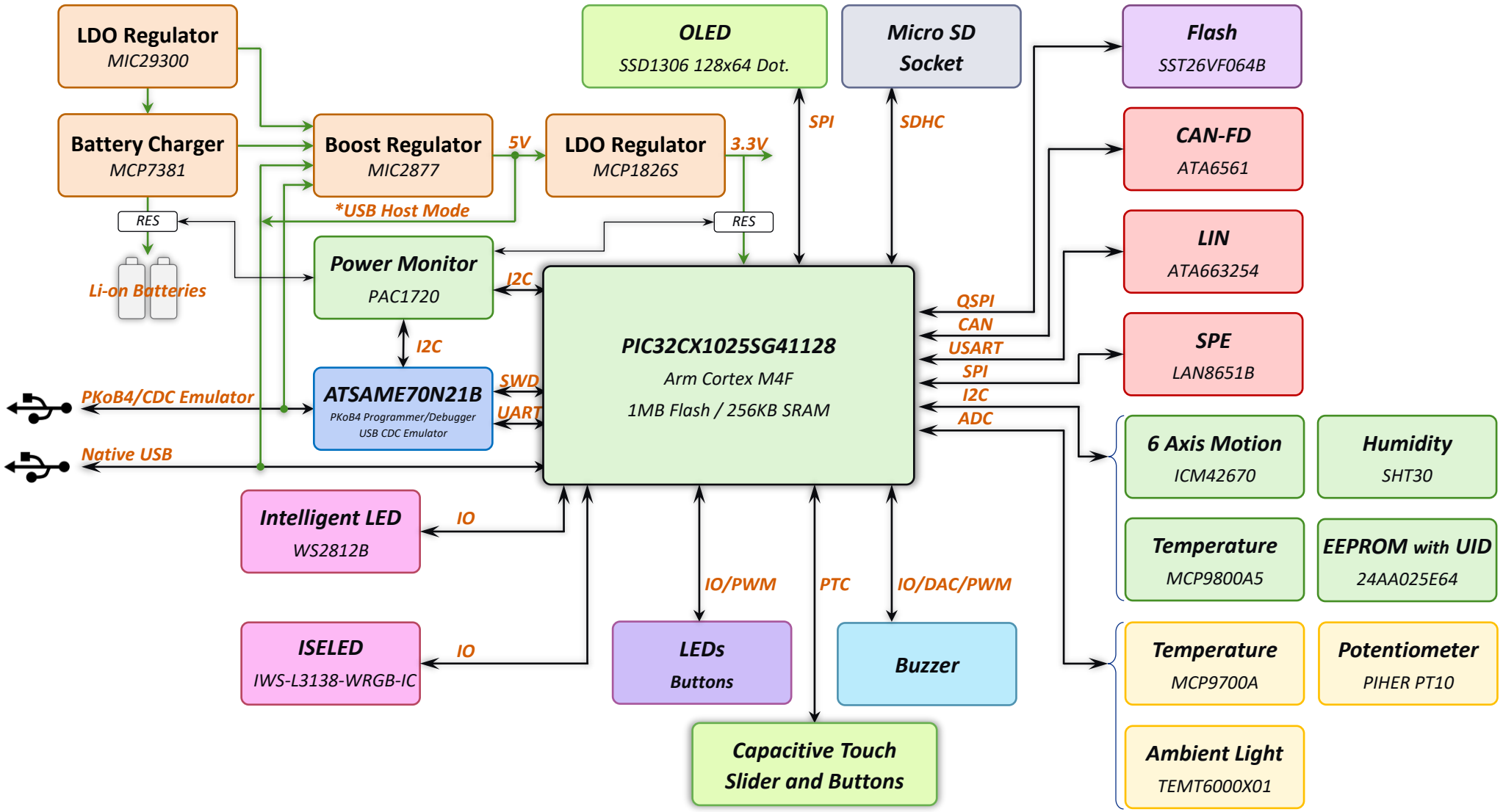


MICROCHIP

APP055 v1.10$_{R1}$ Accessories



APP055 v1.10$_{R1}$ Block Diagram



Getting Started with First Project

Try to create your first project to use MPLAB X IDE and MCC.

Getting Started with First Project

- Try to create your first project to use MPLAB X IDE and MCC.
- Learn how to use edit, build, project manager, debug and programming functions.

How to Start ?

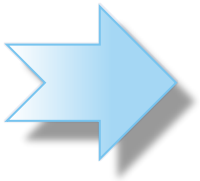
Getting Started with First Project

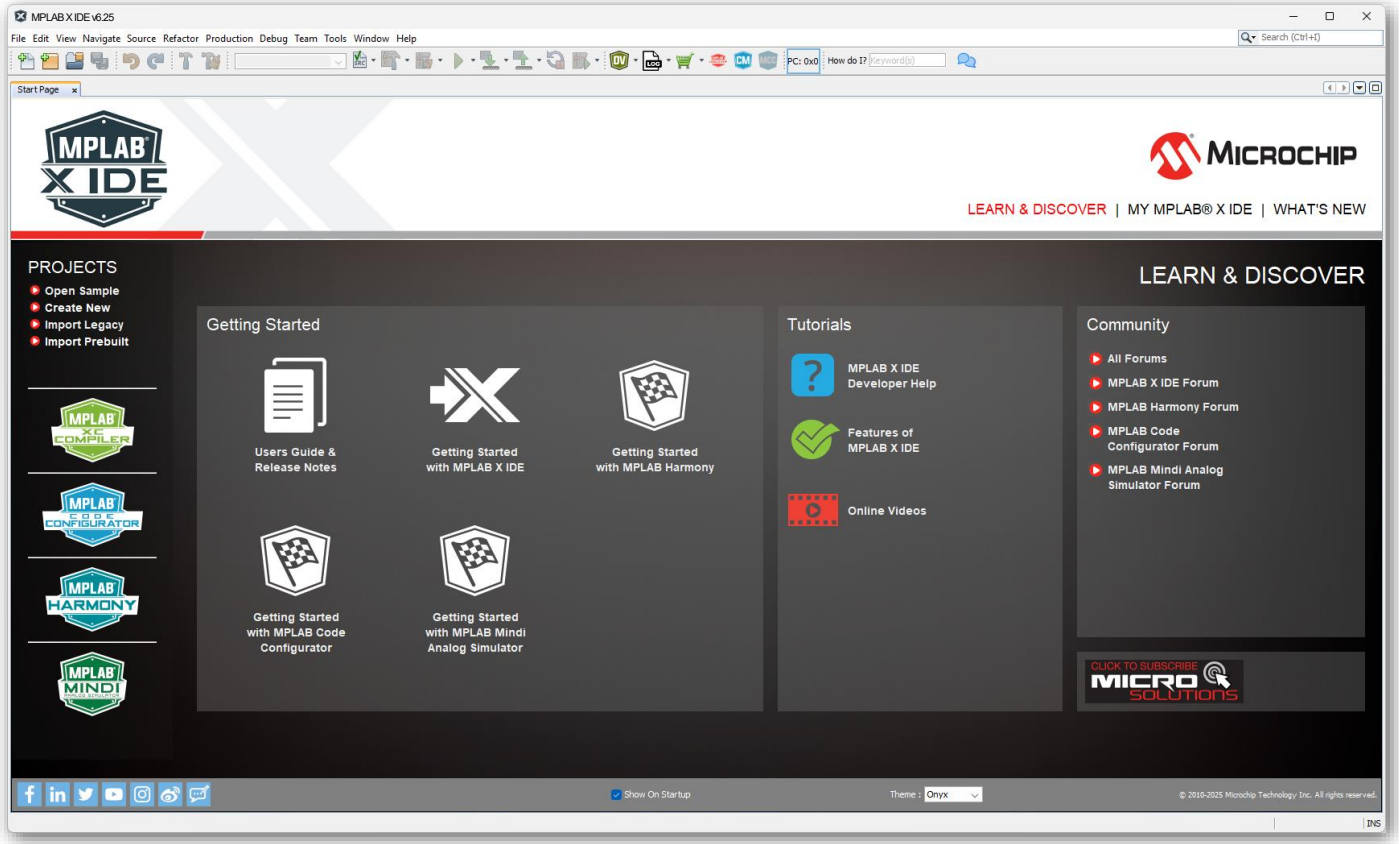
Step 1

- **Open MPLAB X IDE v6.25 firstly.**

MPLAB X IDE
v6.25

Double Click!



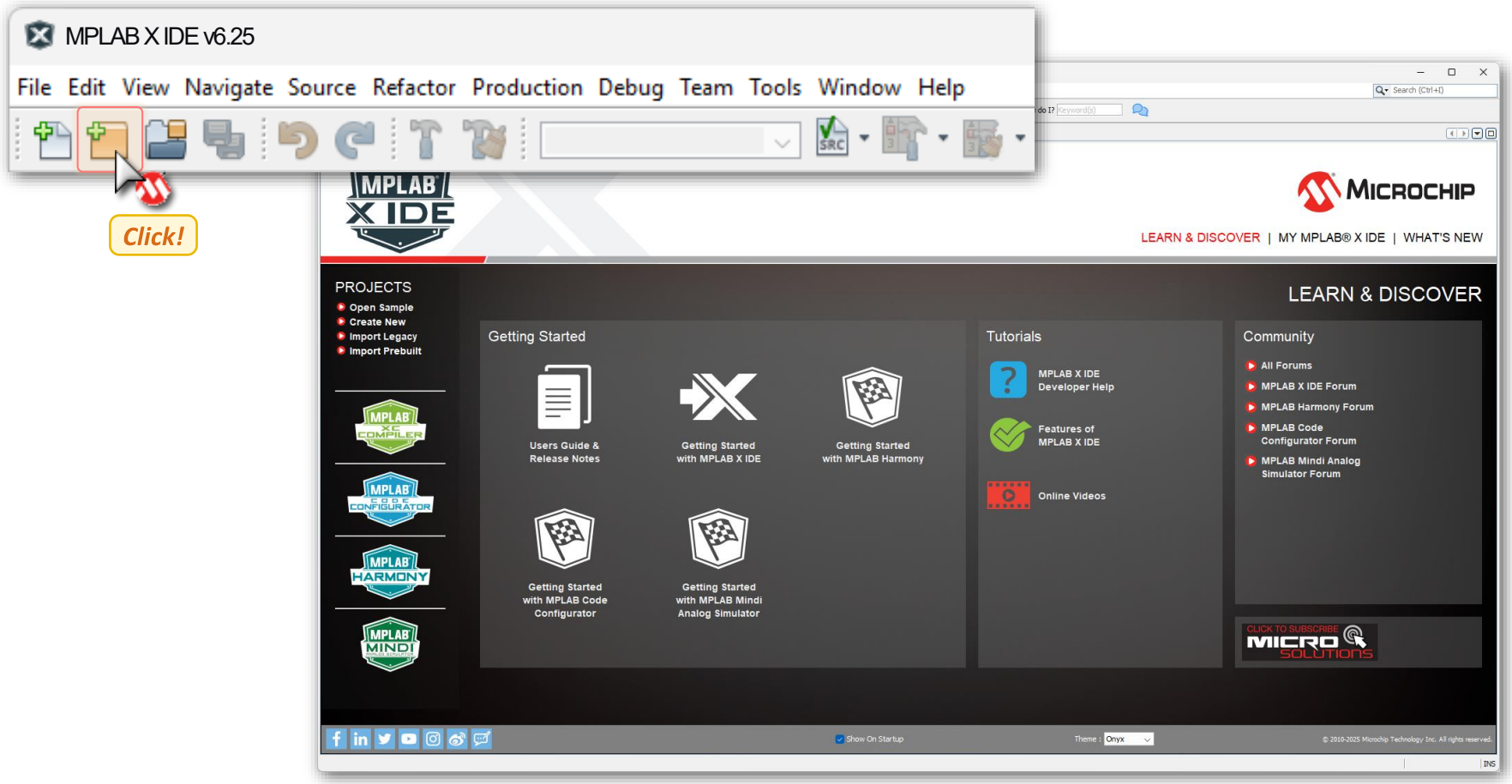


MICROCHIP

Getting Started with First Project

Step 2

- **Select to File > New Project or Click icon**

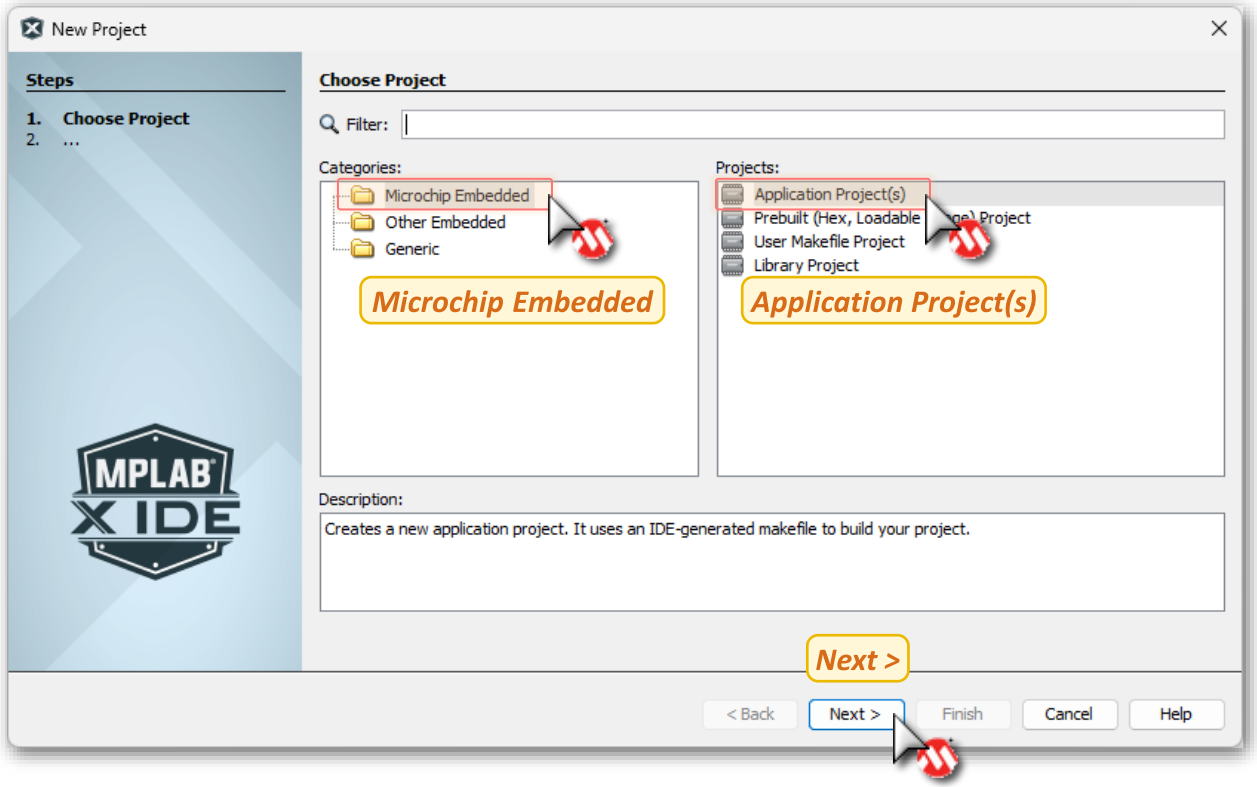


MICROCHIP

Getting Started with First Project

Step 3

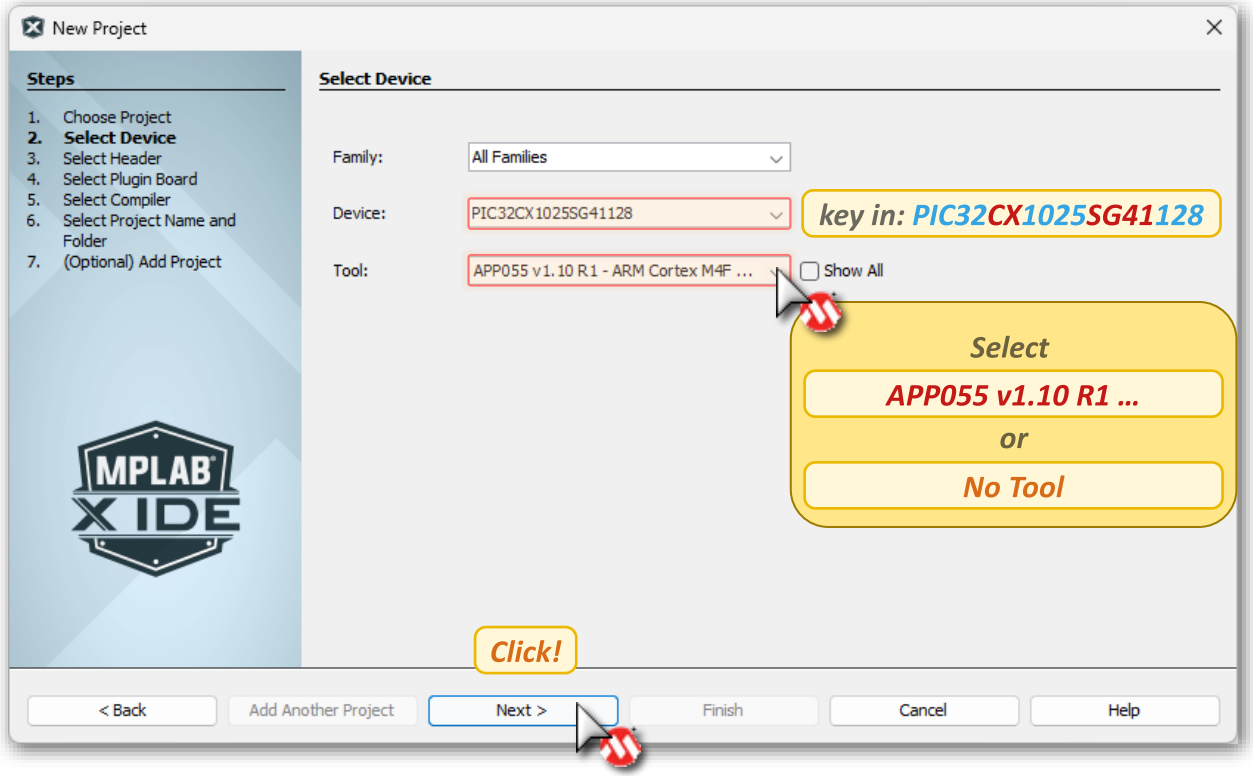
- **Categories** : Microchip Embedded
- **Projects** : Application Project(s)



Getting Started with First Project

Step 4

- **Family** : **All Familes**
- **Device** : **PIC32CX1025SG41128** ***Don't miss any character***
- **Tool** : **APP055 v1.10 R1 - ARM Cortex M4F PIC32CX SG Family … or No Tool**

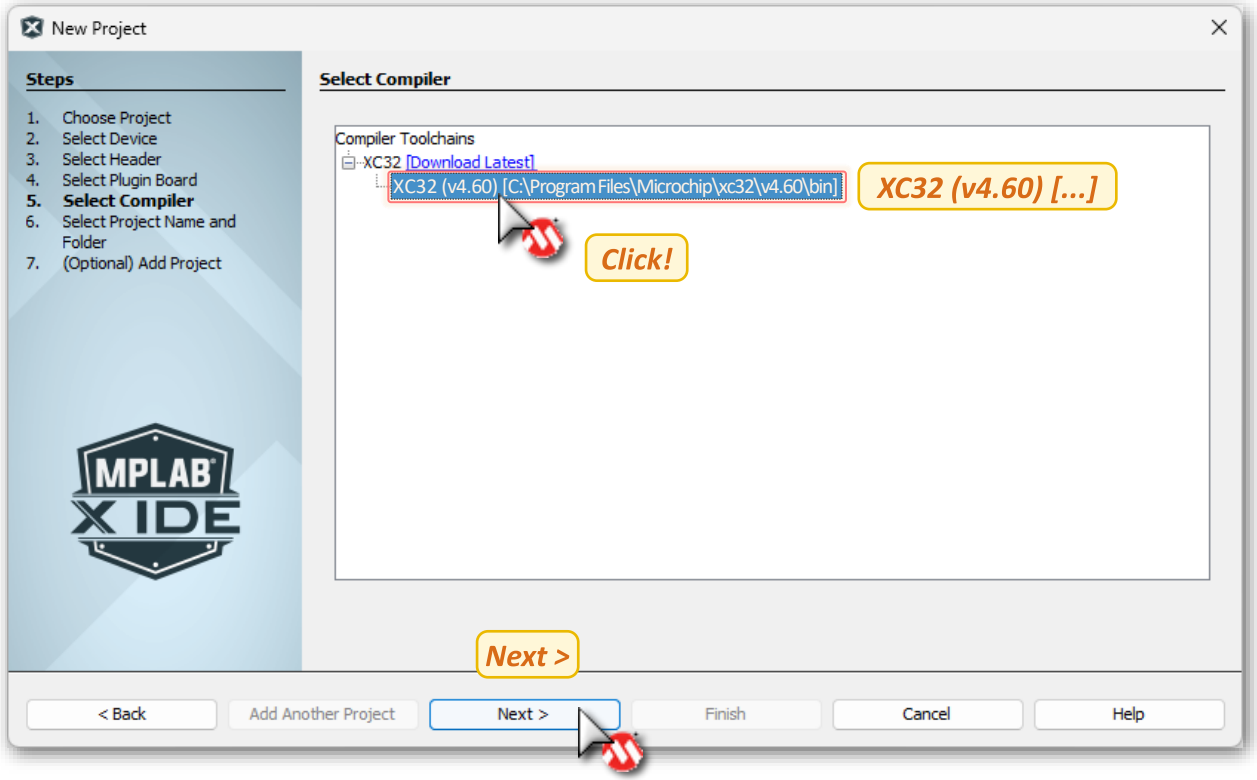


Getting Started with First Project

Step 5

Assign Compiler Toolchains

- **XC32** : **XC32 (v4.60)[…]**

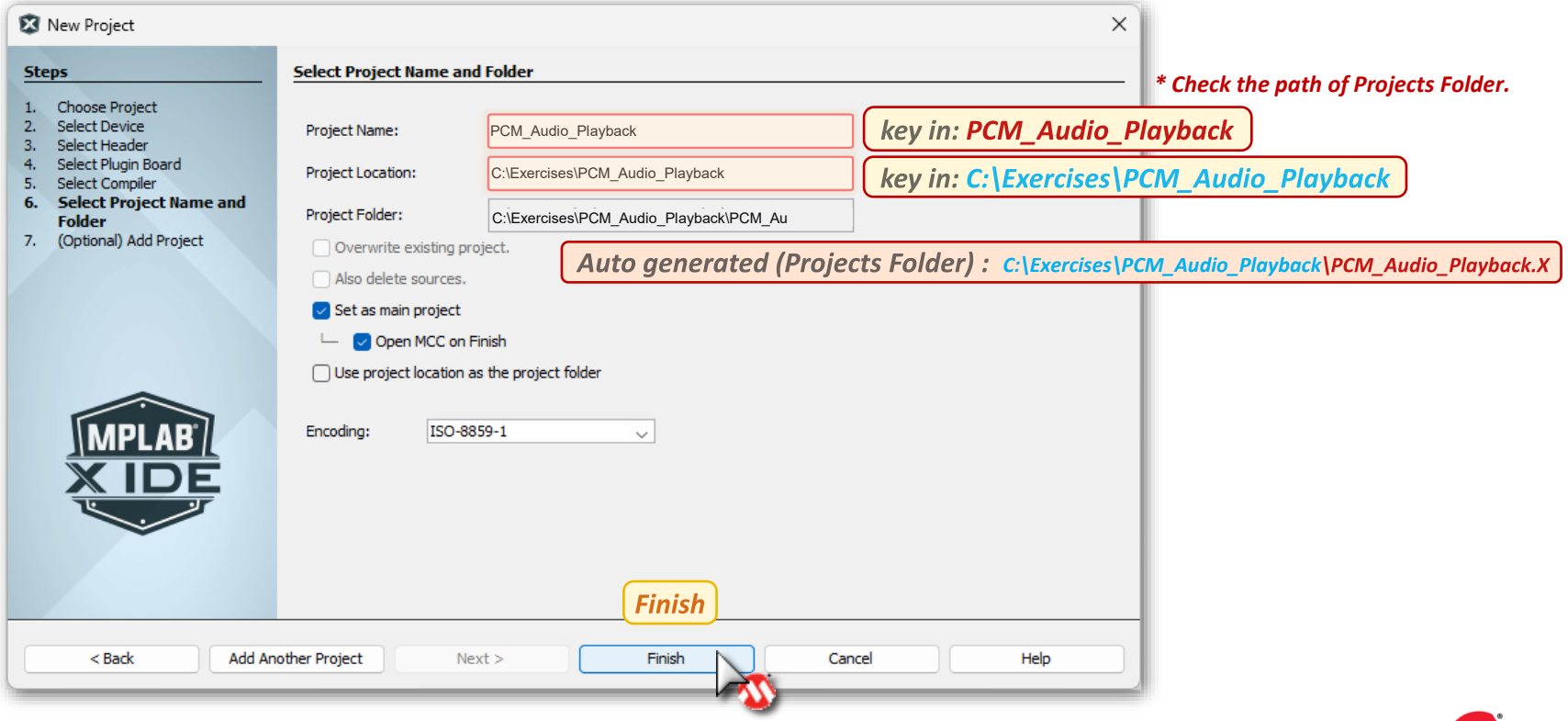


Getting Started with First Project

Step 6

Configure the Project Location and Project Folder Name.

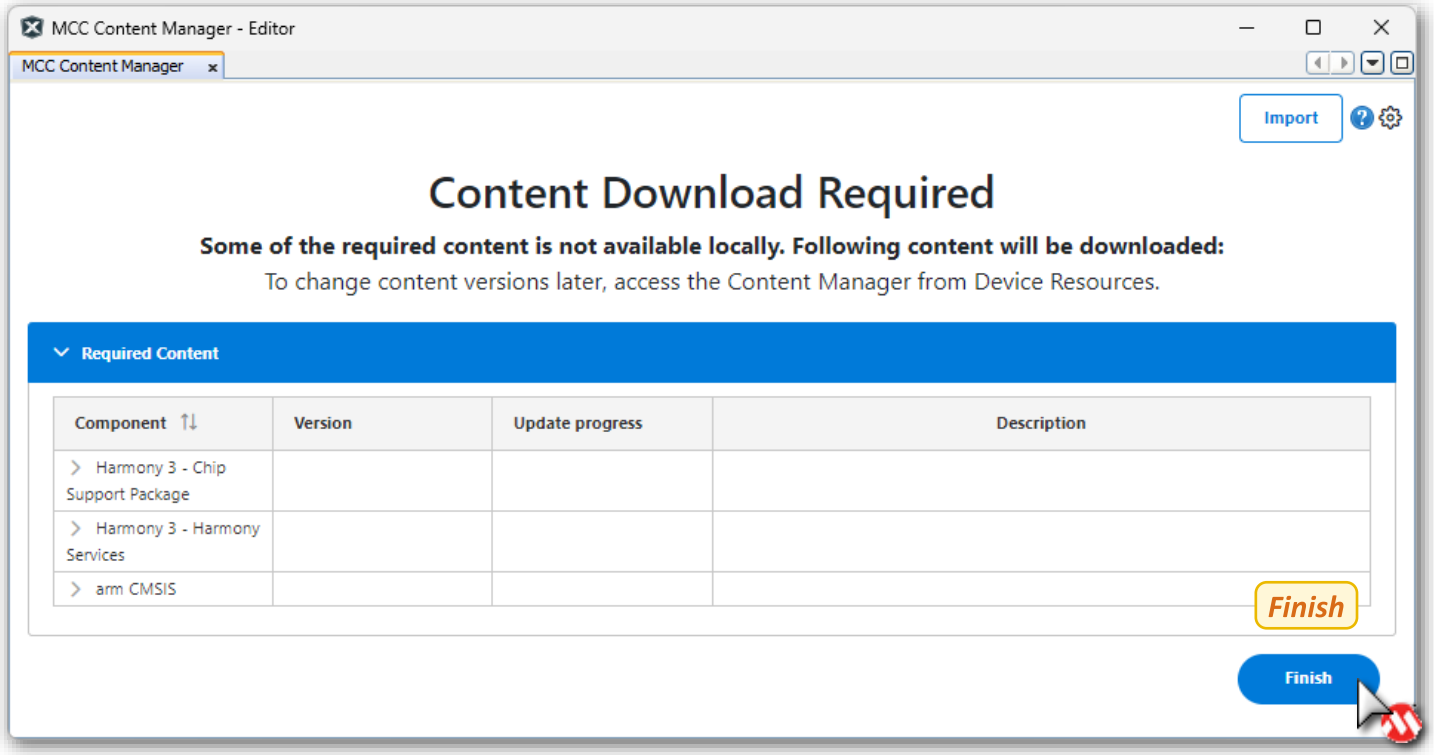
- **Project Name : PCM_Audio_Playback** *No white space of project name or path*
- **Project Location : C:\Exercises\PCM_Audio_Playback**



Getting Started with First Project

Step 7

- **Click Finish and Waiting for automatic download MCC Framework.**



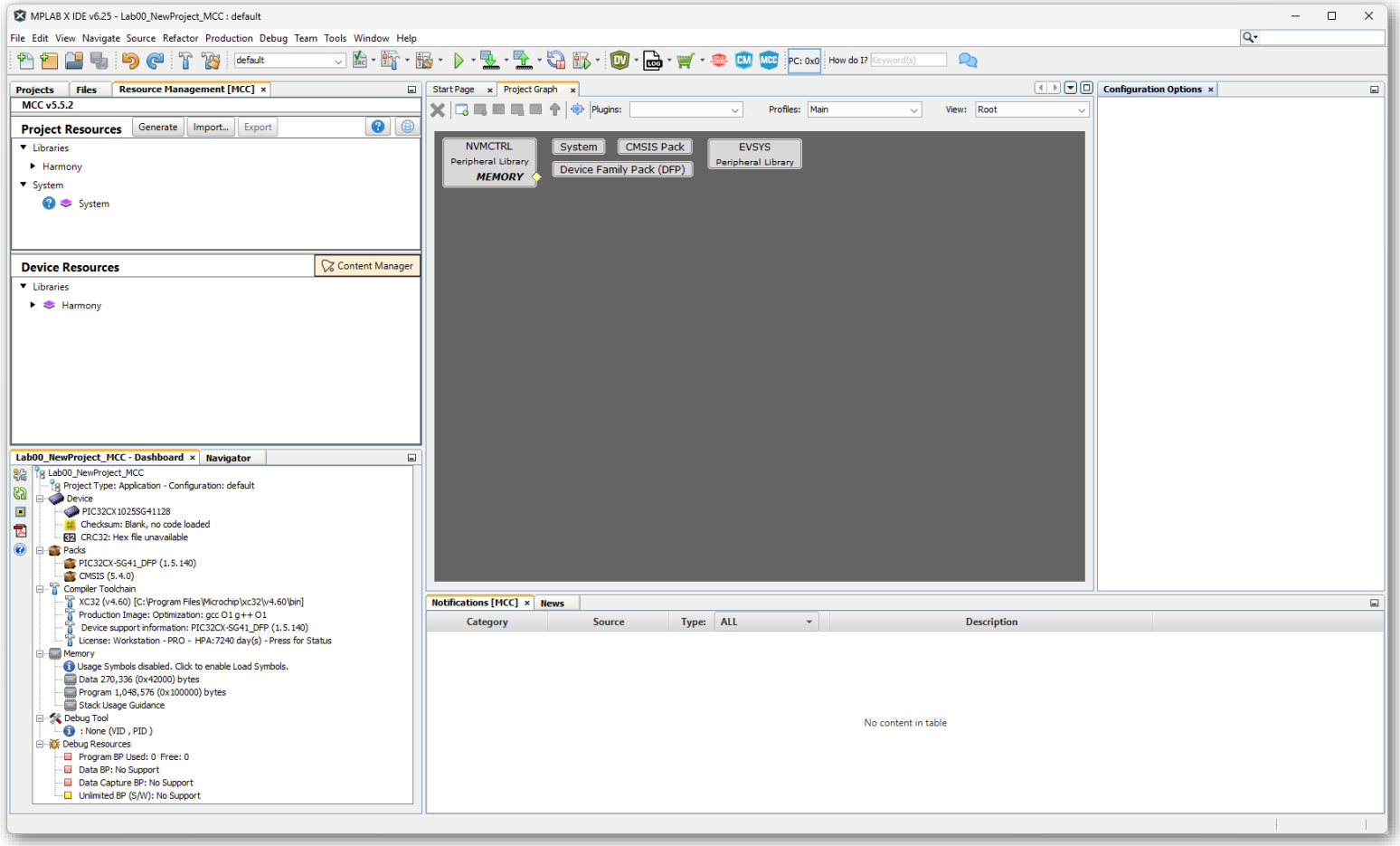
** MCC will download the latest Framework, which only needs to be performed once.*
** This step will not occur in the course because the MCC Framework is already installed on the computer.*

MICROCHIP

Getting Started with First Project

Step 8

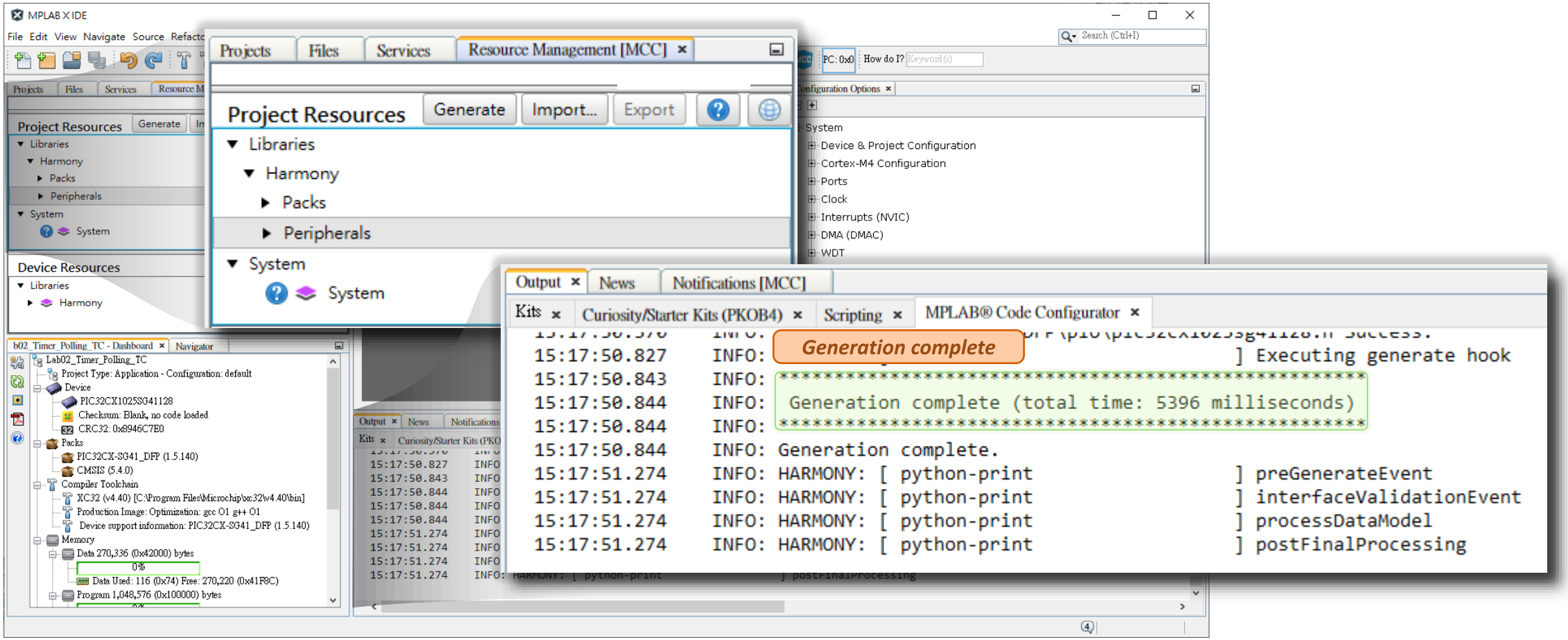
- **MCC Harmony Configurator interface show up.**



Getting Started with First Project

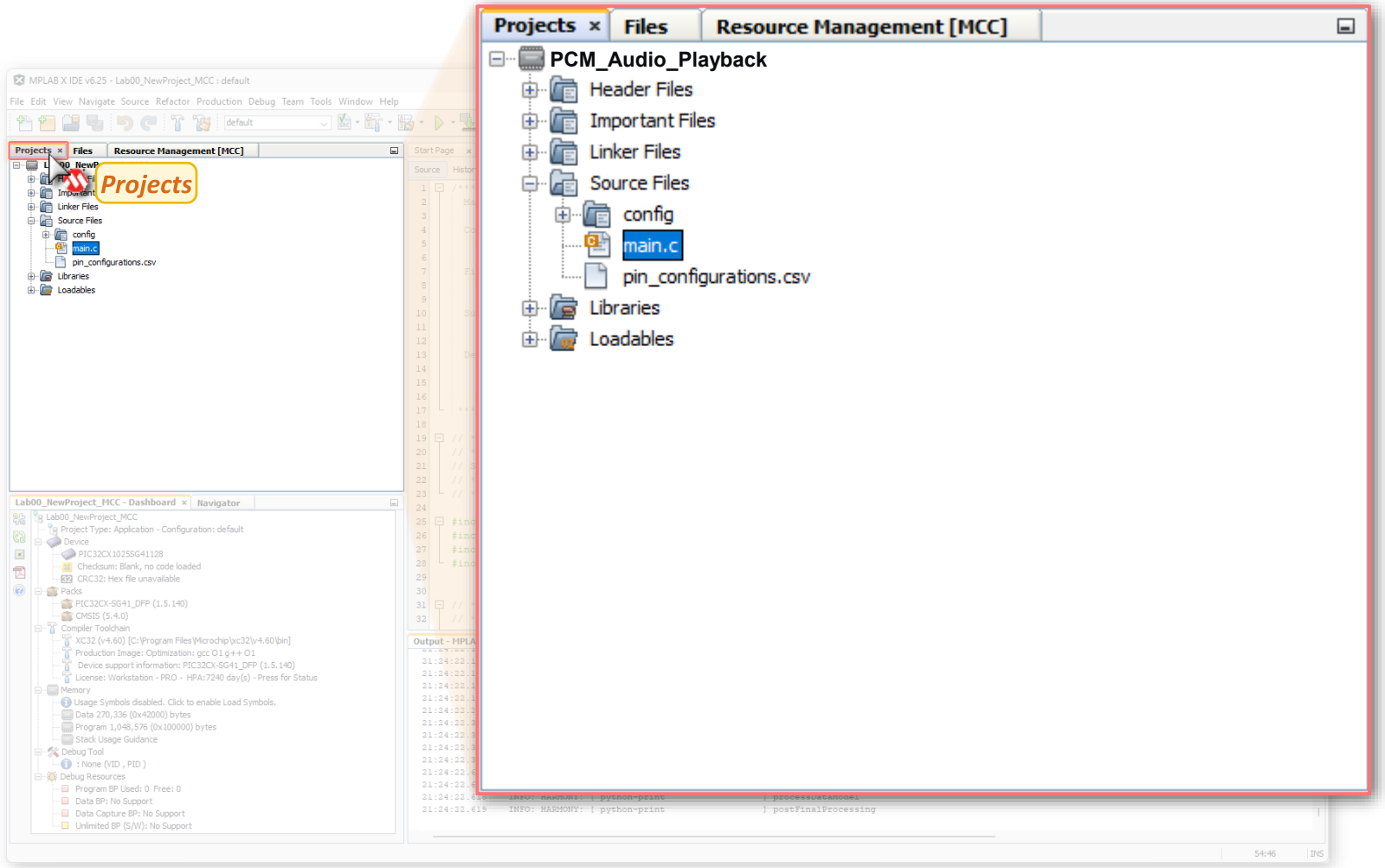
Step 9

- **Click on** Generate **button to generate your code after configuration.**



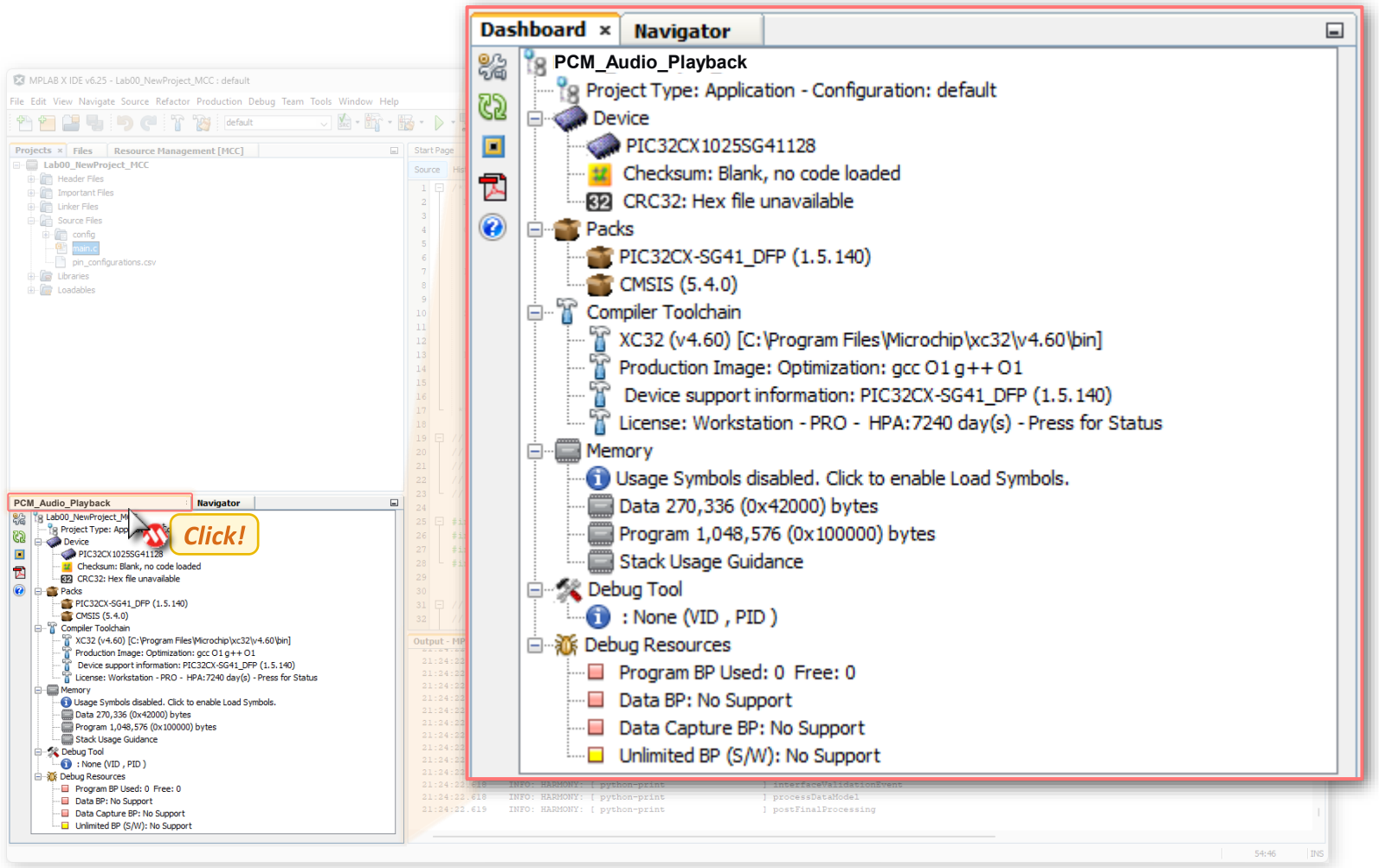
MPLAB X IDE Interface

- **Project window.**



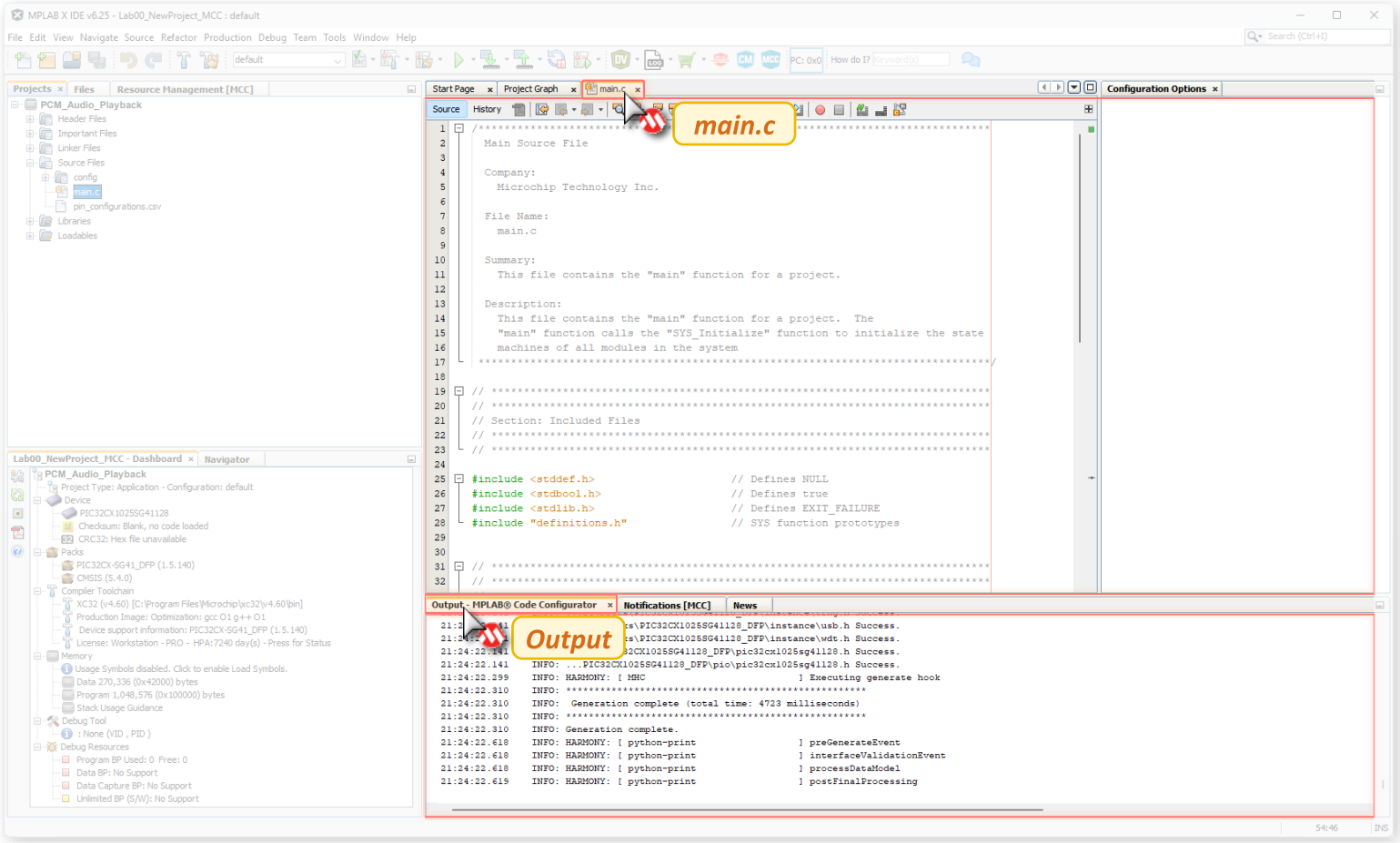
MPLAB X IDE Interface

- **Dashboard window.** **(The version of modules might be different to yours.)**



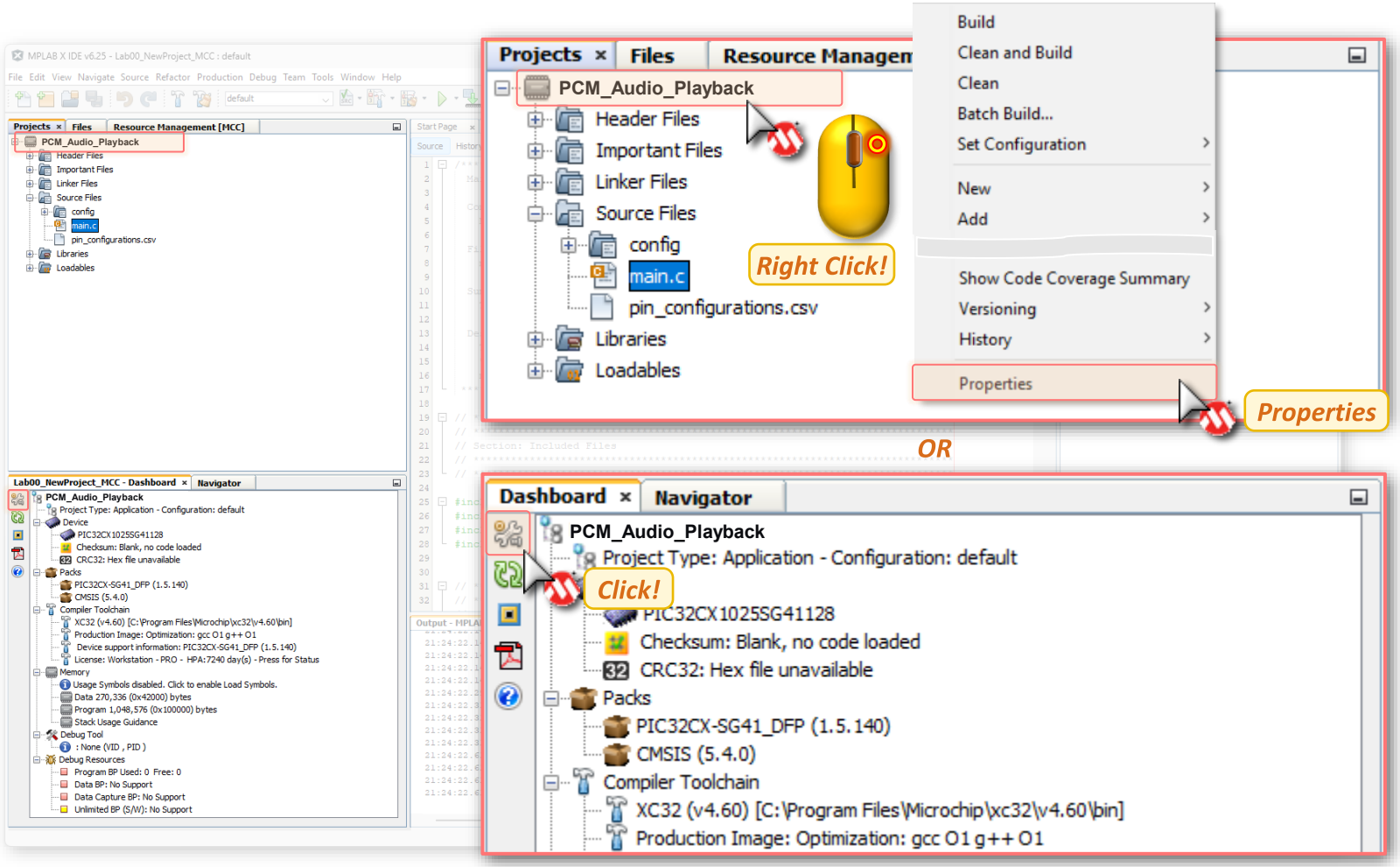
MPLAB X IDE Interface

- **Source code and Output window.**



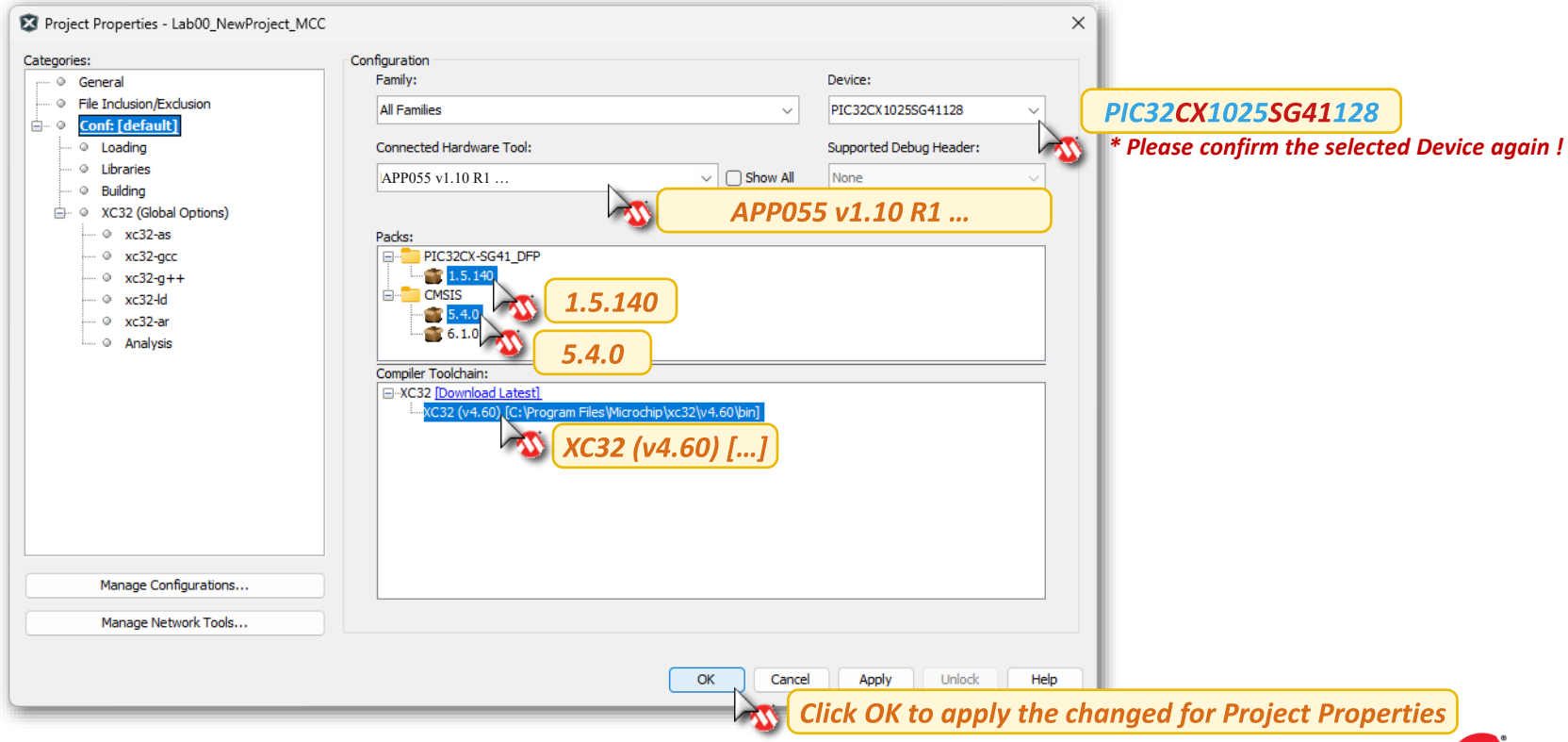
Project Properties

- Click  on Dashboard or Right click on project and select the **Properties**.



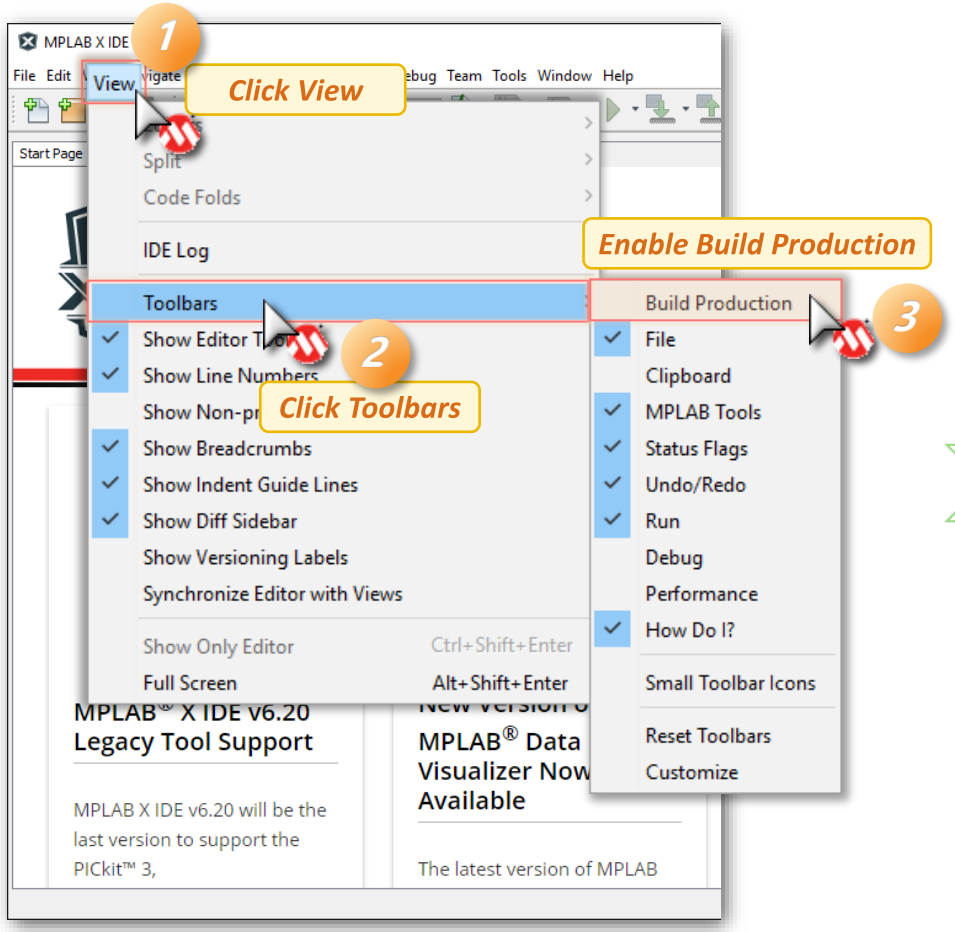
Project Properties

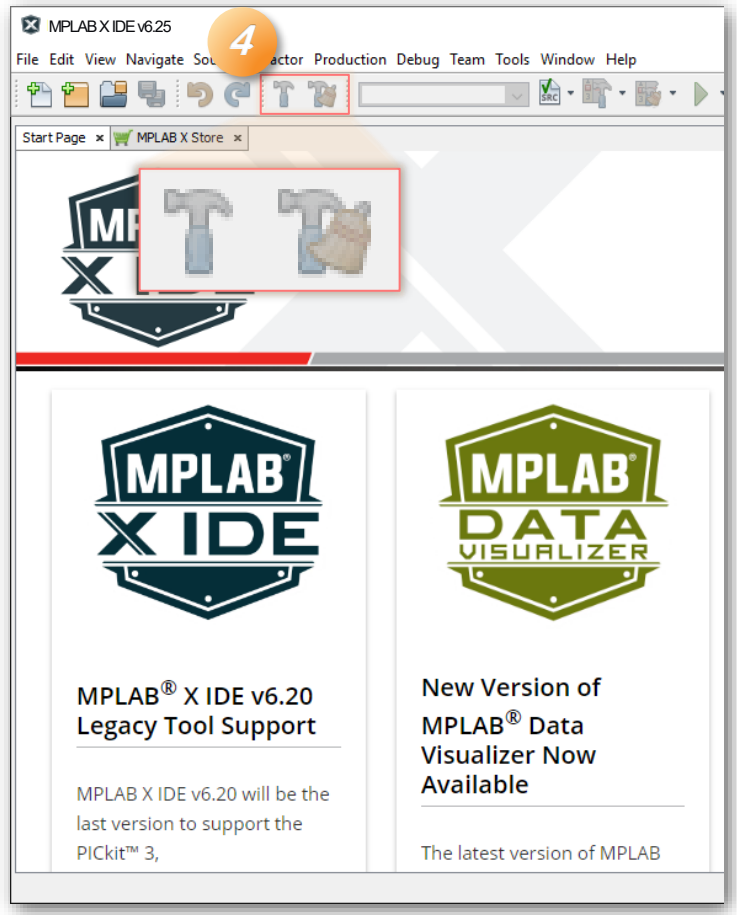
- Connected Hardware Tool : APP055 v1.10 R1 …
- Packs - PIC32CX-SG41_DFP : 1.5.140
- Packs - CMSIS : 5.4.0 or Newest
- Compiler Toolchain : XC32 (v4.60) […]



Build Production Tools

- **Add the Build Production to Toolbars to make it easy to Clear and Build projects.**

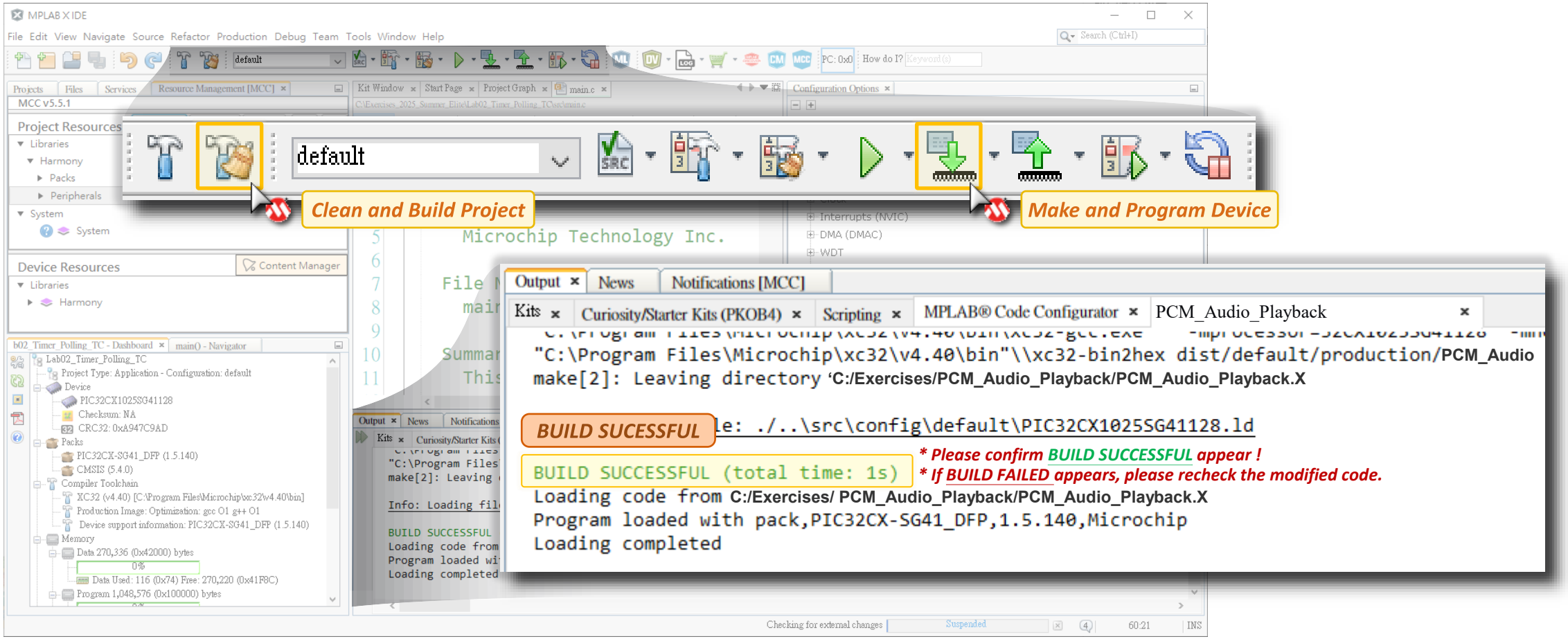




Getting Started with First Project

Step 10

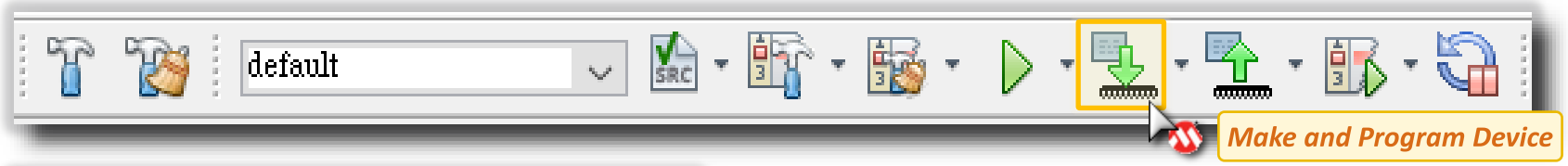
- Click on icon  to Clean and Build Project and then  to Program code to EVB.

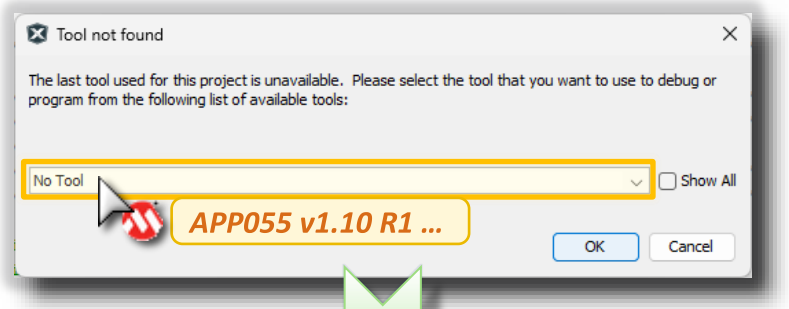


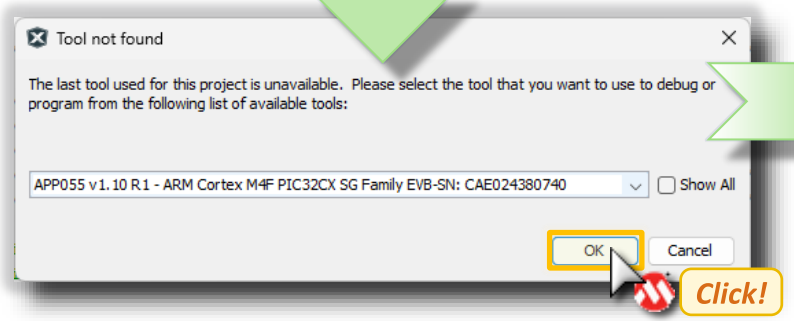
Getting Started with First Project

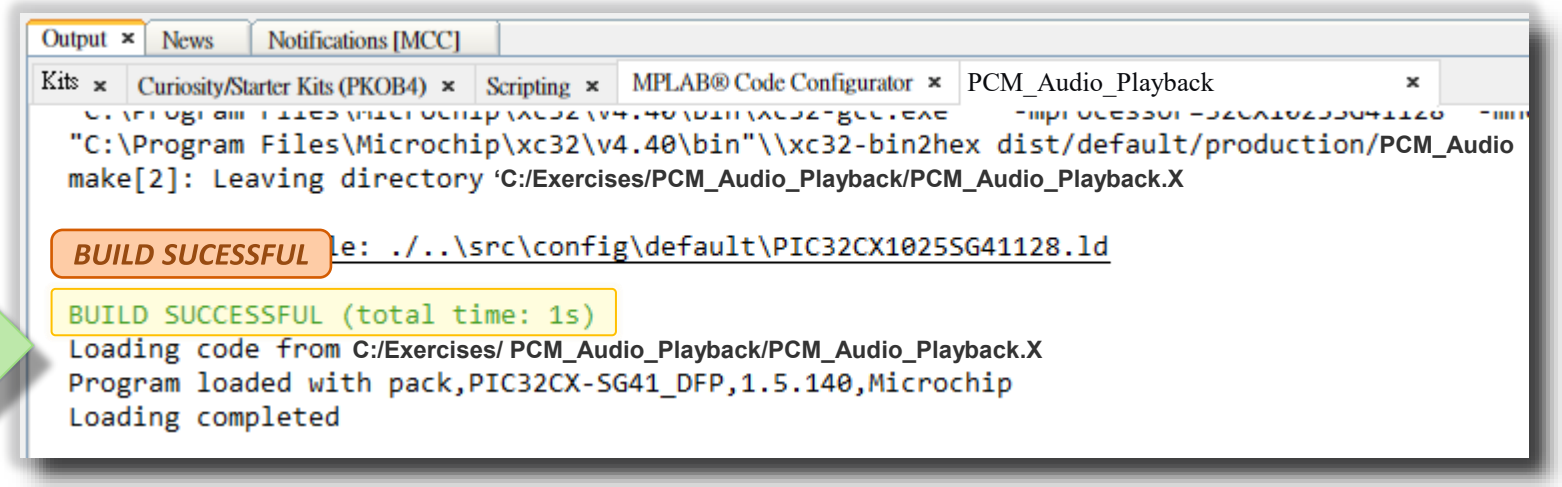
Hint - Tool not found

- If your project has not selected the tool before, "Tool not found" window will appear, please select "APP055 v1.10 R1 …" and then click OK, programming will continue.





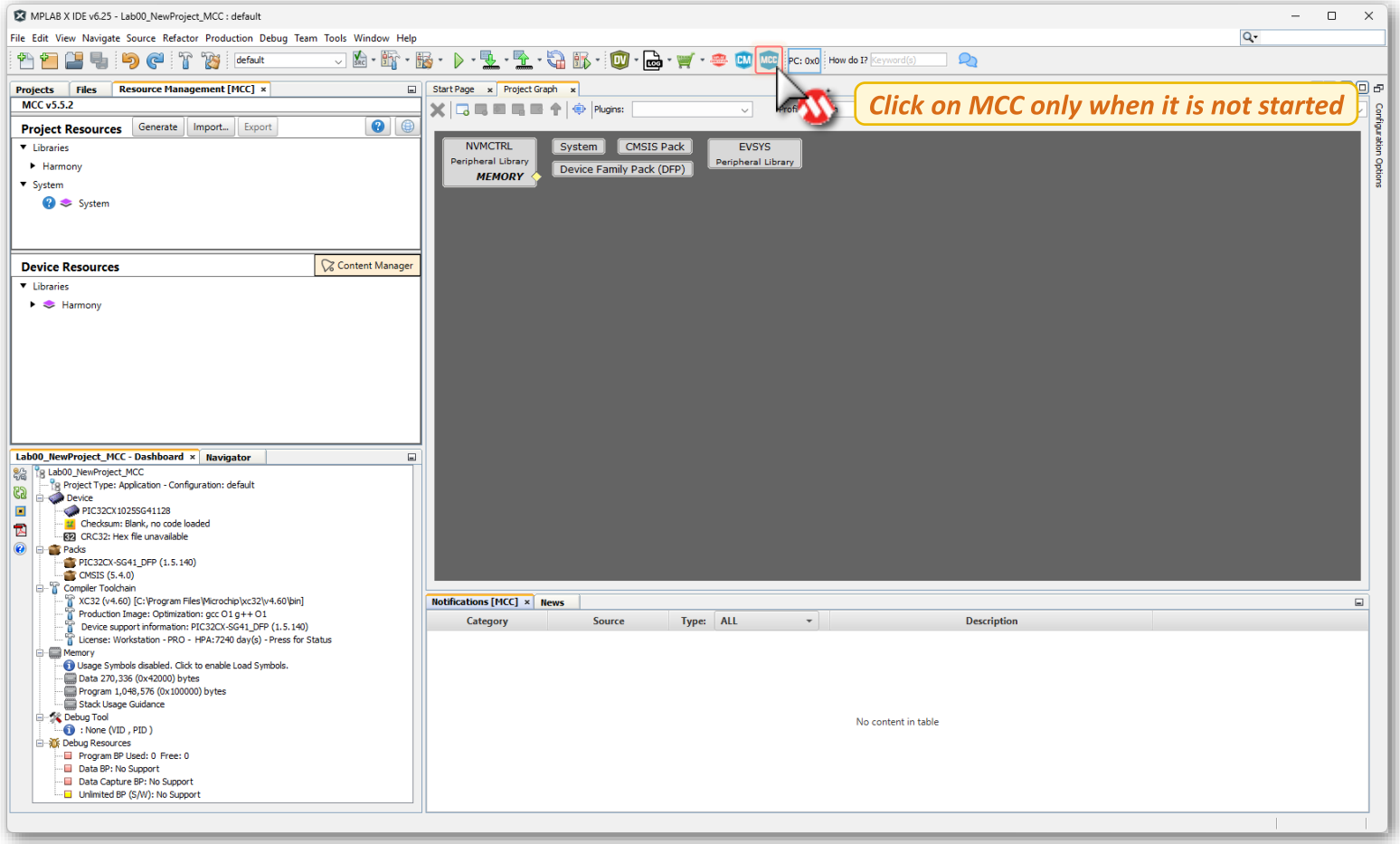




MPLAB® Code Configurator (MCC) Harmony Interface Introduction

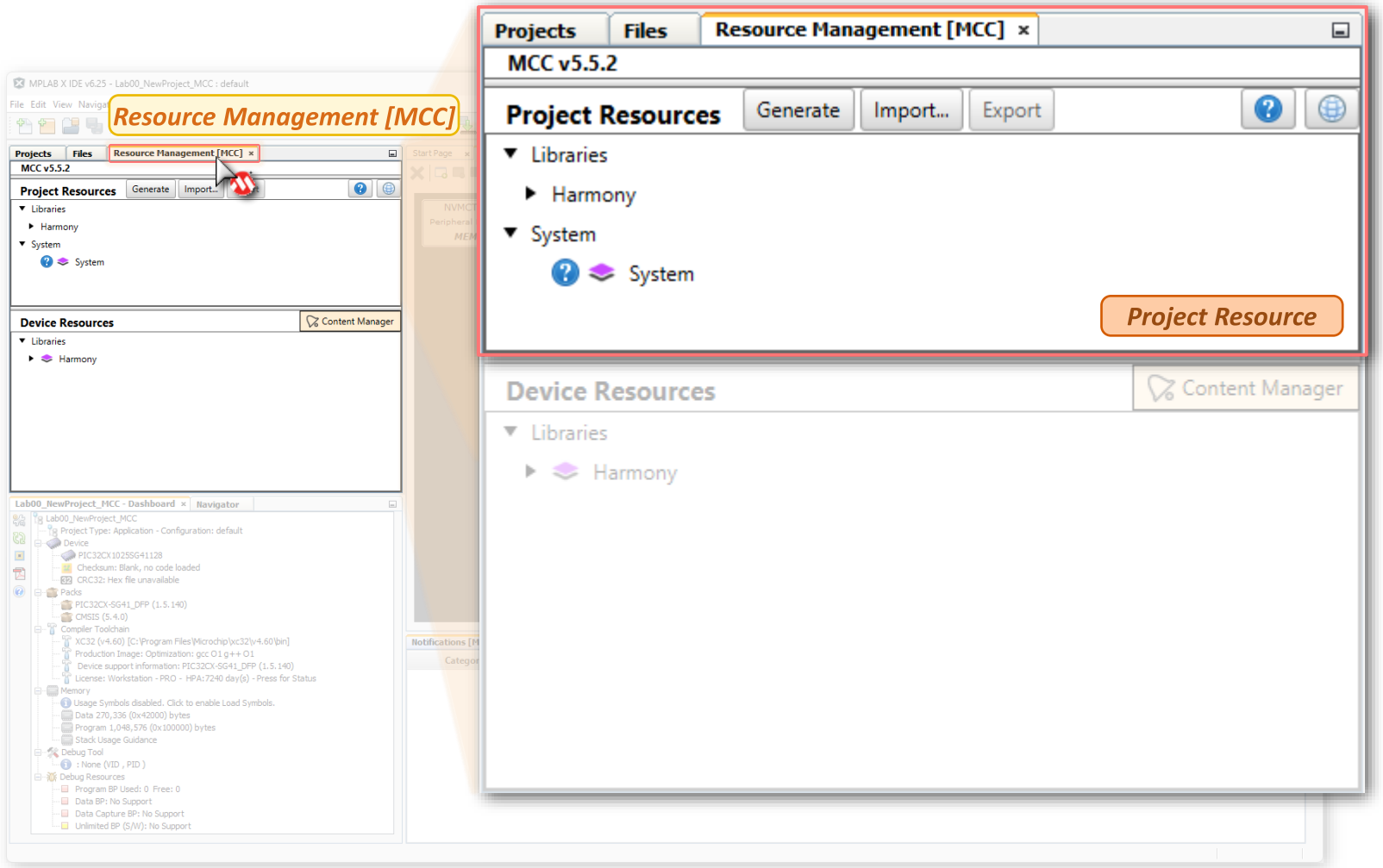
MCC Harmony Interface

- Let's go back to MCC Harmony and view the interface.
- **Notice !** If you close MCC already , please click the MCC icon and reopen MCC.



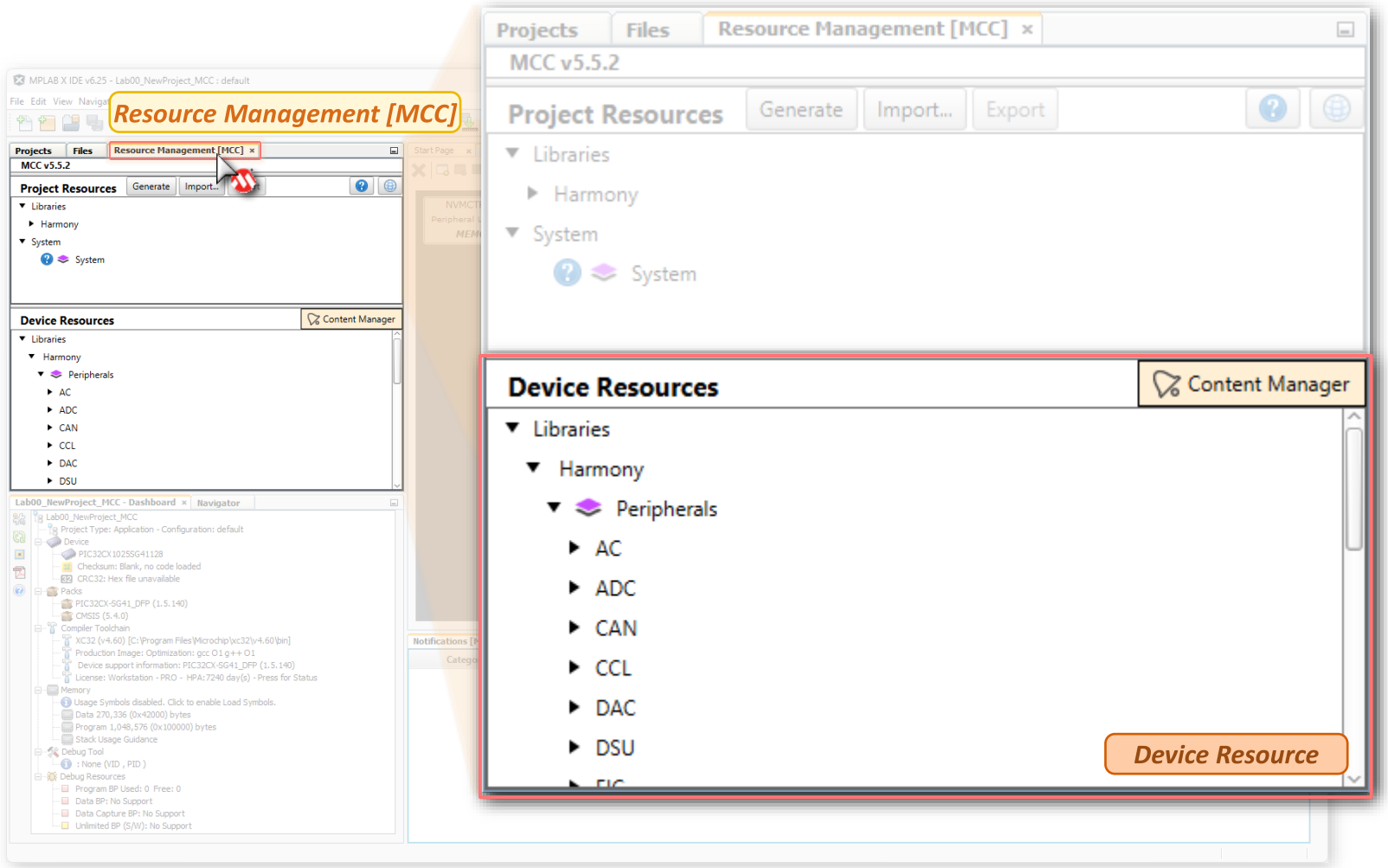
MCC Harmony Interface

- **Project Resource, the modules already added in current project.**



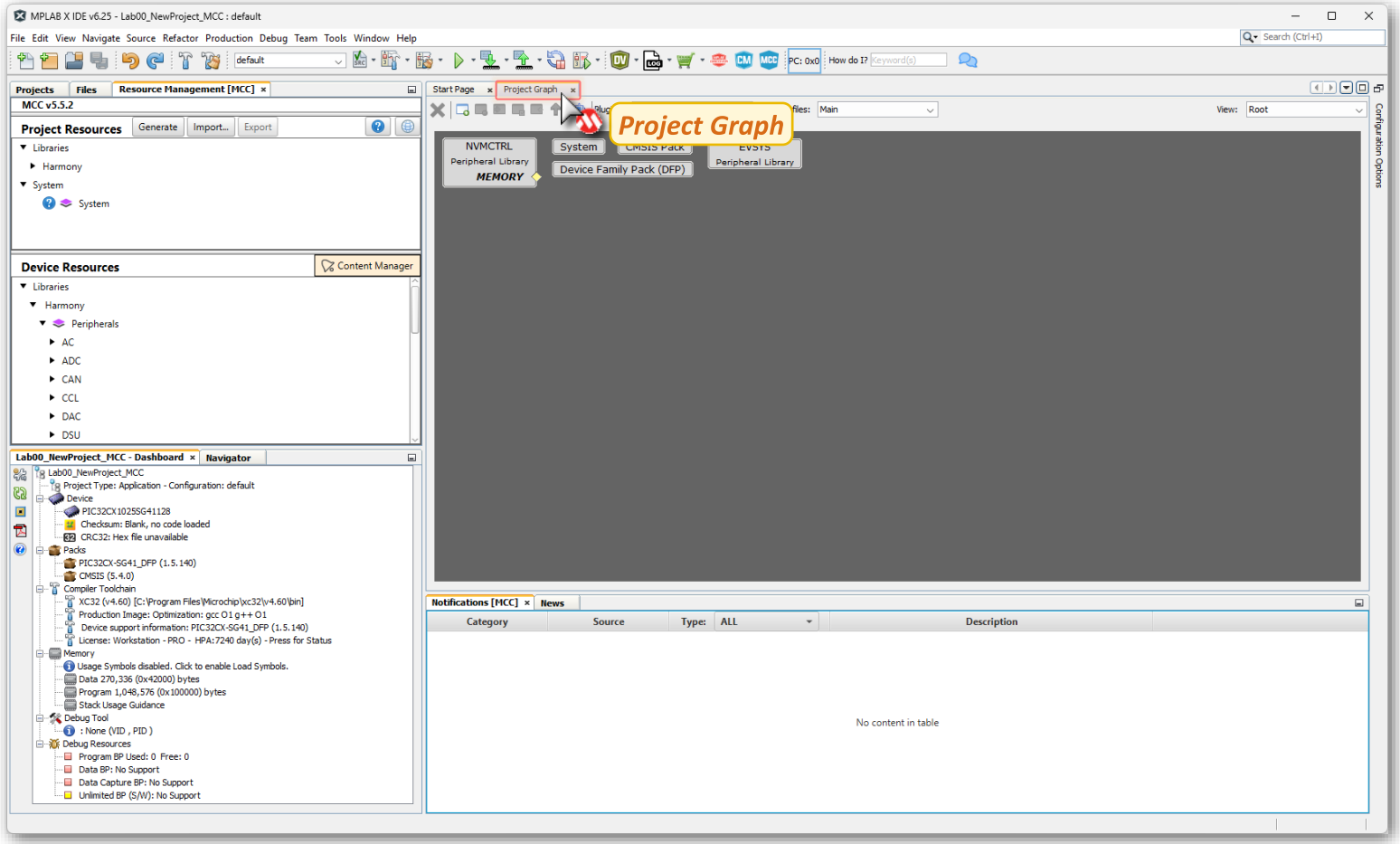
MCC Harmony Interface

- **Device Resource, the modules not in current project but available to be added.**



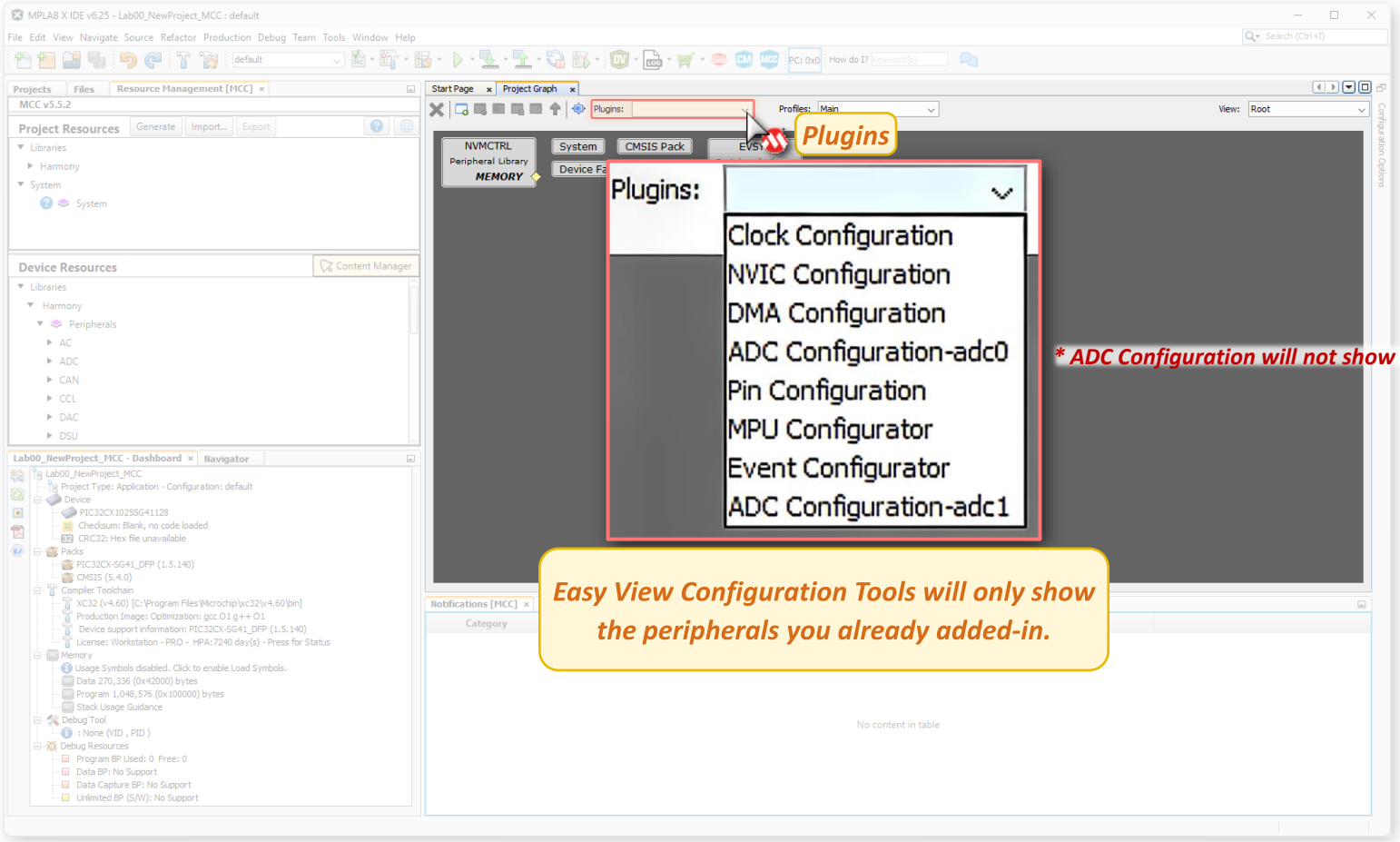
MCC Harmony Interface

- **Project Graph, the connection diagram between modules.**



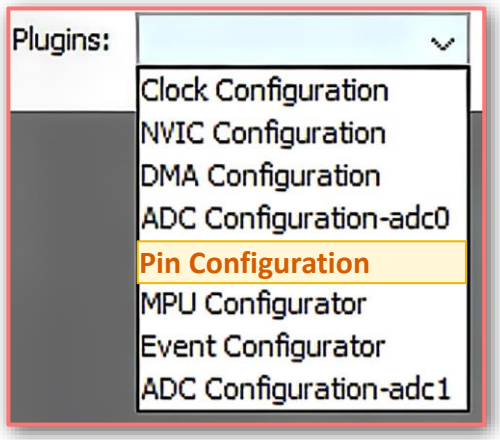
MCC Harmony Interface

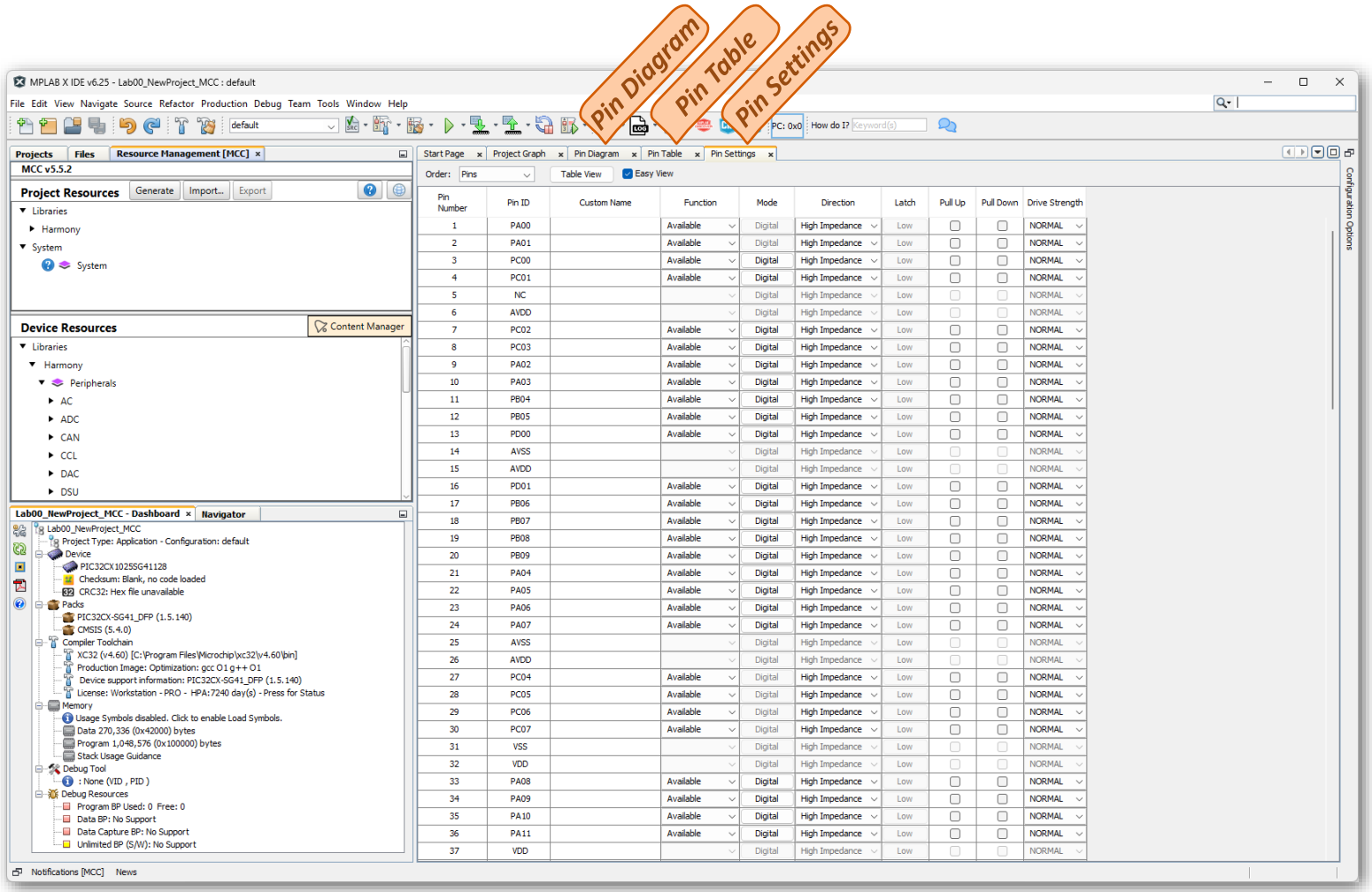
- **Project Graph → Plugins, the Easy View Configuration of modules.**



MCC Harmony Interface

- **Project Graph → Plugins : Pin Configuration**

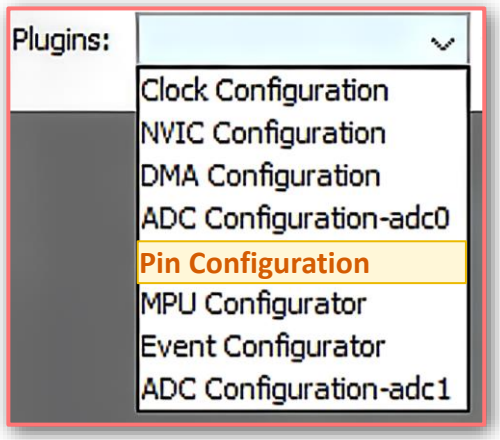


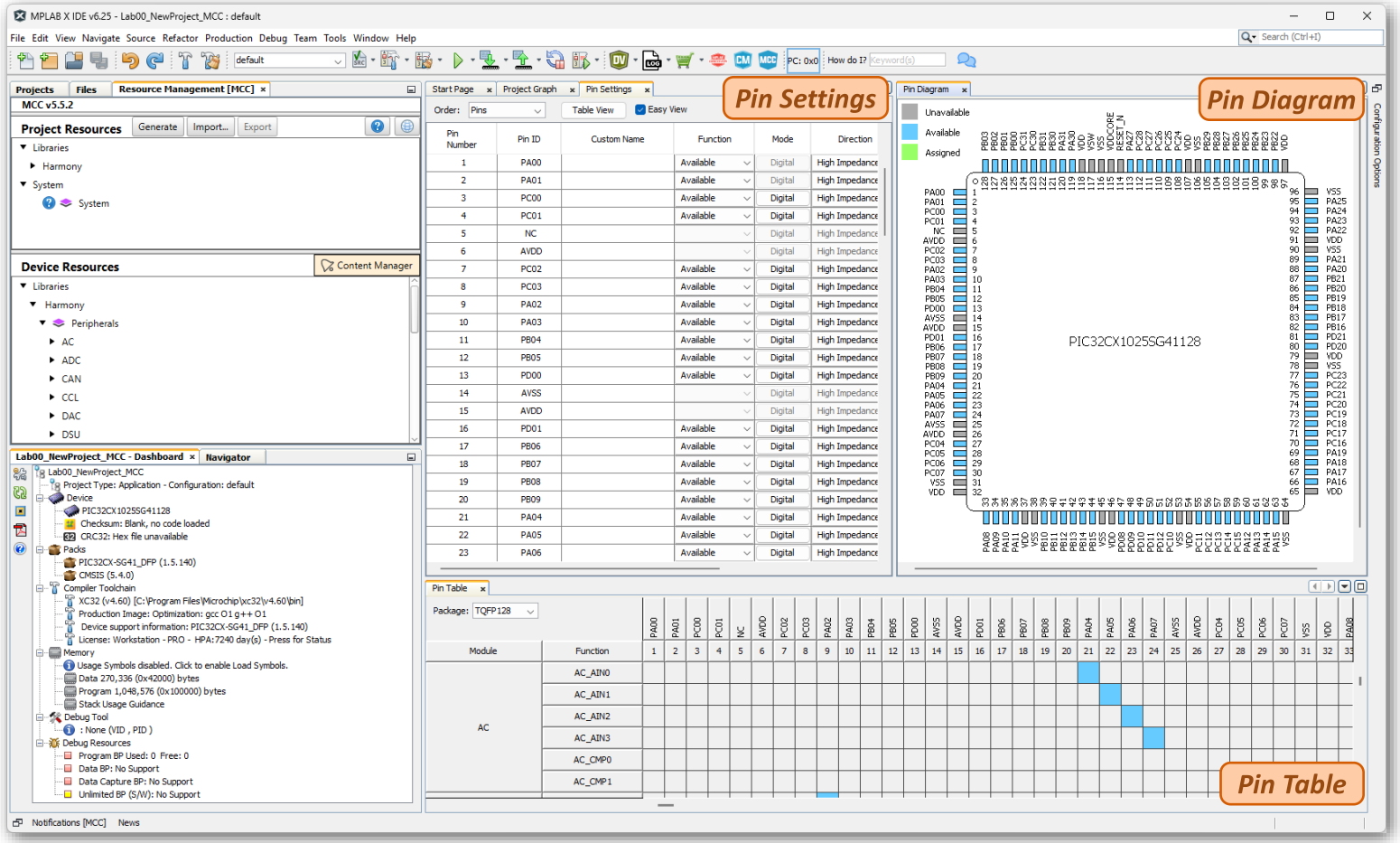


MICROCHIP

MCC Harmony Interface

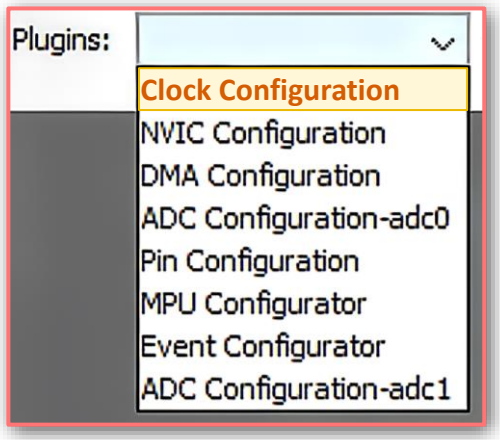
- **Project Graph → Plugins : Pin Configuration (Dock three windows to different locations)**

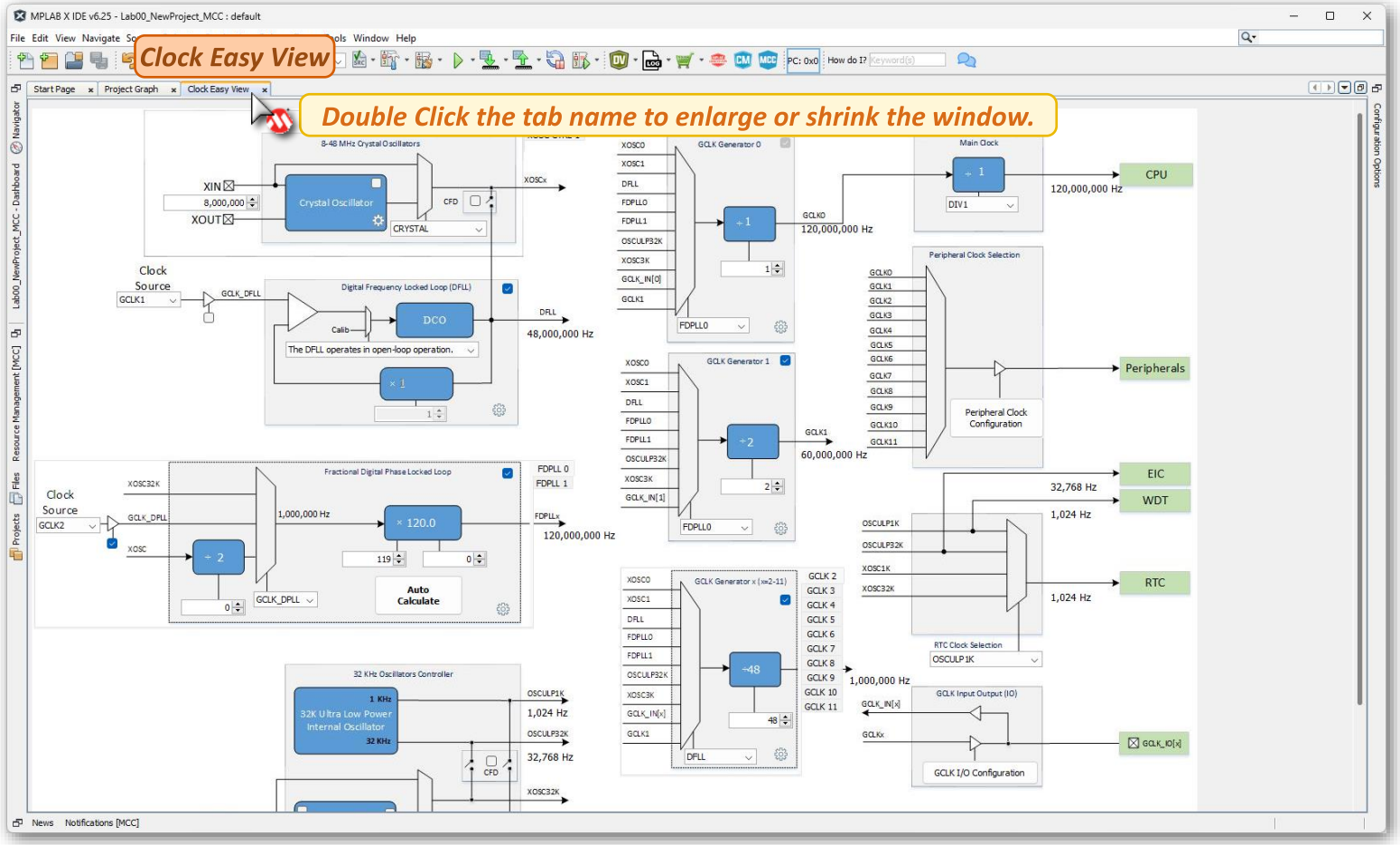




MCC Harmony Interface

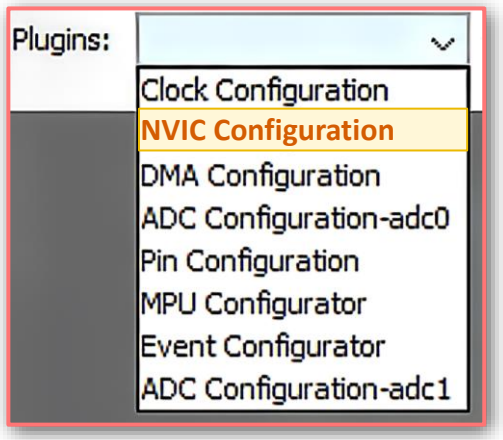
- **Project Graph → Plugins : Clock Configuration**



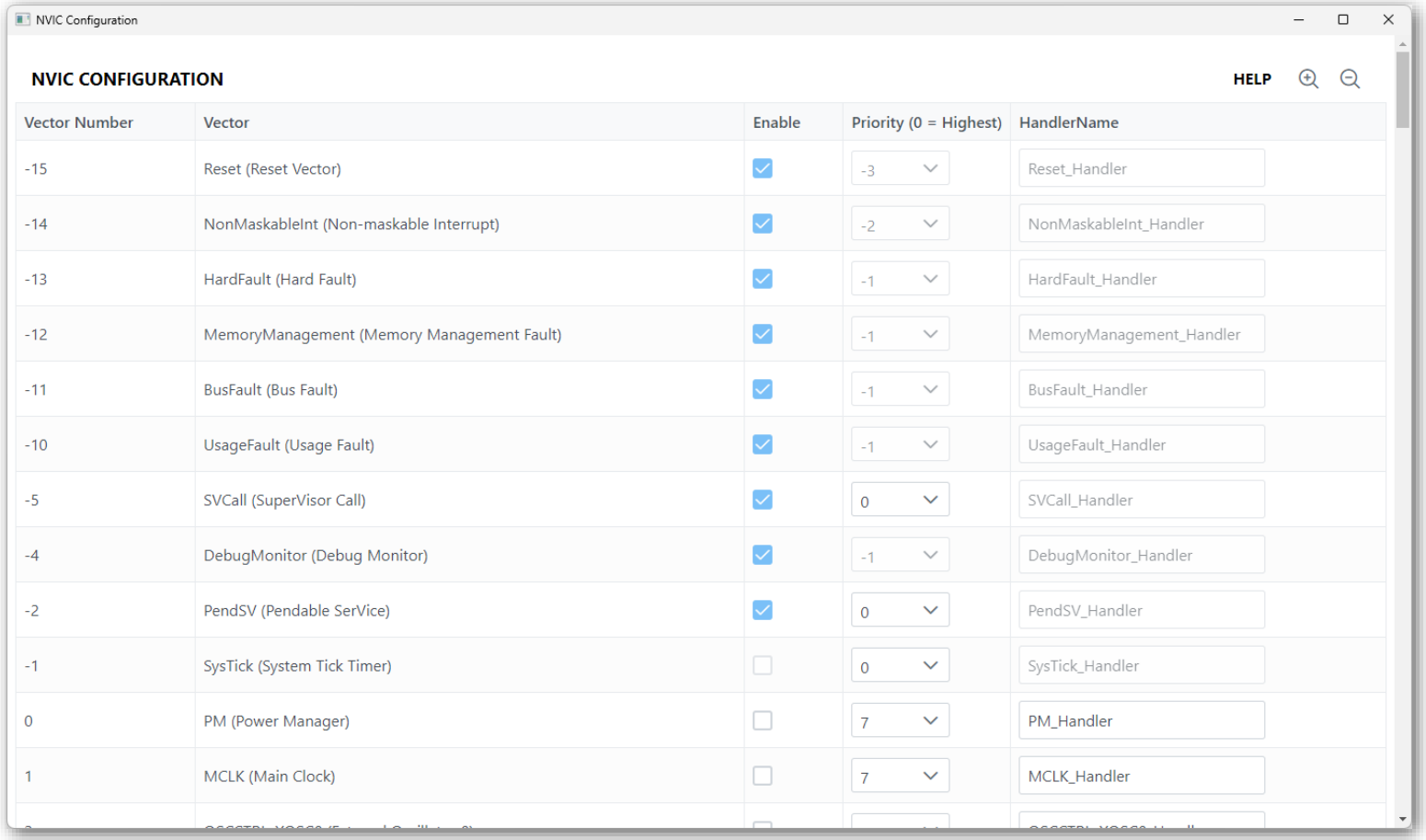


MCC Harmony Interface

- **Project Graph → Plugins : NVIC Configuration (Open a new window after clicked)**



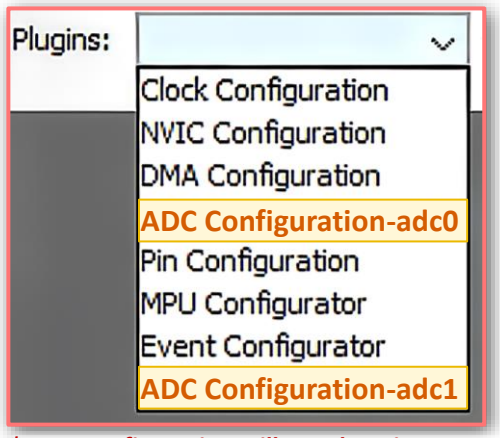
NVIC Configuration



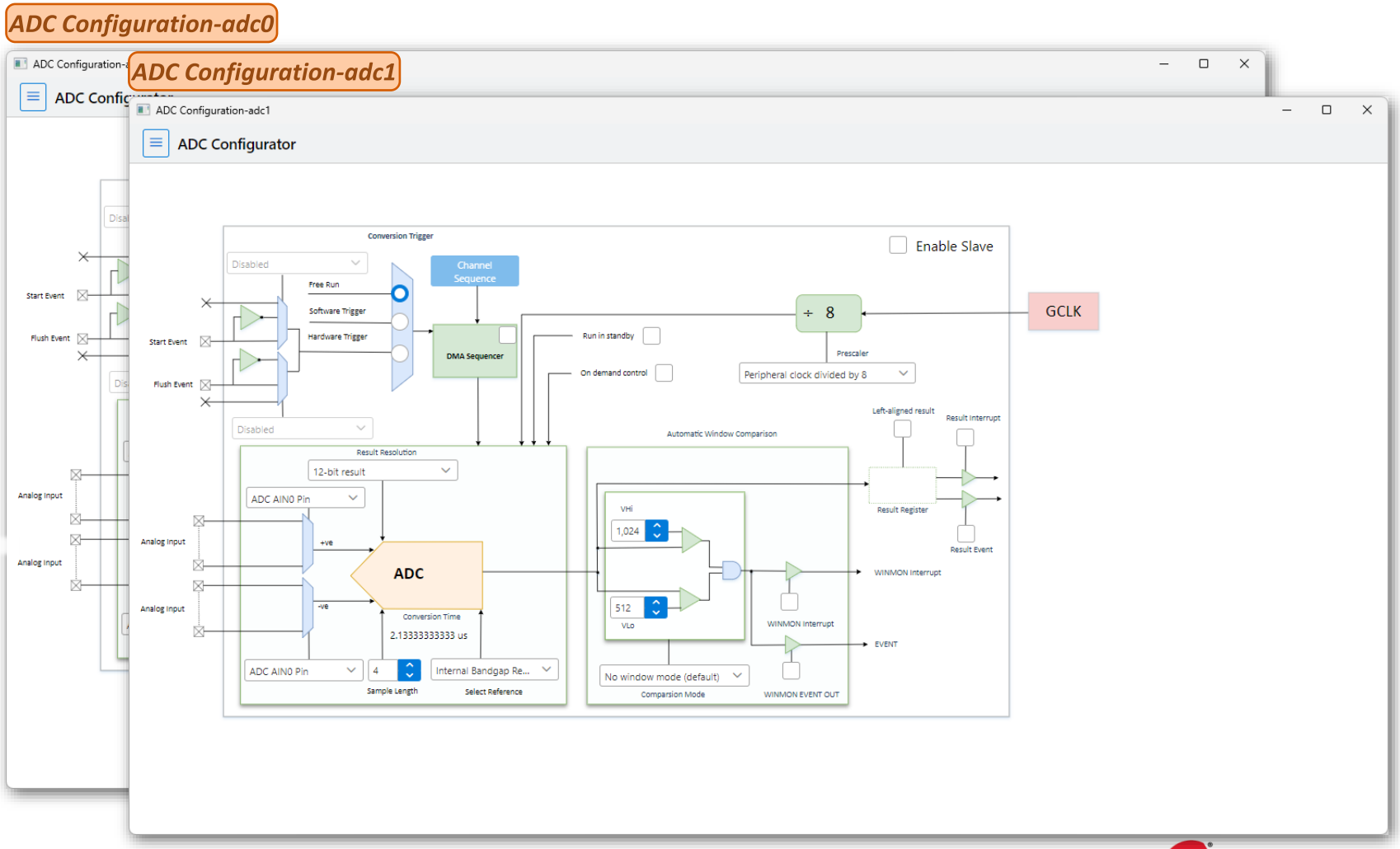
Vector Number	Vector	Enable	Priority (0 = Highest)	HandlerName
-15	Reset (Reset Vector)	☑	-3	Reset_Handler
-14	NonMaskableInt (Non-maskable Interrupt)	☑	-2	NonMaskableInt_Handler
-13	HardFault (Hard Fault)	☑	-1	HardFault_Handler
-12	MemoryManagement (Memory Management Fault)	☑	-1	MemoryManagement_Handler
-11	BusFault (Bus Fault)	☑	-1	BusFault_Handler
-10	UsageFault (Usage Fault)	☑	-1	UsageFault_Handler
-5	SVCall (SuperVisor Call)	☑	0	SVCall_Handler
-4	DebugMonitor (Debug Monitor)	☑	-1	DebugMonitor_Handler
-2	PendSV (Pendable SerVice)	☑	0	PendSV_Handler
-1	SysTick (System Tick Timer)	☐	0	SysTick_Handler
0	PM (Power Manager)	☐	7	PM_Handler
1	MCLK (Main Clock)	☐	7	MCLK_Handler

MCC Harmony Interface

- **Project Graph → Plugins : ADC Configuration (Open a new window after clicked)**

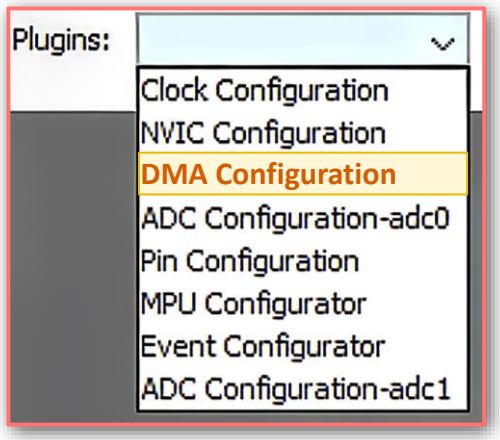


** ADC Configuration will not show in a new project*

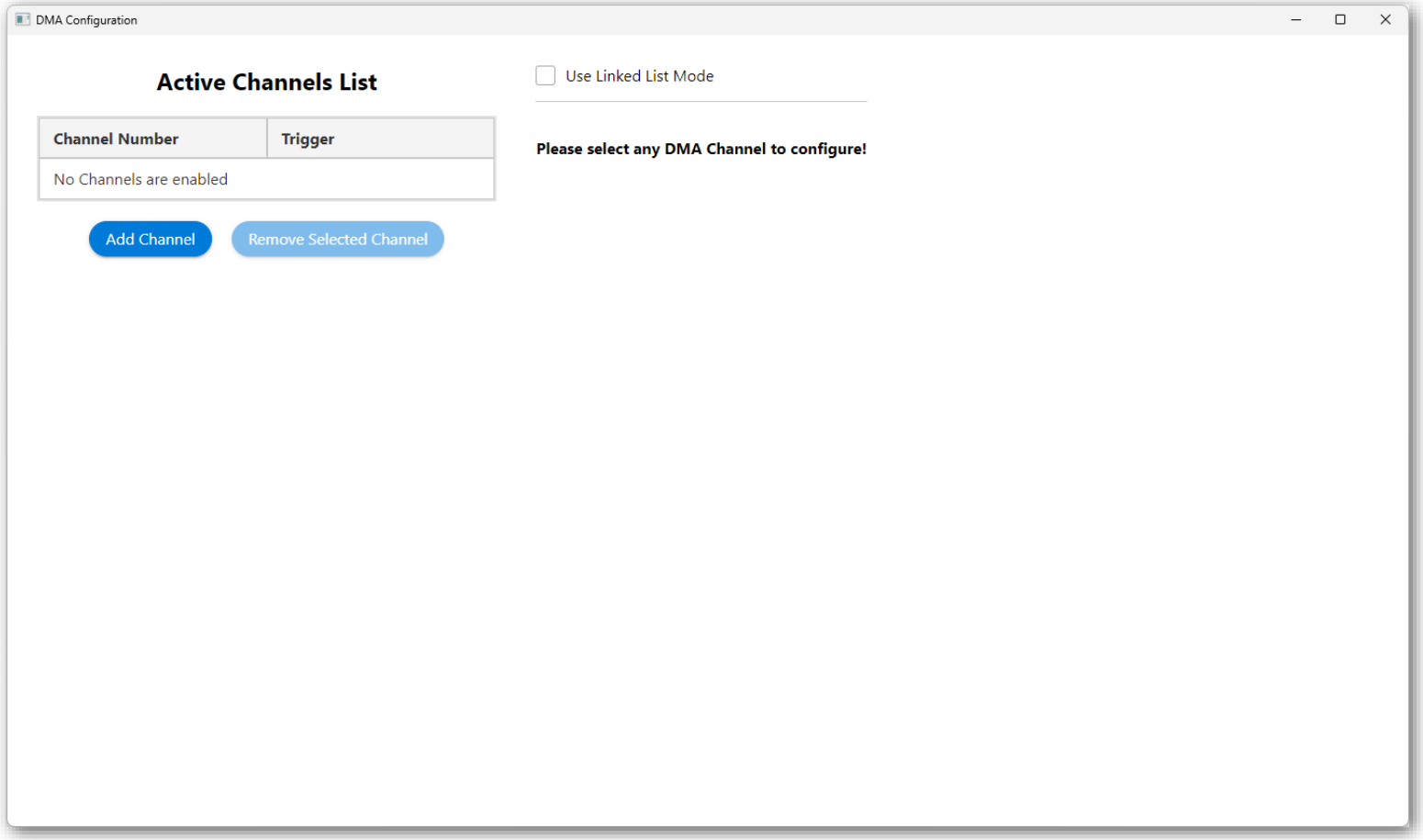


MCC Harmony Interface

- **Project Graph → Plugins : DMA Configuration (Open a new window after clicked)**

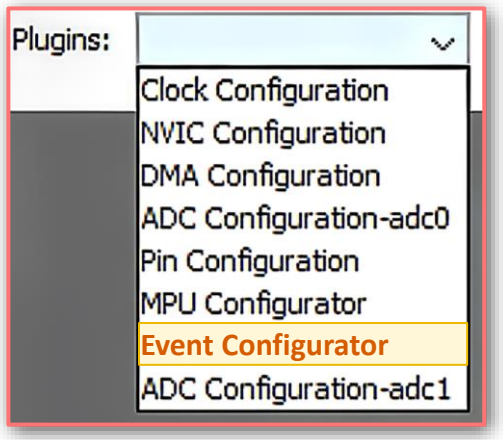


DMA Configuration

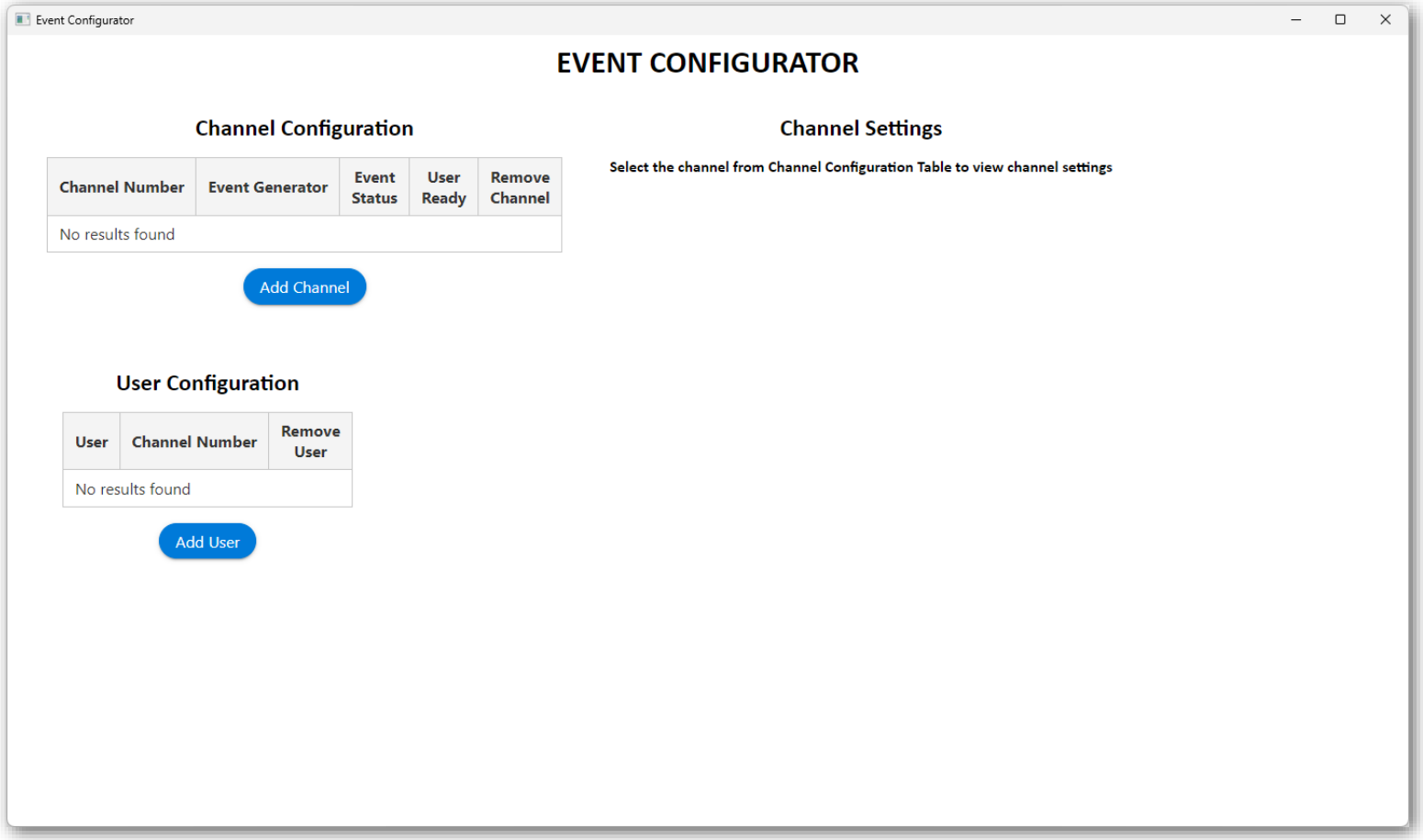


MCC Harmony Interface

- **Project Graph → Plugins : Event Configurator (Open a new window after clicked)**



Event Configurator

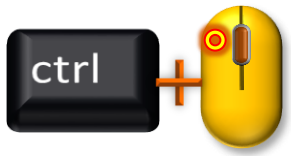


Appendix :
Code Trace and PORT Register

MPLAB X IDE Shortcut key

- **MPLAB X IDE Provide a few shortcut key to enhance your performance.**
  - 

    Show All Code Completion Popup (Code Automatically Complete)

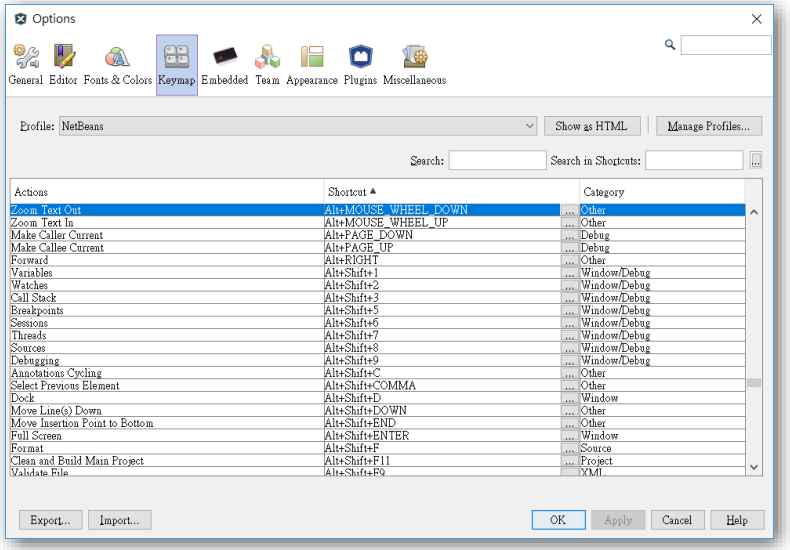
    (Alternative shortcut : Ctrl **+ ** , Ctrl + Alt + Space**)**
  - 

    Open Macro or Function (Code Trace)
  - 

    Backward (Code Trace)
  - 

    Code Format
  - 

    Toggle Comment
- **More ? Tools > Options > Keymap**



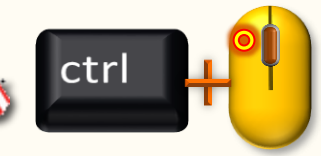
MPLAB X IDE Code Trace

Prototype of Macros or Functions

- **Using shortcut keys, you can easily find the code prototype of Macros or Functions, and return to the previous location at any time.**

main.c

```
int main (void)
{
    …
    while( true )
    {
        if ( ++i >= 2000000 )
        {
            i = 0;
            LED1_Toggle();
            LED2_Toggle();
        }
        …
    }
}
```



plib_port.h

```
…
/*** Macros for LED1 pin ***/
#define LED1_Set()              (PORT_REGS->GROUP[0].PORT_OUTSET = ((uint32_t)1U << 12U))
#define LED1_Clear()            (PORT_REGS->GROUP[0].PORT_OUTCLR = ((uint32_t)1U << 12U))
#define LED1_Toggle()           (PORT_REGS->GROUP[0].PORT_OUTTGL = ((uint32_t)1U << 12U))
#define LED1_OutputEnable()     (PORT_REGS->GROUP[0].PORT_DIRSET = ((uint32_t)1U << 12U))
#define LED1_InputEnable()      (PORT_REGS->GROUP[0].PORT_DIRCLR = ((uint32_t)1U << 12U))
#define LED1_Get()              (((PORT_REGS->GROUP[0].PORT_IN >> 12U)) & 0x01U)
#define LED1_PIN                PORT_PIN_PA12
…
```

** Macros will be named according to your custom GPIO name.*

MICROCHIP

PCM Audio

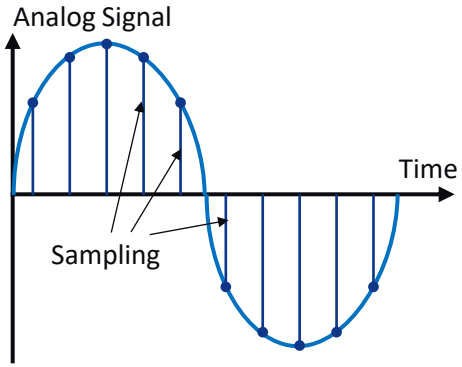
PCM Audio

Pulse Code Modulation (PCM):
A method of taking an analog sound wave (continuous) and converting it into a series of digital numbers (discrete samples). It's the most common way to digitally represent analog audio signals.

- How it works:
  - The analog signal is **sampled** at regular time intervals (e.g., 44.1 kHz = 44,100 samples per second, used in CDs).
  - Each sample's amplitude is **quantized** to the nearest digital value (e.g., 16-bit depth → 65,536 possible levels).
  - These numbers are stored or transmitted as a sequence of binary values.

- Types of PCM:
  - **Linear PCM (LPCM):** The amplitude values are stored directly. This is the most common form (used in WAV, AIFF, CD audio).
  - **Compressed PCM (like μ-law, A-law):** Used in telephony, applies a logarithmic scale to reduce data size.

- Where you encounter PCM audio:
  - Audio CDs → 44.1 kHz, 16-bit, stereo LPCM.
  - WAV and AIFF files (uncompressed PCM).
  - Inside formats like MP3 or AAC → the compressed stream is decoded back to PCM before playback.
  - HDMI, SPDIF digital audio streams carry PCM as a baseline format.



- PCM audio is just the raw digital form of sound waves, before any compression (like MP3, AAC, etc.).

PCM Audio Storage Consumption

Assumptions: Record 10 seconds of mono audio

Samples = Sample rate × 10 s

Size (Padded to 16-bit) = Samples × 2 bytes (Minimum buffer size to store 10-bit or 12-bits raw data)

Sample rate (Hz)	Samples	Size (kB)	Common label	Typical uses
8,000	80,000	160	Telephony / Narrowband speech	PSTN/VoIP speech, IVR, G.711 μ-law/A-law
11,025	110,250	220.5	Lo-fi music (legacy)	Early multimedia, small web audio, down sampled CD
12,000	120,000	240	Speech / Low-rate music (codec-friendly)	AAC/Opus profiles, speech, low-bandwidth streams
16,000	160,000	320	Wideband speech	Modern VoIP, conferencing, ASR (speech recognition), podcasts
22,050	220,500	441	Lo-fi music (half-CD)	Web audio, mobile apps, sound effects (music-centric)
24,000	240,000	480	Speech/music (streaming profiles)	Streaming codecs (AAC/Opus), VR/AR voice, compact music beds
32,000	320,000	640	Broadcast/Video (mid-band)	DVB/ATSC audio, FM broadcast chains, mini DV, voice/music mix
44,100	441,000	882	CD music standard	Music distribution (WAV/AIFF), consumer playback, CD audio
48,000	480,000	960	Video/Pro standard	Film/TV, DAW sessions for video, HDMI/SPDIF PCM baseline
96,000	960,000	1,920	Hi-res audio	Studio recording/mixing, archival, post-production

SAMD21G18A (APP045) has 256K Byte Flash with 10bits DAC

PIC32CX1025SG41128 (APP055) has 1024K Bytes Flash with 12 bits DAC

Buzzer

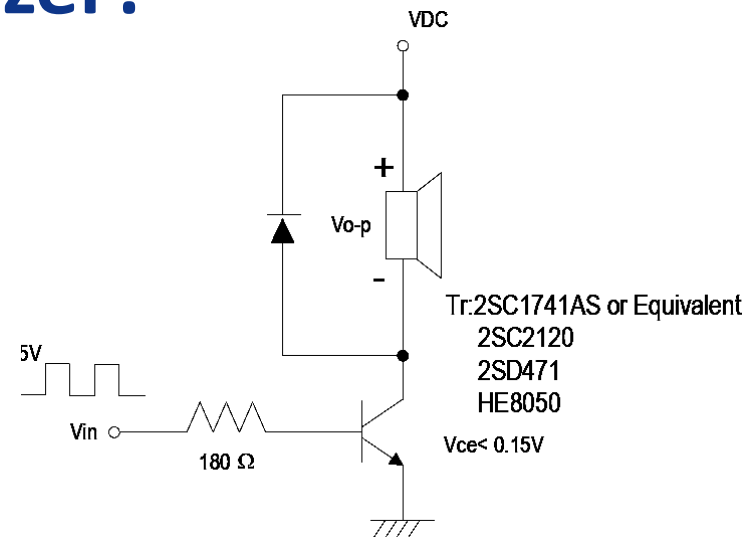
Buzzer

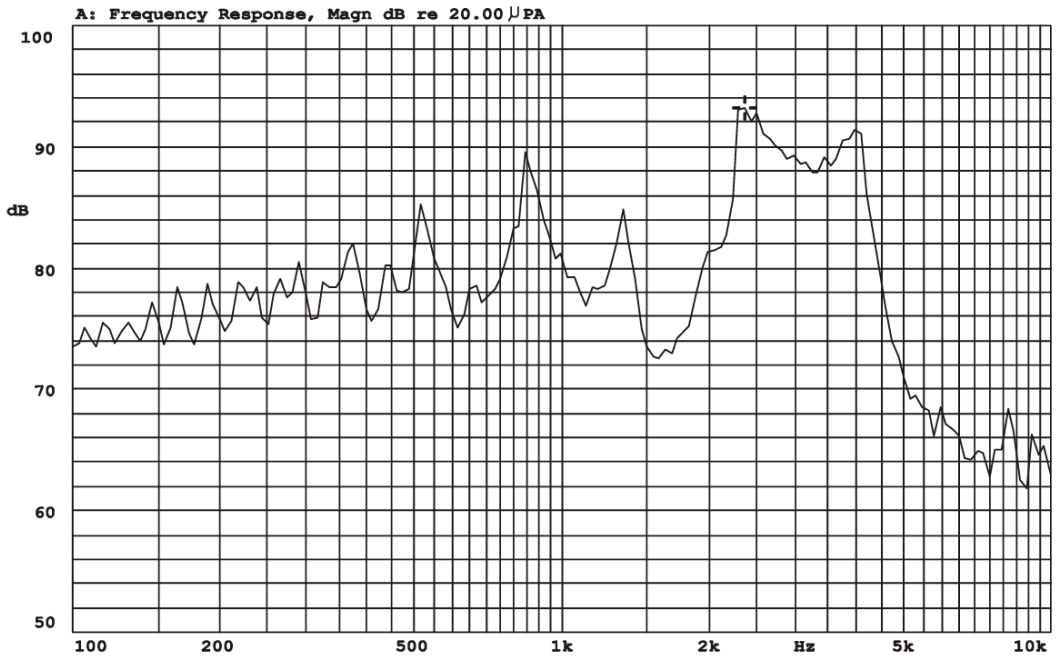
- **The AATC AC-1205D-LF is an electromagnetic (EM) buzzer designed for audio signaling applications. It operates at a frequency of 2.4 kHz, delivering a sound pressure level of 85 dBA at a distance of 10 cm. This buzzer is suitable for use in stable environments where a clear audio indication is required.**
- **Key Specifications:**
  - Frequency: 2.4 kHz
  - Sound Pressure Level: 85 dBA at 10 cm
  - Operating Voltage: 5 VAC
  - Current Consumption: 150 mA
  - Mounting Style: Through-hole
  - Dimensions: 12 mm (diameter) × 10 mm (height)
  - Termination Style: Pin

- **This buzzer is designed to be externally driven, meaning it requires an external oscillator or signal source to produce sound. It is commonly used in applications such as alarms, timers, and other devices where audible alerts are necessary.**

Is 8K PCM Audio Suitable for Use in a Buzzer?

- The AC-1205D buzzer works best at 2.4 kHz, while PCM audio at 8 kHz sample rate means the system can capture frequencies up to 4 kHz (Nyquist theorem).
- Since 2.4 kHz < 4 kHz, the buzzer's tone can be fully represented in 8 kHz PCM audio.
- 2.4 kHz is near the mid-high range of the 0–4 kHz band.
- Human ears are very sensitive around 2–4 kHz, so the buzzer sounds sharp and clear.
- PCM 8 kHz is also used in telephony (300–3400 Hz), which easily covers the buzzer's tone.
- In short: The 2.4 kHz buzzer sound fits perfectly into 8 kHz PCM audio and can be recorded or transmitted clearly.

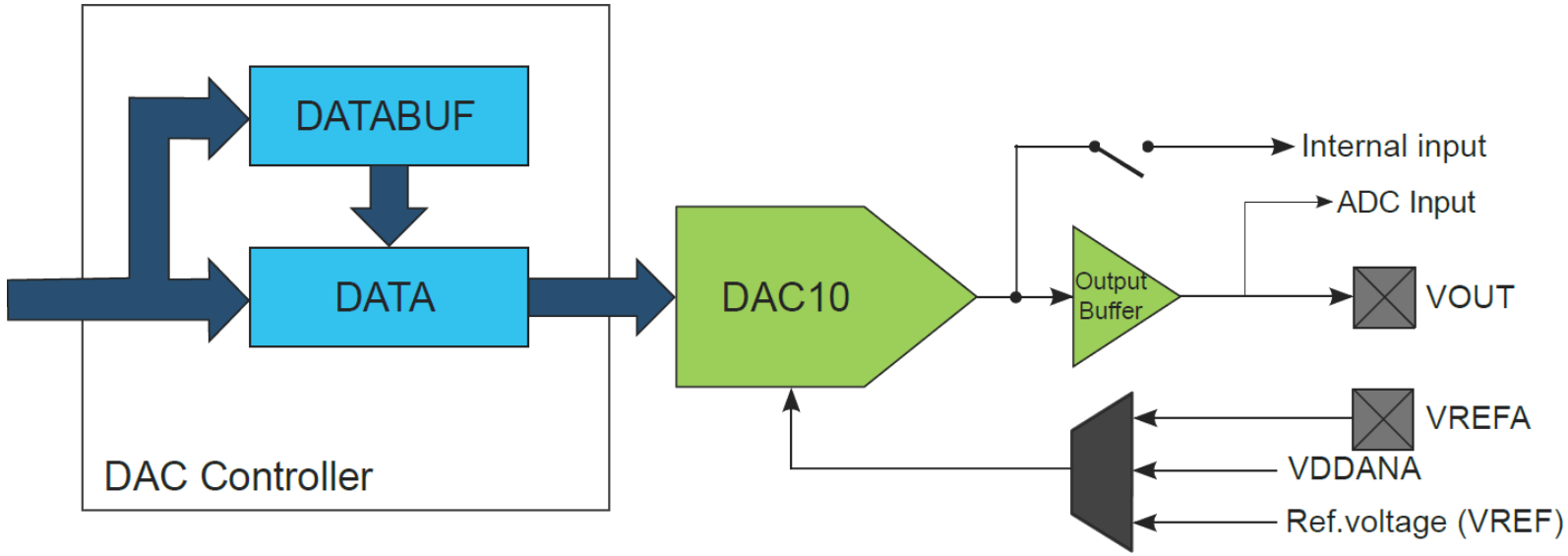




DAC (Digital to Analog Converter)

Digital-to-Analog Converter (DAC) - SAMD21

- One channel analog output, 10-bits resolution. Up to 350ksps.
- Multiple trigger sources, High-drive capabilities and DMA support.
- Output can be used as input to the Analog Comparator (AC) or Analog-to-Digital Converter (ADC).

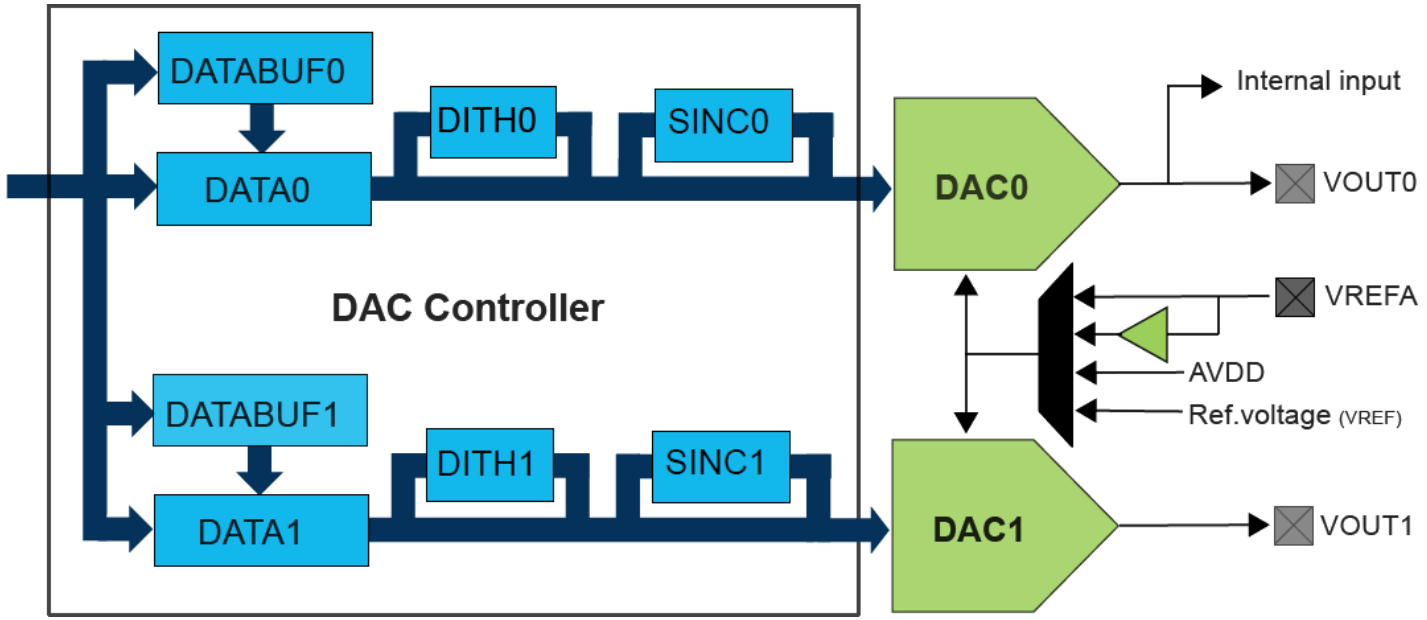


$$VOUT = \frac{DATA}{2^{10} - 1} \times V_{Ref} = \frac{DATA}{1023} \times V_{Ref}$$

MICROCHIP

Digital-to-Analog Converter (DAC) - PIC32CX-SG

- Two independent analog outputs, or single differential output.
- 12-bits resolution. Up to 1 Msps. Hardware support for 16-bits with dithering.
- Multiple trigger sources, high-drive capability and DMA support.
- DAC0 output can also be used as an internal input.



$$VOUT_x = \frac{DATA_x}{2^{12} - 1} \times V_{Ref} = \frac{DATA_x}{4095} \times V_{Ref}$$

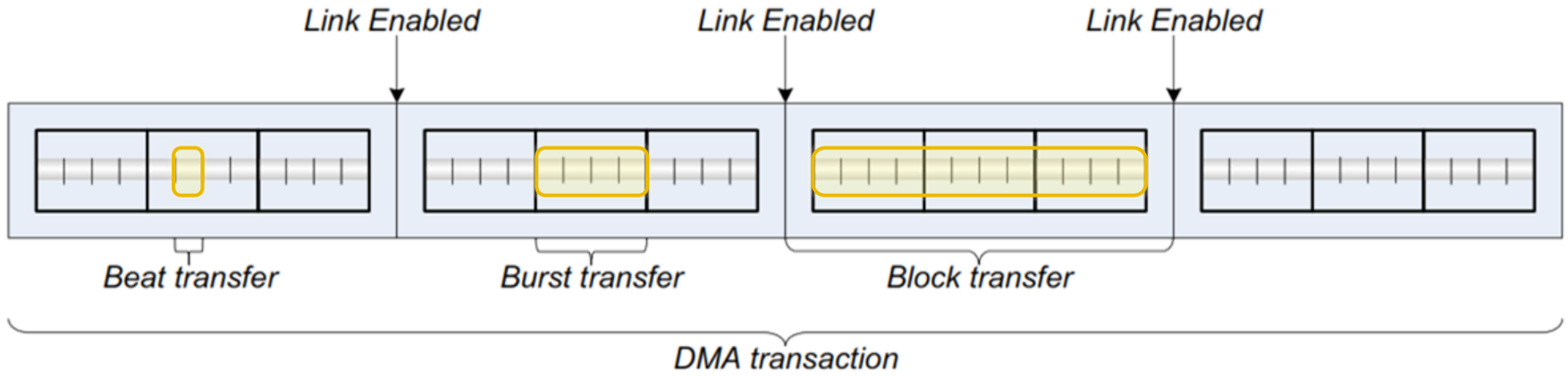
Direct Memory Access Controller (DMAC)

Direct Memory Access Controller (DMAC)

- Data transfer types: Peripheral↔peripheral, Peripheral↔memory, Memory↔peripheral, Memory↔memory.
- Transfer trigger sources: Software, Events from Event System, Dedicated peripheral requests.
- SRAM-based transfer descriptors: Single-transfer descriptor; multi-buffer / circular buffer by linking descriptors.
- Channel count & per-channel handling:
  - Pultiple independent channels with automatic descriptor fetch, suspend/resume per channel.
  - **SAMD21 up to 12 channels and PIC32CX-SG up to 32 channels.**
- Flexible arbitration: 4 configurable priority levels per channel; fixed or round-robin within each level.
- Single block transfer size: 1 to 256 KB.
- Addressing modes: Static and configurable increment schemes.
- Optional interrupt generation: On block complete, on error, on channel suspend.
- Event inputs:
  - Per-channel event inputs selectable to trigger normal/periodic/conditional transfers or to suspend/resume channels.
  - **SAMD21 has 4 event inputs (for 4 LSB channels) and PIC32CX-SG has 8 event inputs (for 8 LSB channels)**
- Event outputs: Per-channel event outputs selectable on AHB / block / transaction complete.
- Error management: Write-back memory section per channel to store ongoing descriptor transfer state.
- CRC support: Software-selectable CRC polynomial (CRC-16 / CRC-32).
- DMA features: High throughput, hardware-assisted transfers and close CPU offload.

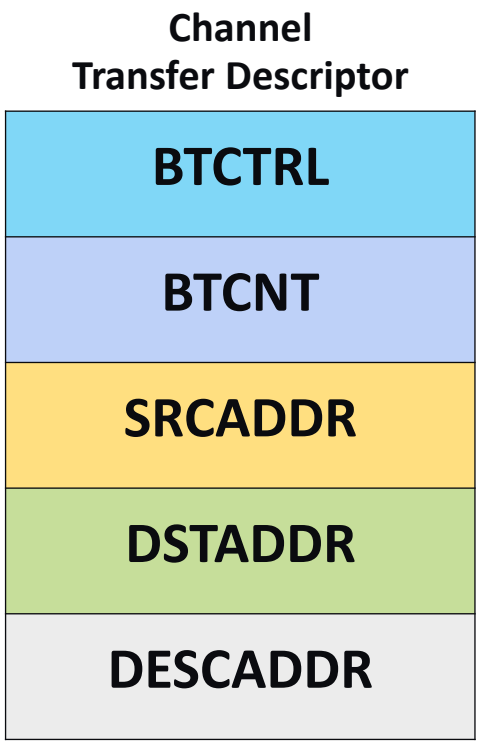
Transfer Types

- **Beat transfer**
  - Size of data length for one of DMAC transfer.
  - 8-bits, 16-bits or 32-bits
- **Burst transfer**
  - n Beat transfer at same time, Burst is "**atomic**" transfer.
  - SAM M0+ families not support.
- **Block transfer**
  - Amount of beat can be transfer at one transfer descriptor.
  - Max. number is 2^{16} * beat => 64K * 4 Bytes = 256K Bytes
- **Transaction**
  - DMAC can only use one or link several transfer descriptors.
  - A transaction is complete transfer of all blocks. (one or all link)

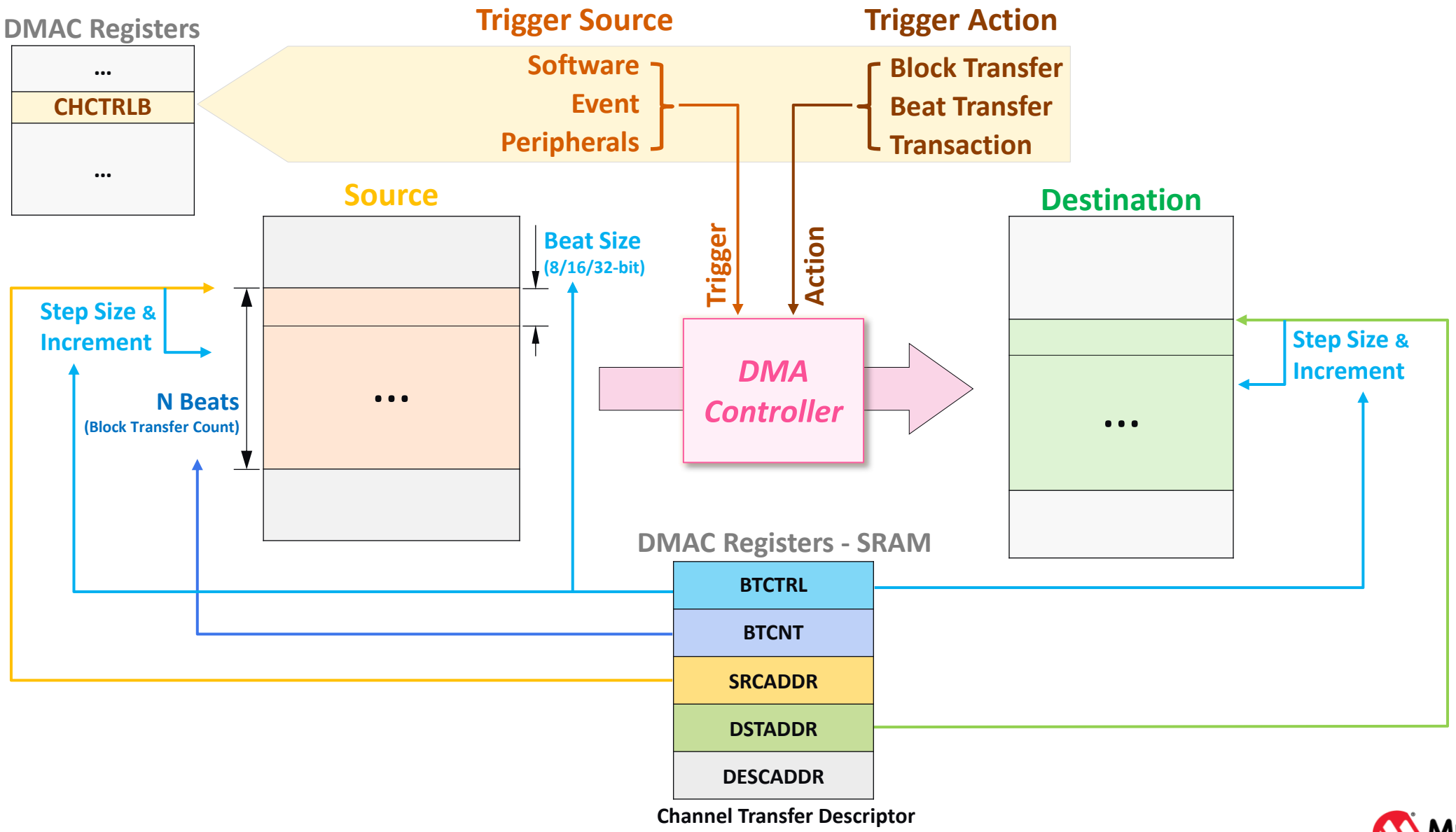


Transfer Descriptor

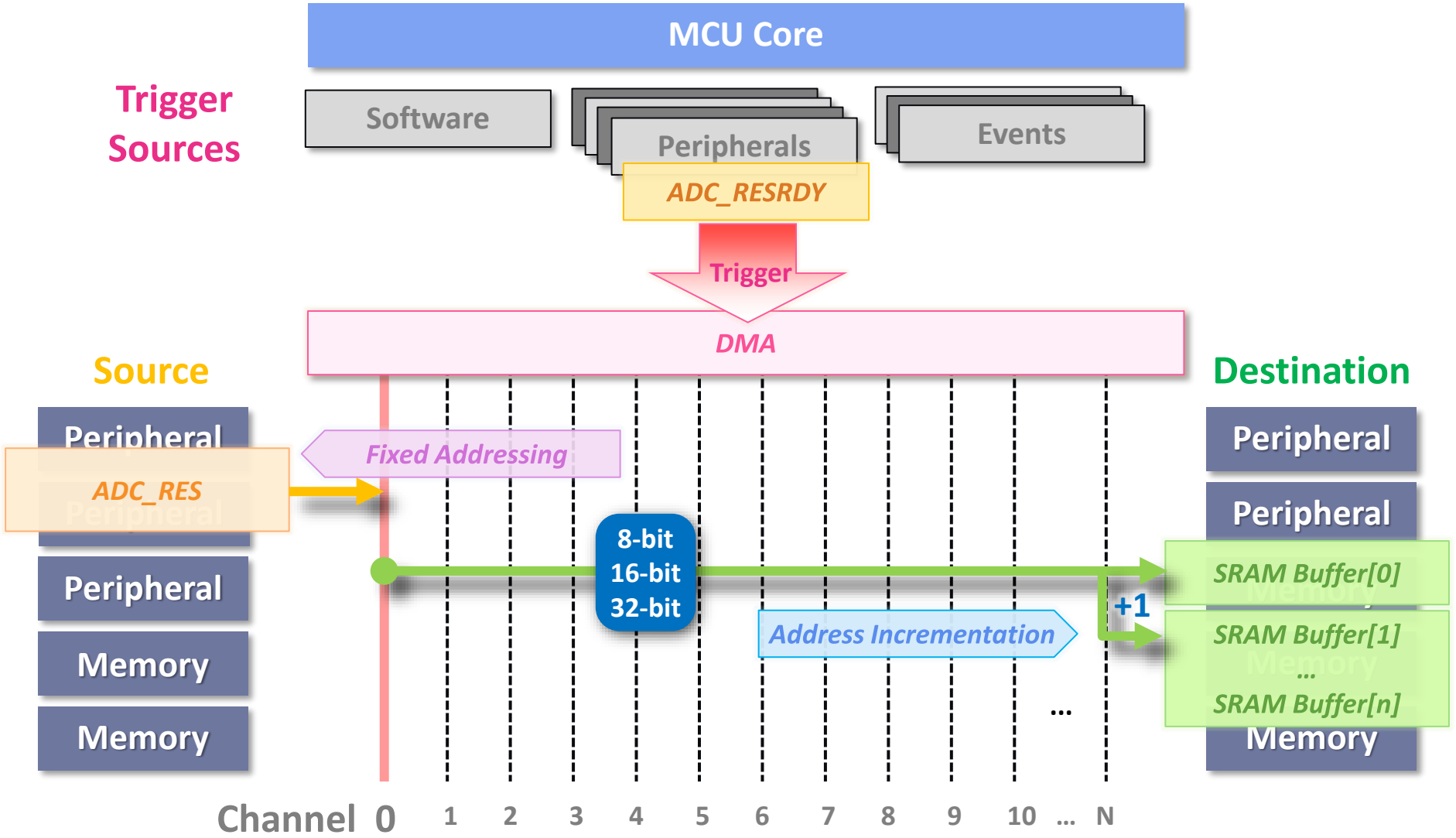
- **The “Transfer Descriptor” use to describe that which data and how many data be transfer when DMAC be triggered.**
- **Channel Transfer Descriptor include below:**
  - Operation mode
  - Beat size, Block Address increment
  - Interrupt action, Event out action
  - Block Size
  - Data from and to
  - Amount of beat to be transfer
  - Where is next Transfer Descriptor



Block diagram of Transaction



DMAC Transfer Example



Event System (EVSYS)

Event System

- **The Event System (EVSYS) allows autonomous, low-latency and configurable communication between peripherals.**

- **Role:**
  - Peripherals that respond to events are called event users.
  - Peripherals that generate events are called event generators.

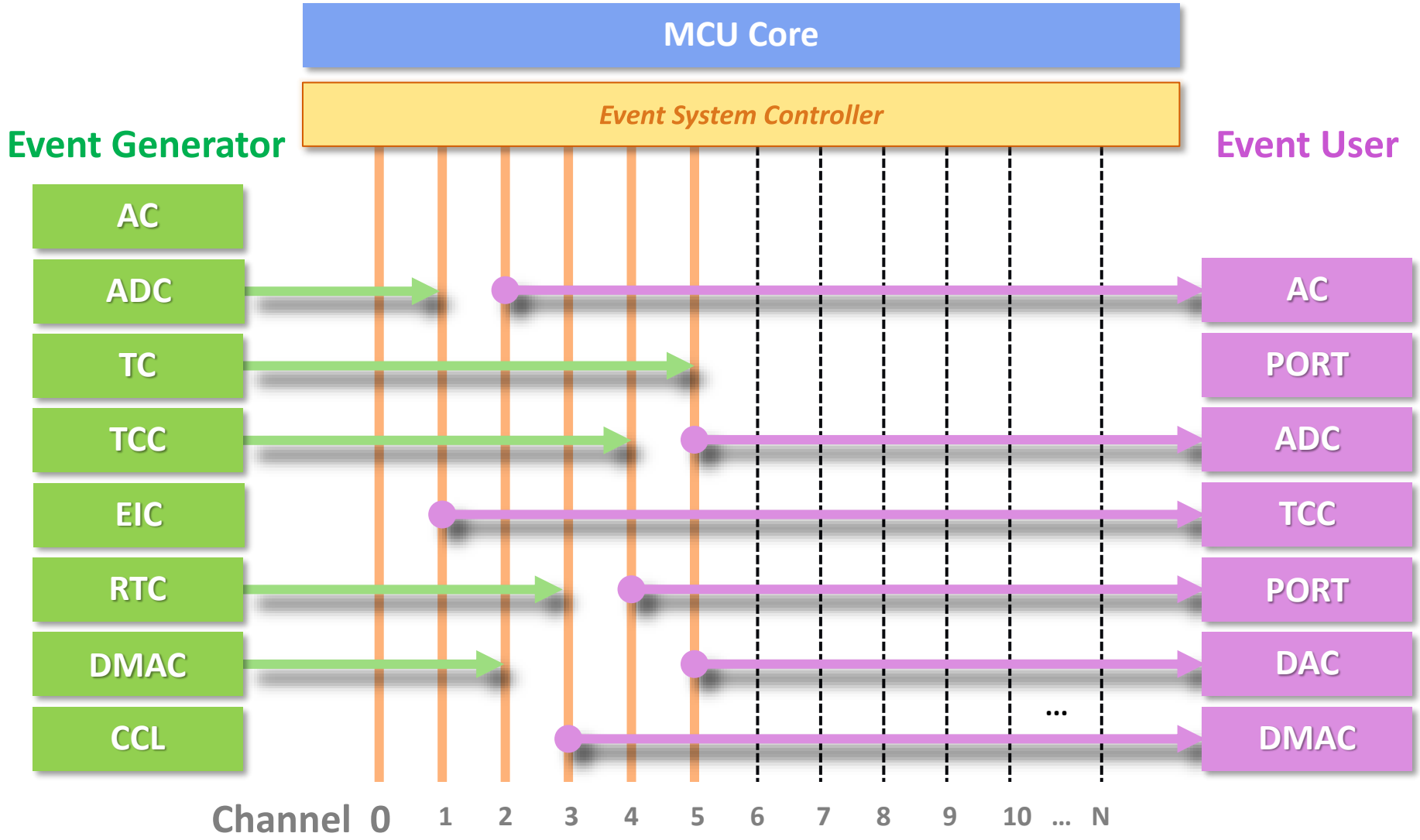
- **A peripheral can have one or more event generators and can have one or more event users, also.**

- **Communication is made without CPU intervention and without consuming system resources such as bus or RAM bandwidth. This reduces the load on the CPU and other system resources, compared to a traditional interrupt-based system.**

Event System

- **Common Features**
  - Peripherals can be event generators, event users, or both.
  - SleepWalking and interrupt for operation in sleep modes.
  - Software event generation.
- **SAMD21**
  - 12 configurable event channels
  - 74 event generators & 29 event users
- **PIC32CX-SG**
  - 32 configurable event channels
  - 119 event generators & 67 event users
  - Optional Static or Round-Robin interrupt priority arbitration

Event System Connection Example



PCM Audio Playback

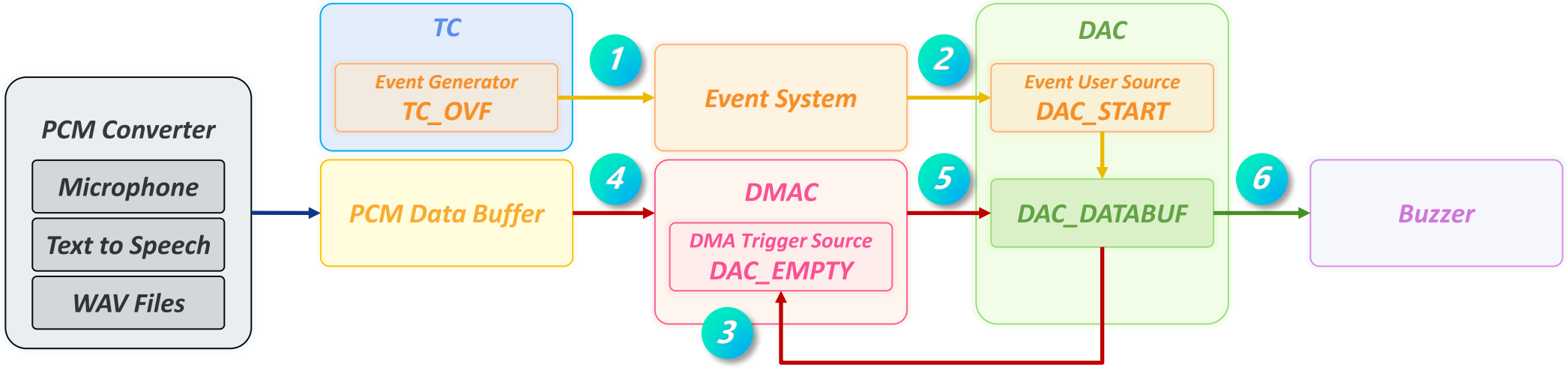
Using TC, Event System, DAC, and DMAC

MICROCHIP

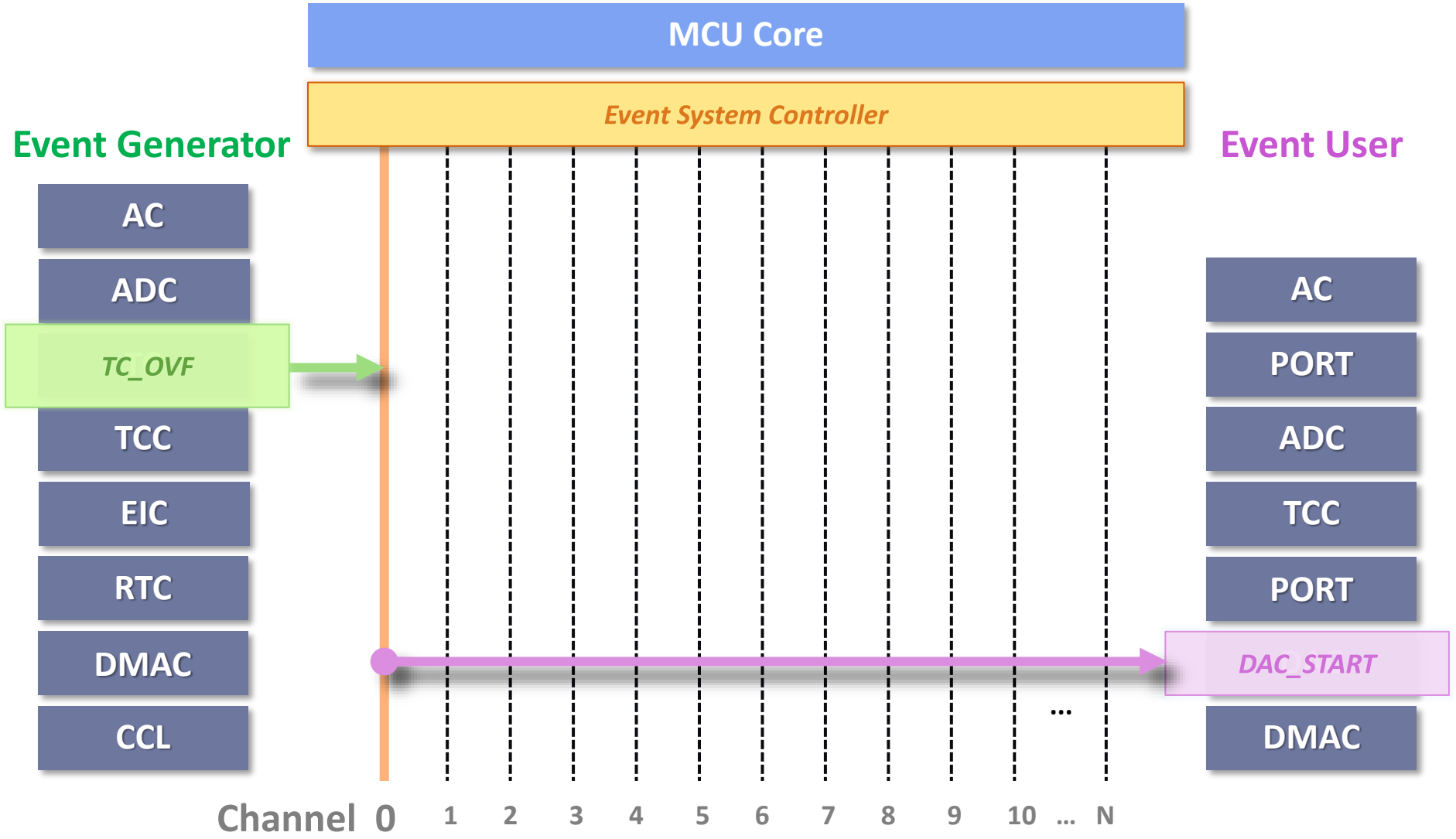
PCM Audio Playback Using TC, Event System, DAC, and DMAC

1. The TC is configured to the PCM sample rate period and generates an event.
2. The Event System triggers a DAC conversion.
3. A DAC buffer-empty condition triggers a DMAC transfer.
4. The DMAC transfers data from the PCM data buffer (with incremented source address).
5. The DMAC writes the data to the DAC_DATABUF register (with fixed destination address).
6. The DAC outputs the conversion result to the Buzzer.

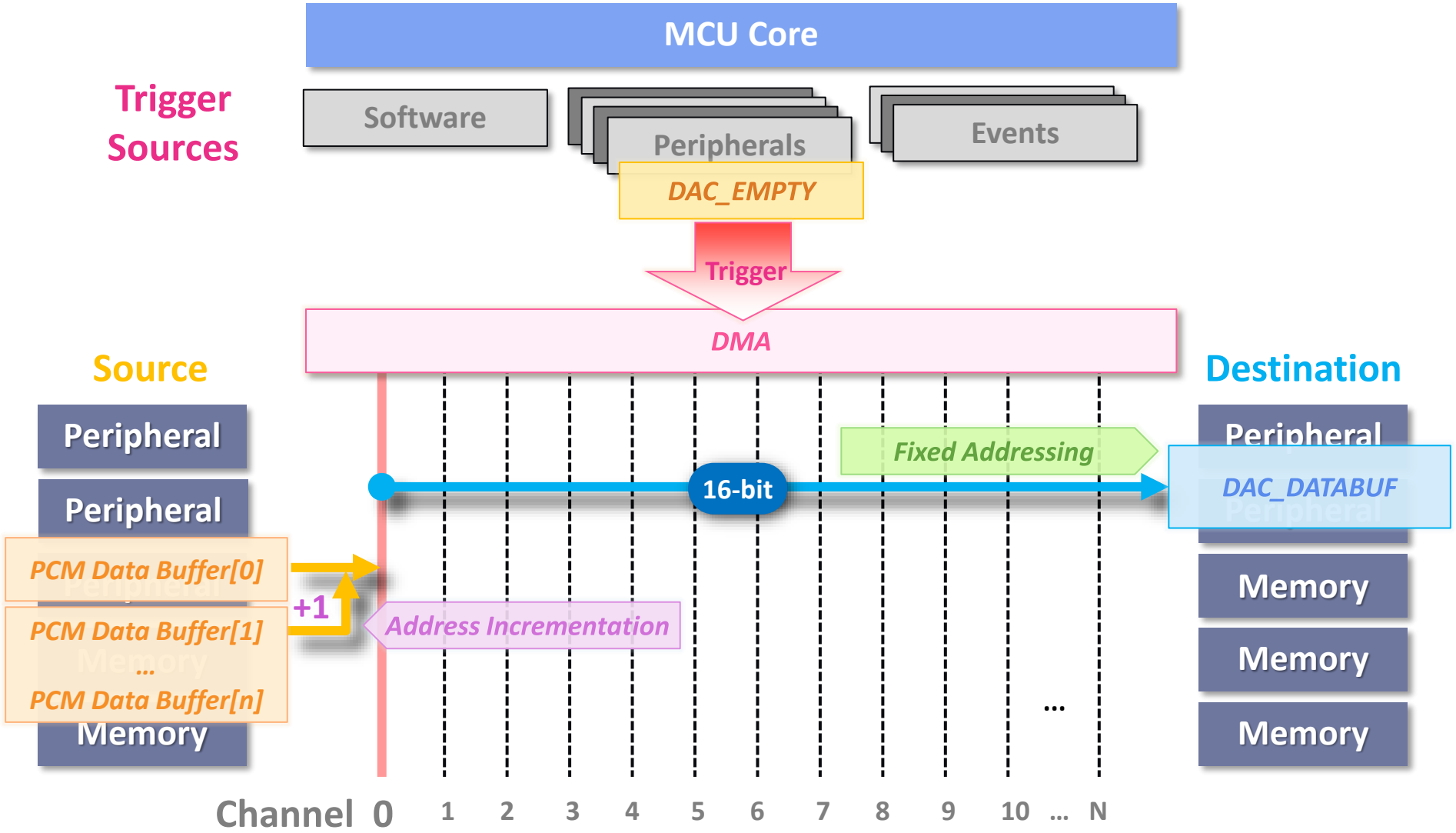
The sequence then repeats, waiting for the TC to generate the next event.



PCM Audio Playback Using TC, Event System, DAC, and DMAC



PCM Audio Playback Using TC, Event System, DAC, and DMAC

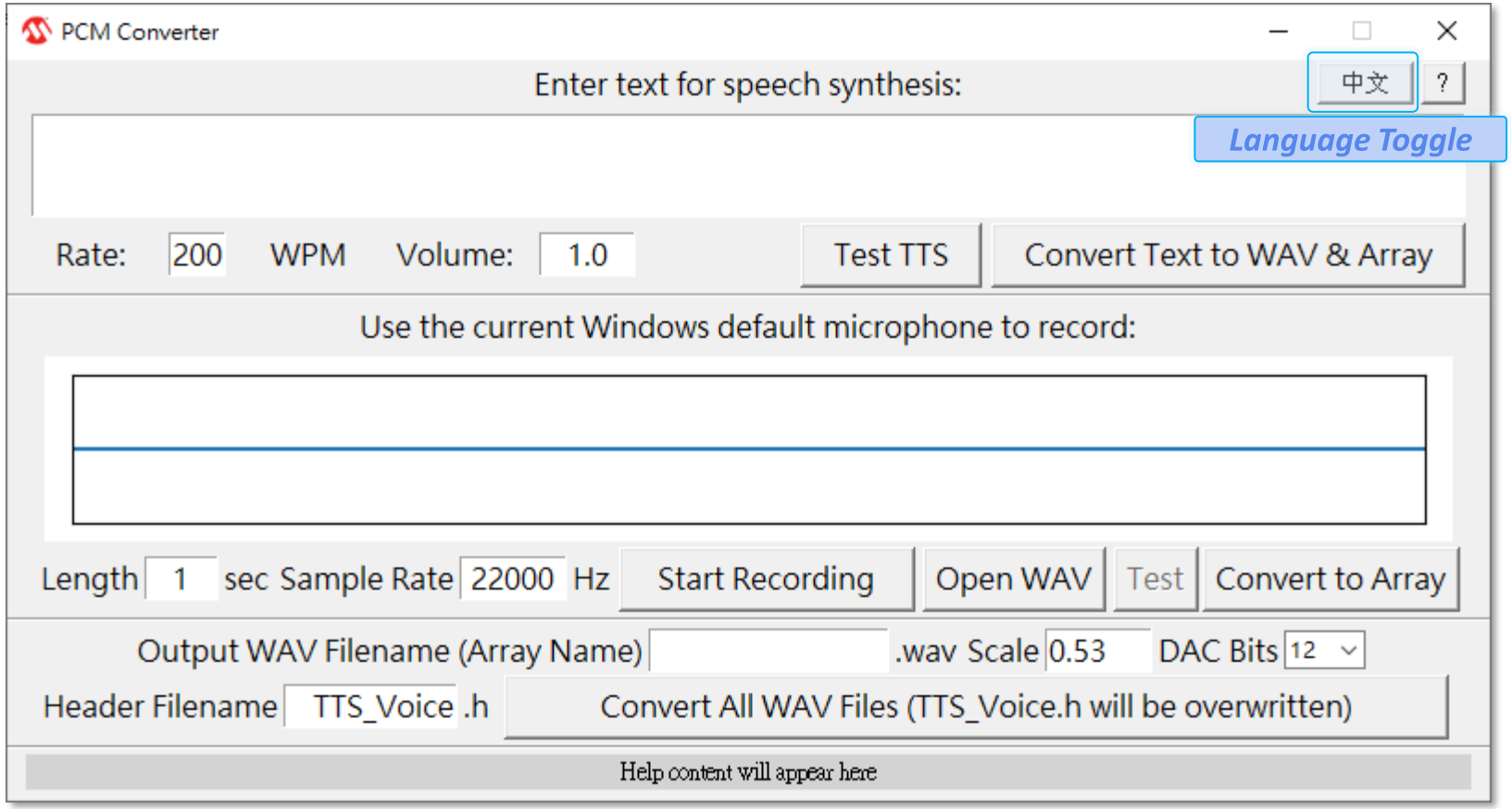


PCM Converter

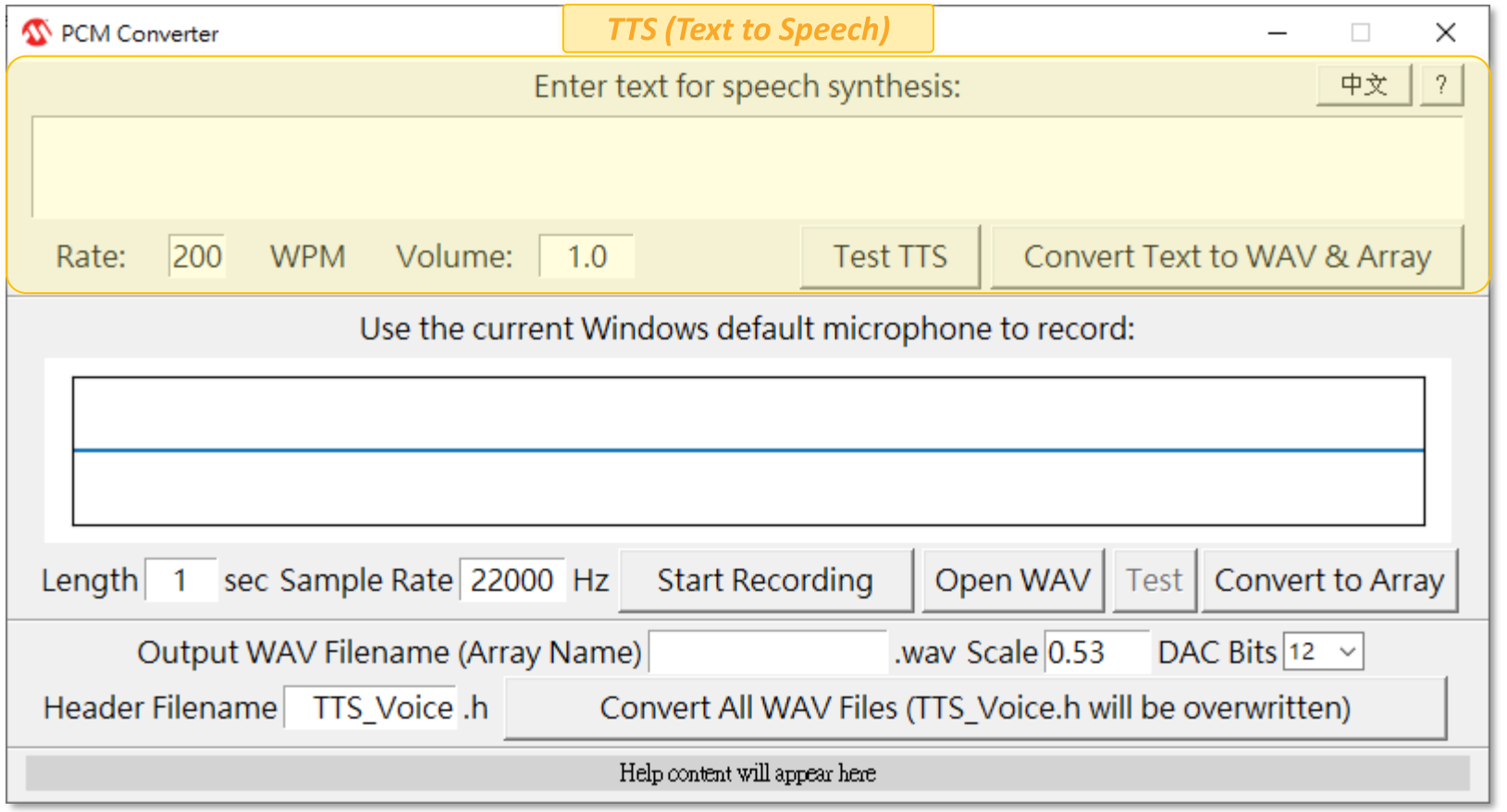
Fantasy Application to Generate the PCM Audio Data For Embedded

MICROCHIP

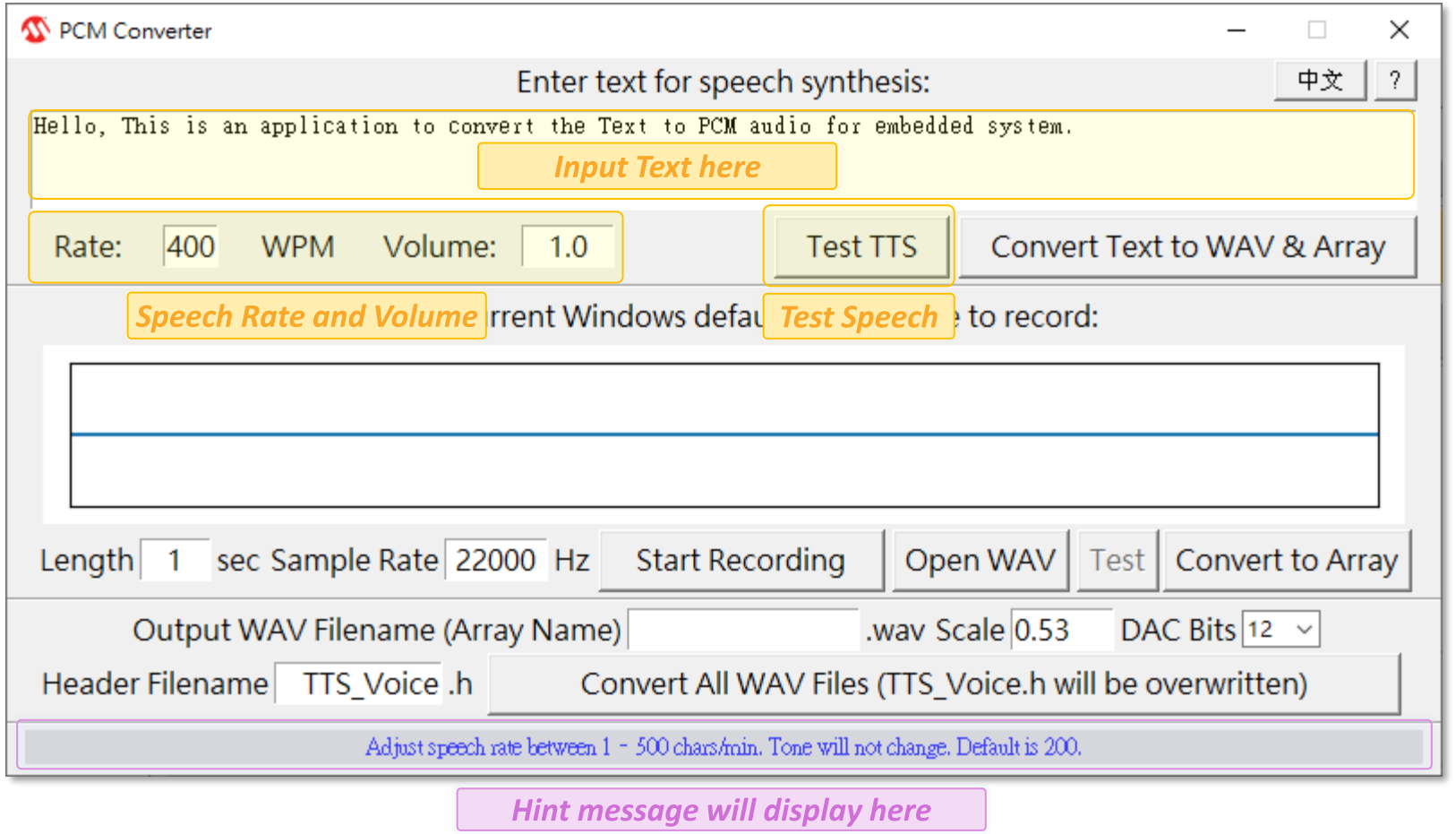
PCM Converter



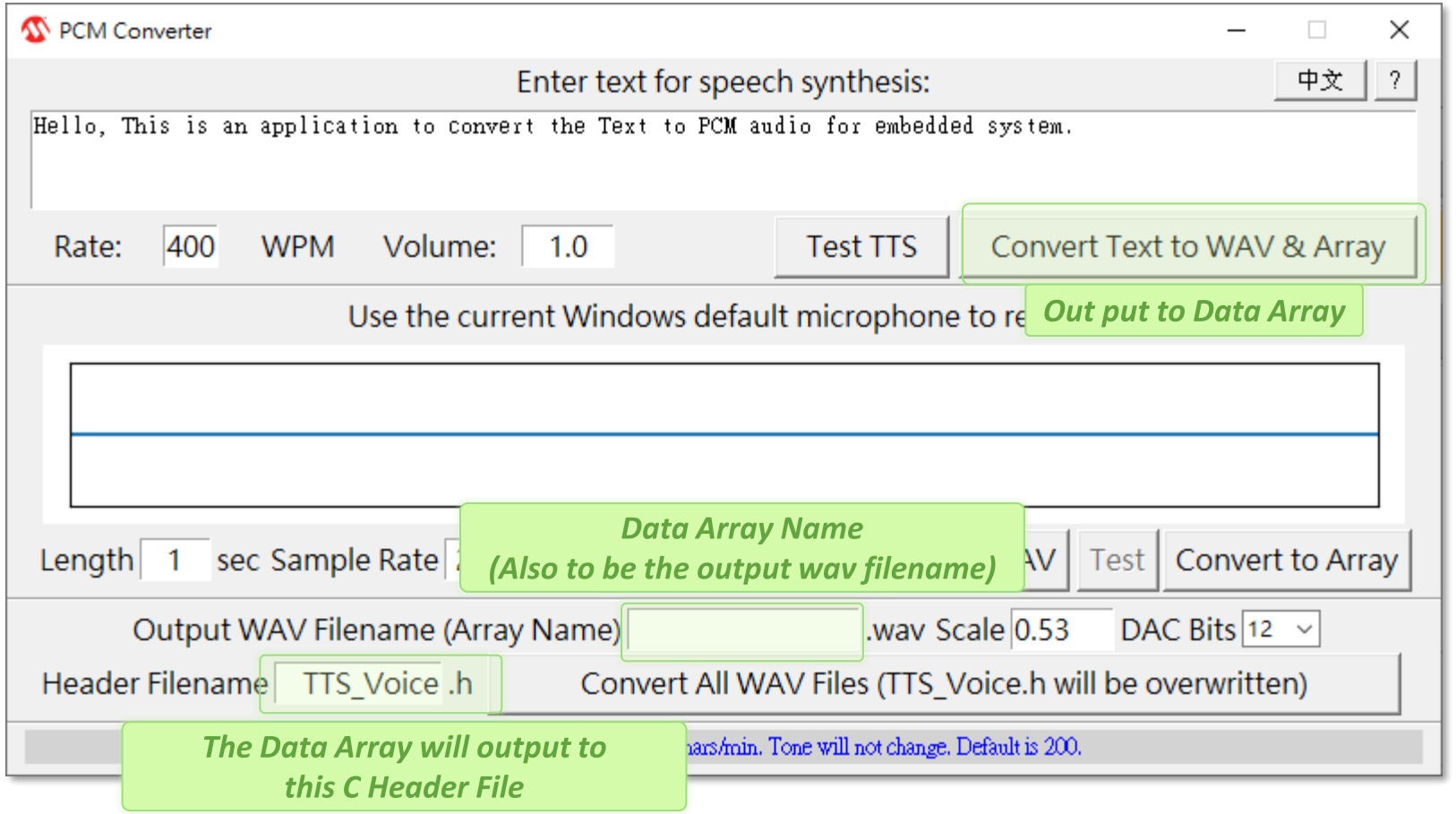
PCM Converter - TTS (Text to Speech)



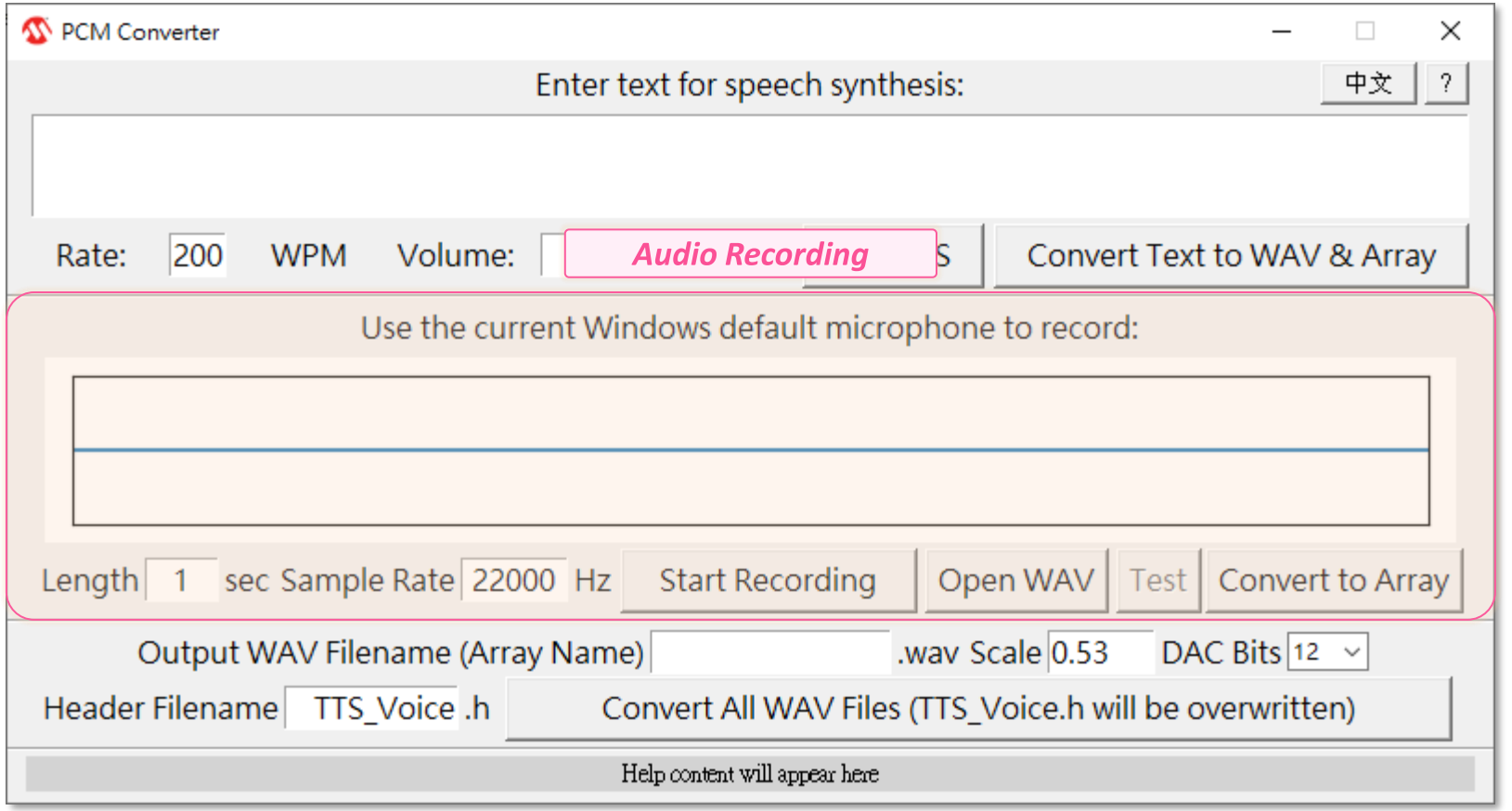
PCM Converter - TTS (Text to Speech)



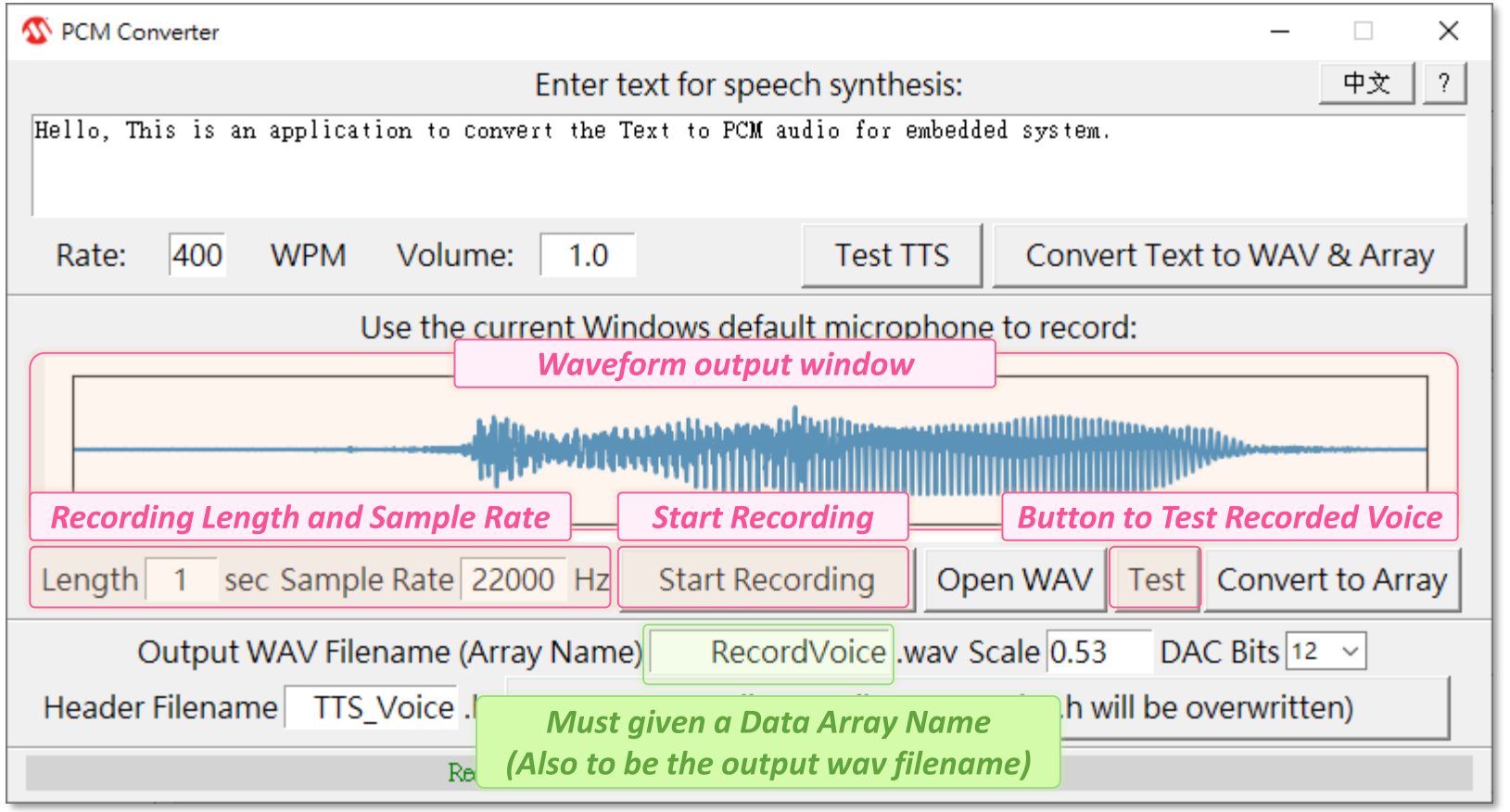
PCM Converter - TTS (Text to Speech)



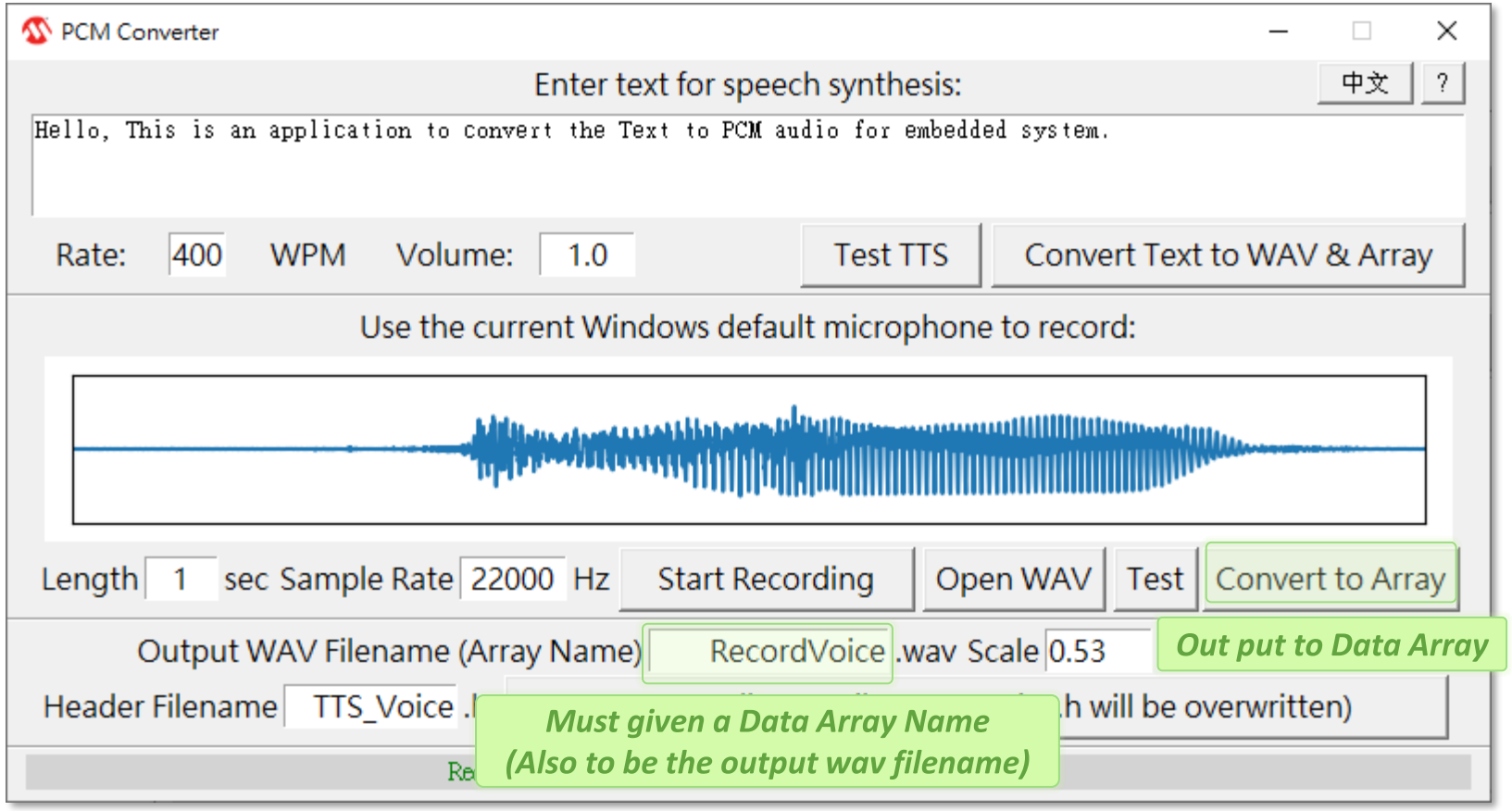
PCM Converter – Audio Recording



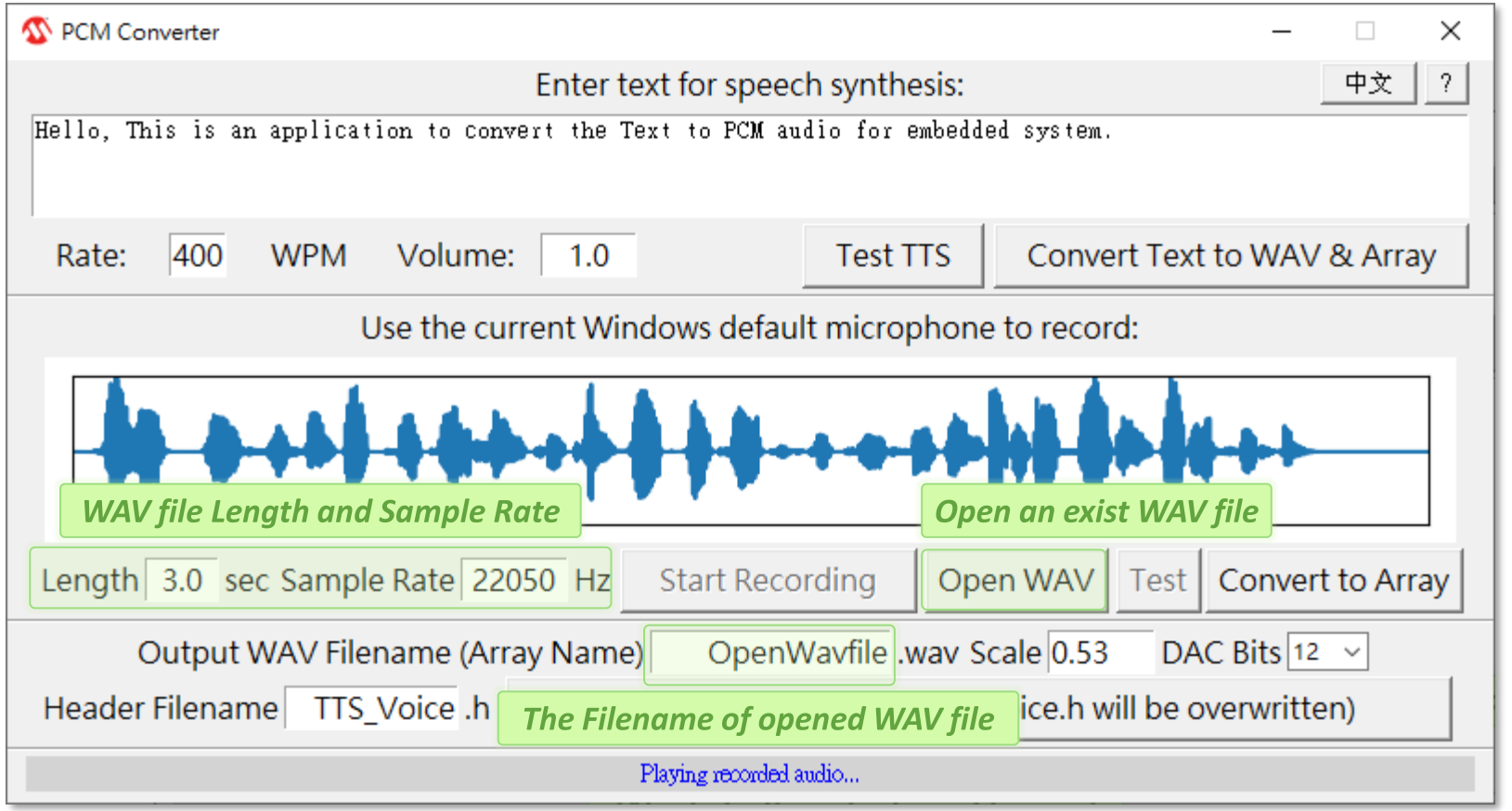
PCM Converter – Audio Recording



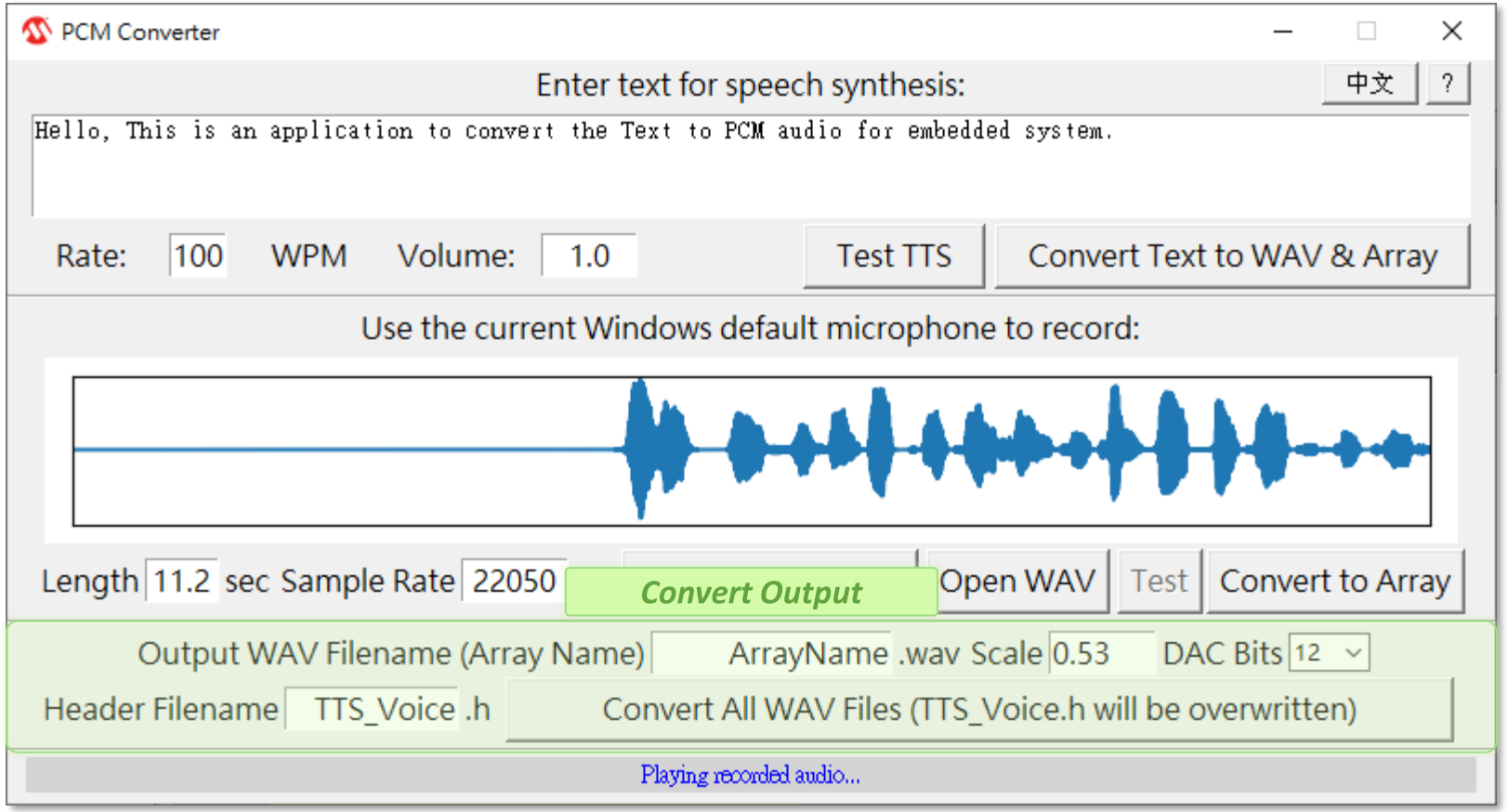
PCM Converter – Audio Recording



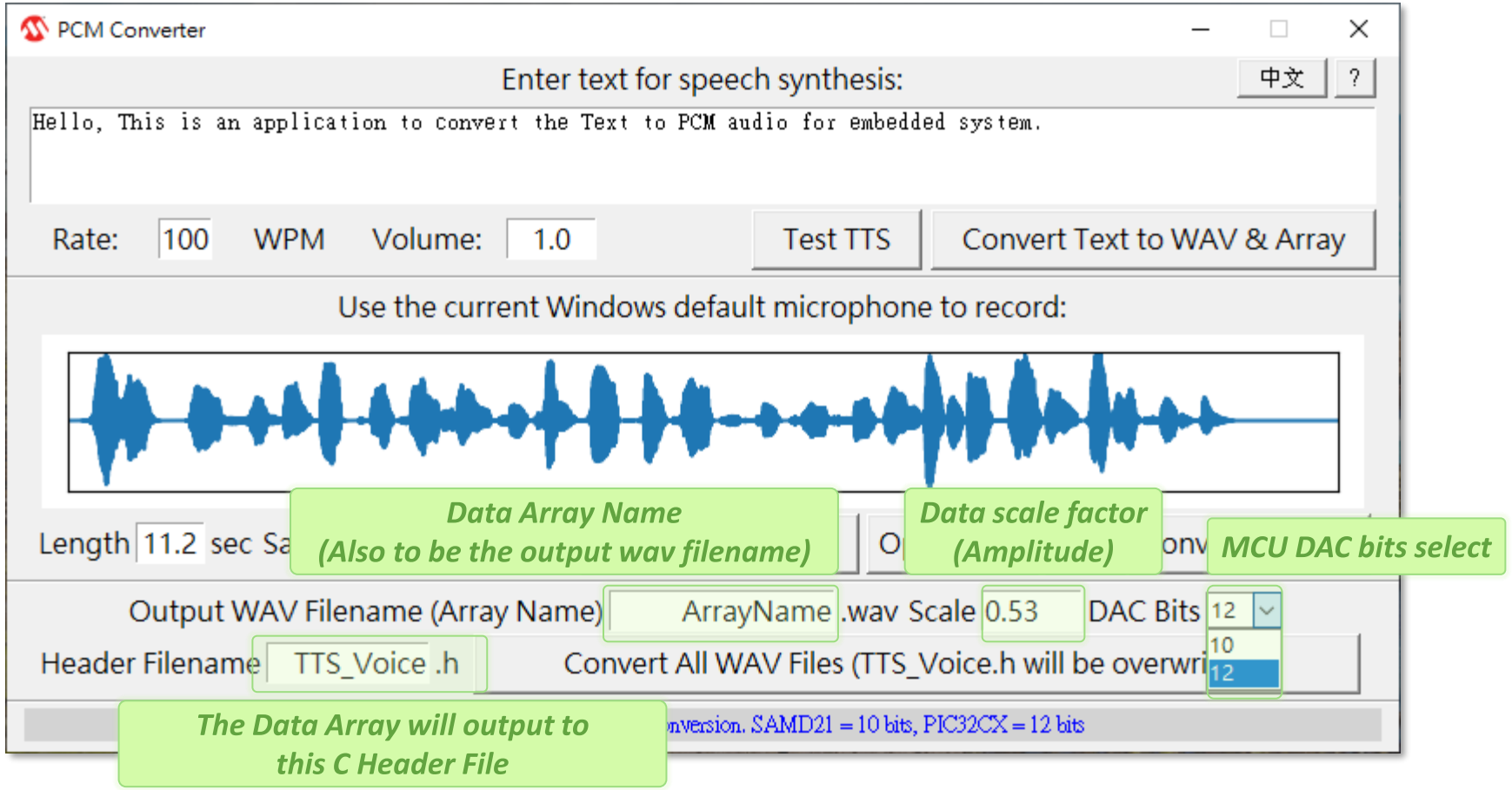
PCM Converter – Audio Recording



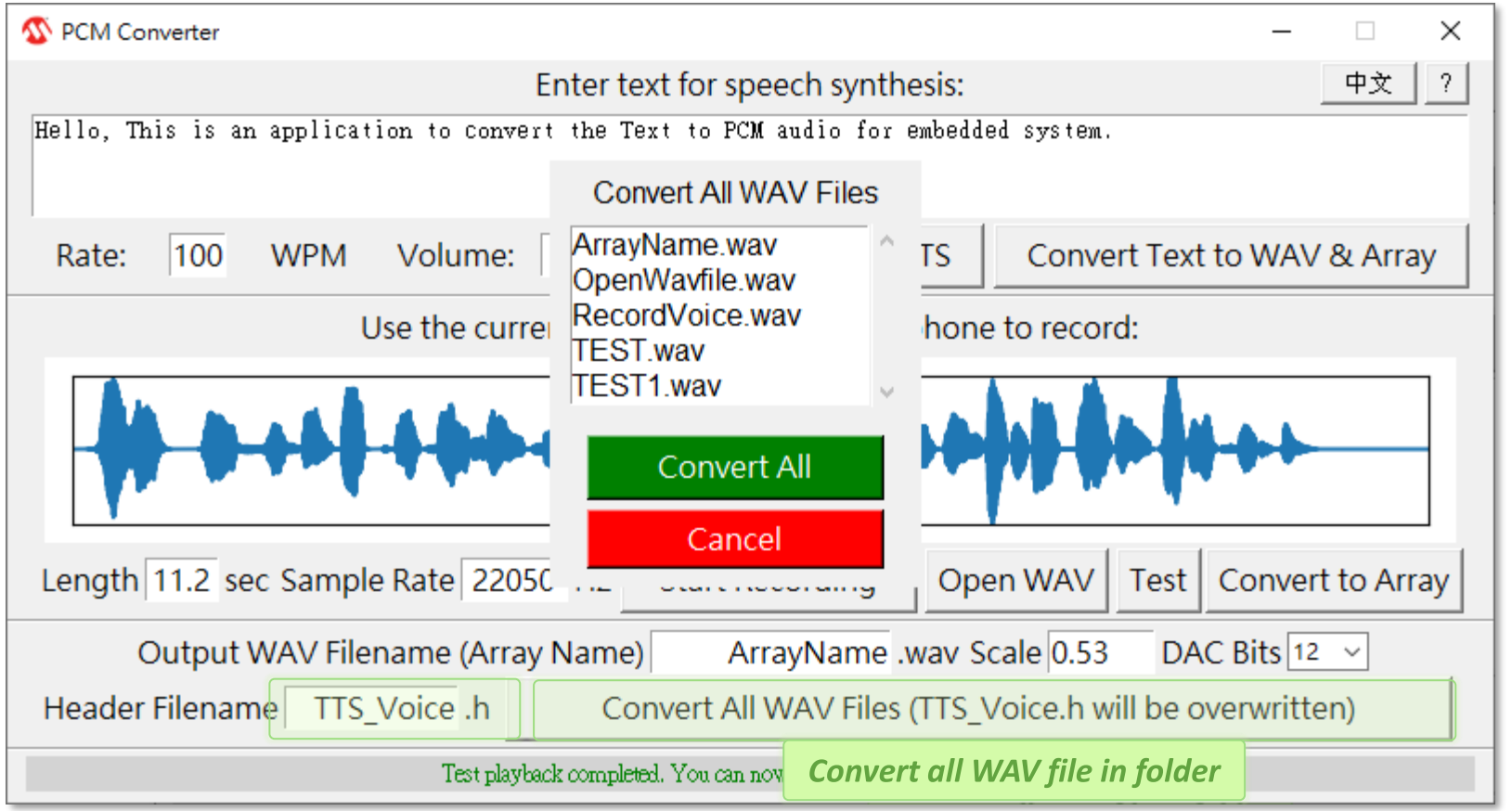
PCM Converter – Convert Output



PCM Converter – Convert Output



PCM Converter – Convert Output

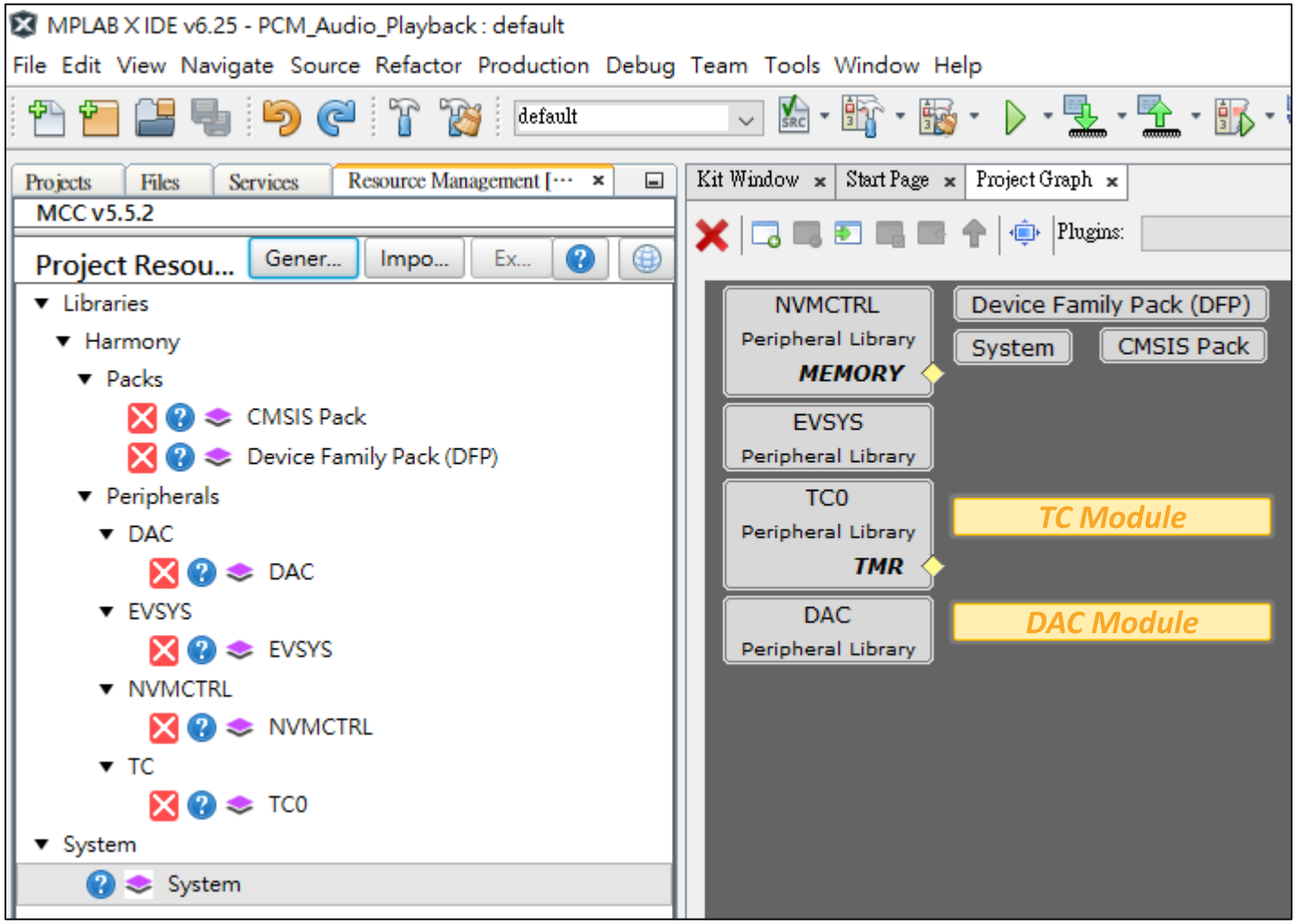


The MCC Configuration

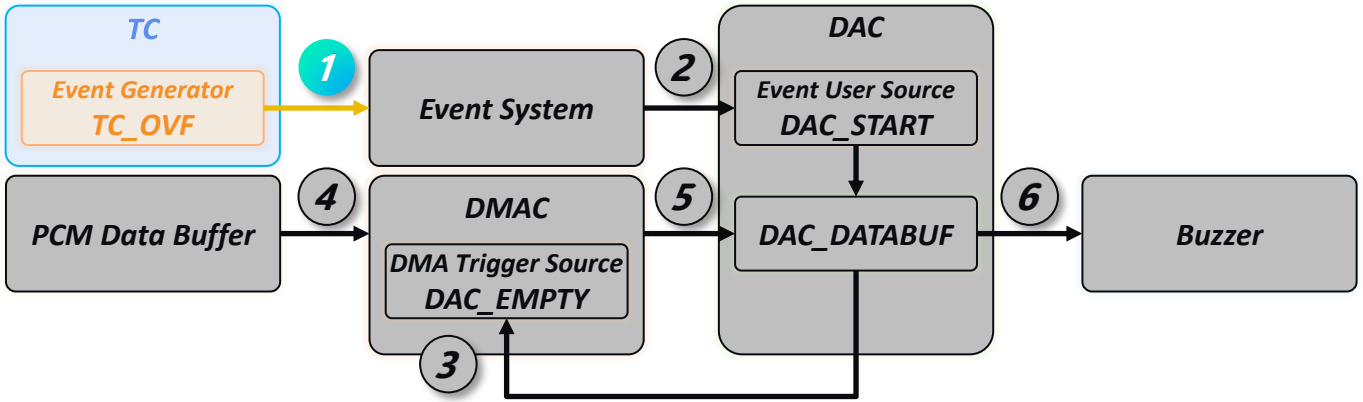
Fantasy Application to Generate the PCM Audio Data For Embedded

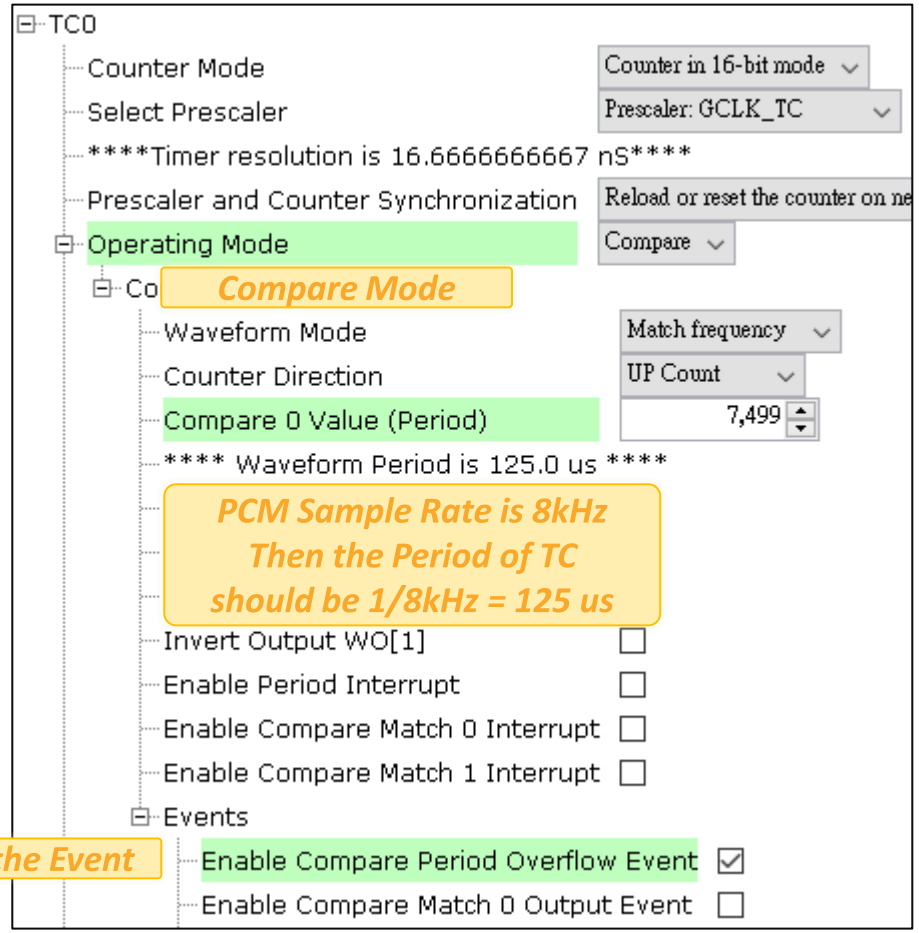
The MCC Configuration

- **Add TC and DAC modules to project.**

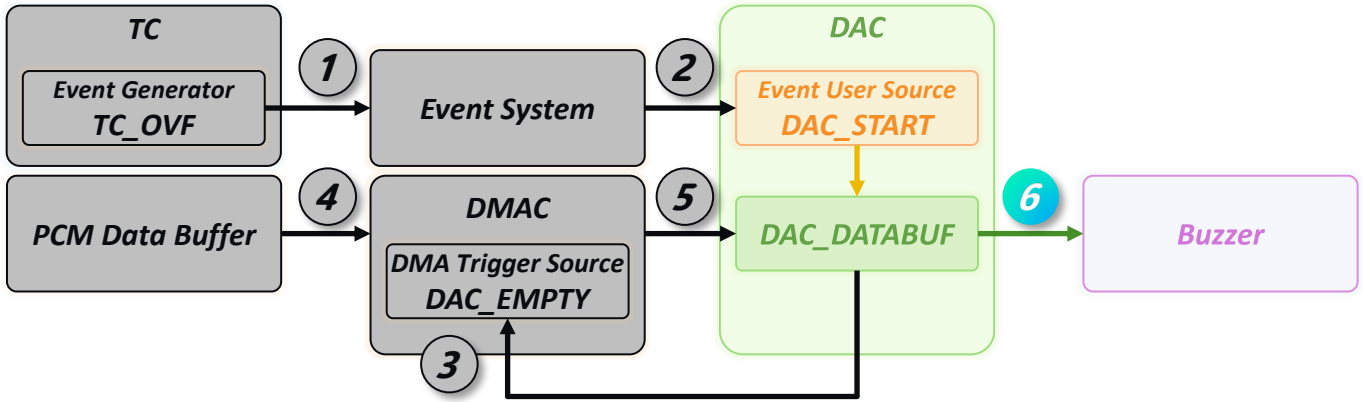


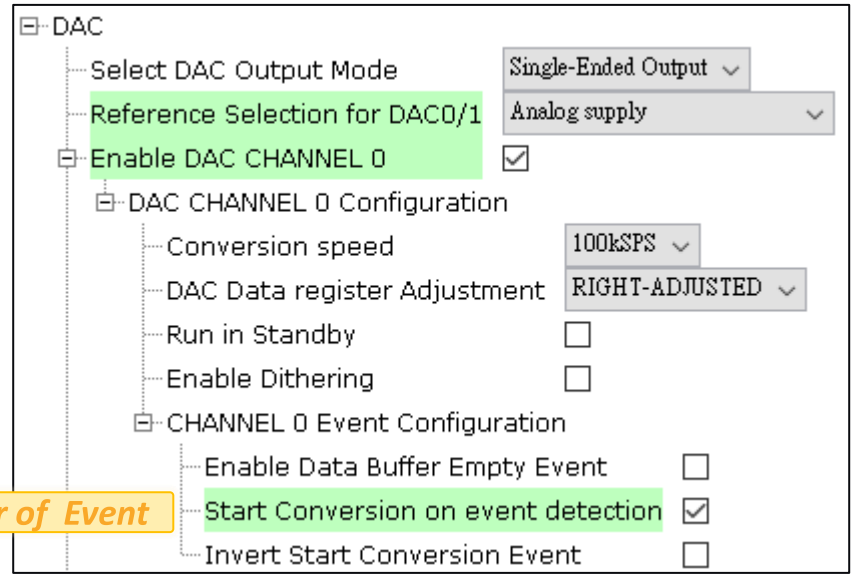
The MCC Configuration





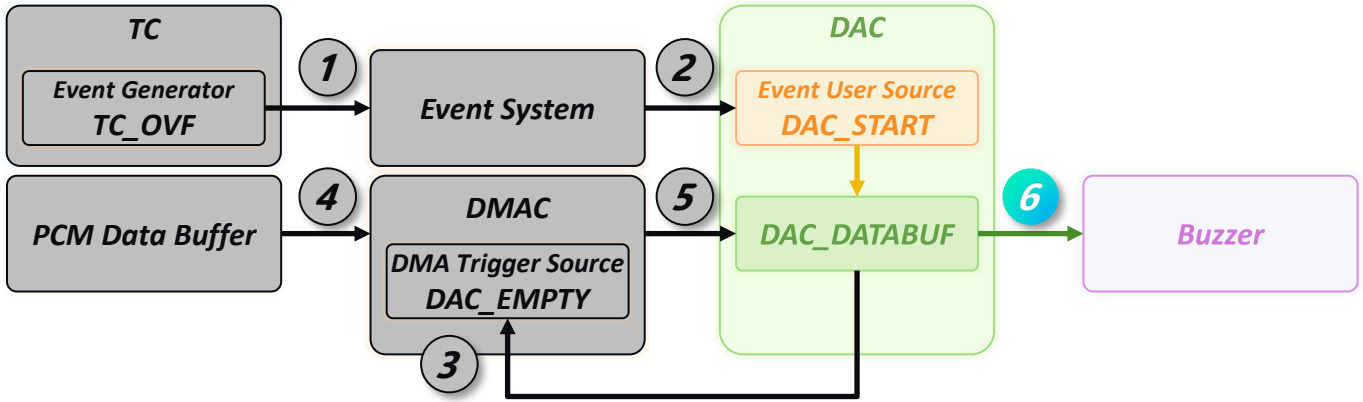
The MCC Configuration





To be User of Event

The MCC Configuration



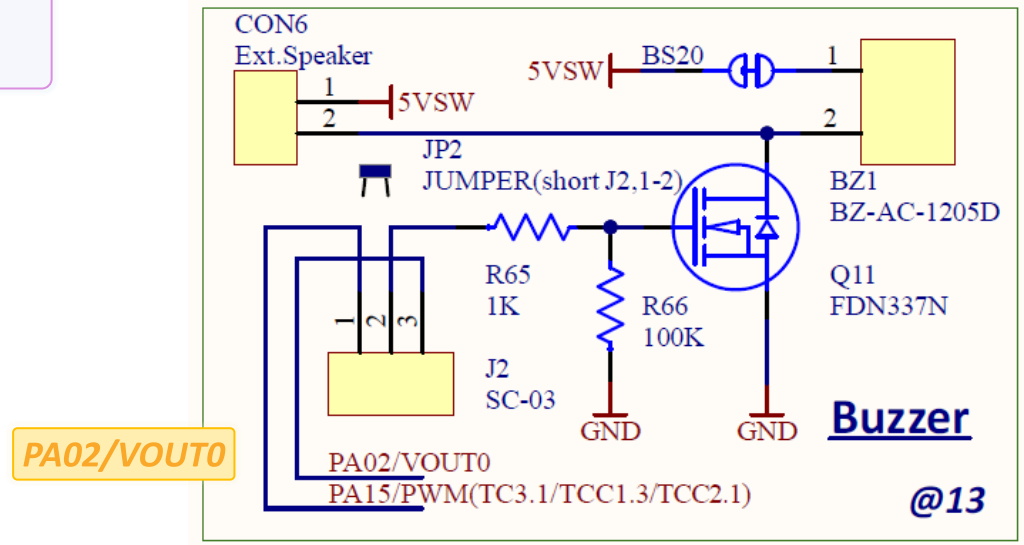
MCC | Pin Diagram | Pin Table | Pin Settings

Order: Ports | Table View | Easy View

Pin Number	Pin ID	Custom Name	Function	Mode
1	PA00		Available	Digital
2	PA01		Available	Digital
	PA02		DAC_VOUT0	Analog
10	PA03		Available	Digital
21	PA04		Available	Digital
22	PA05		Available	Digital
23	PA06		Available	Digital

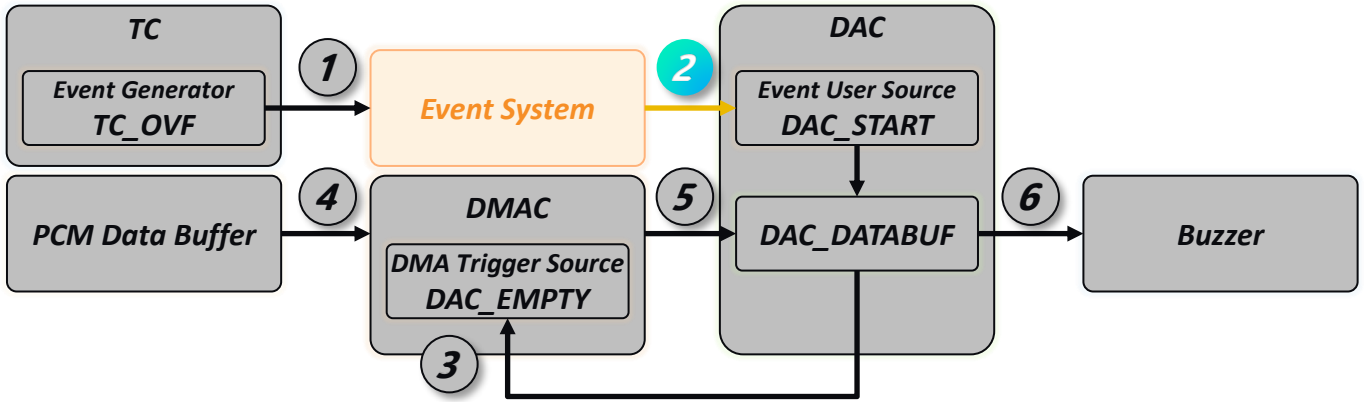
DAC_VOUT0

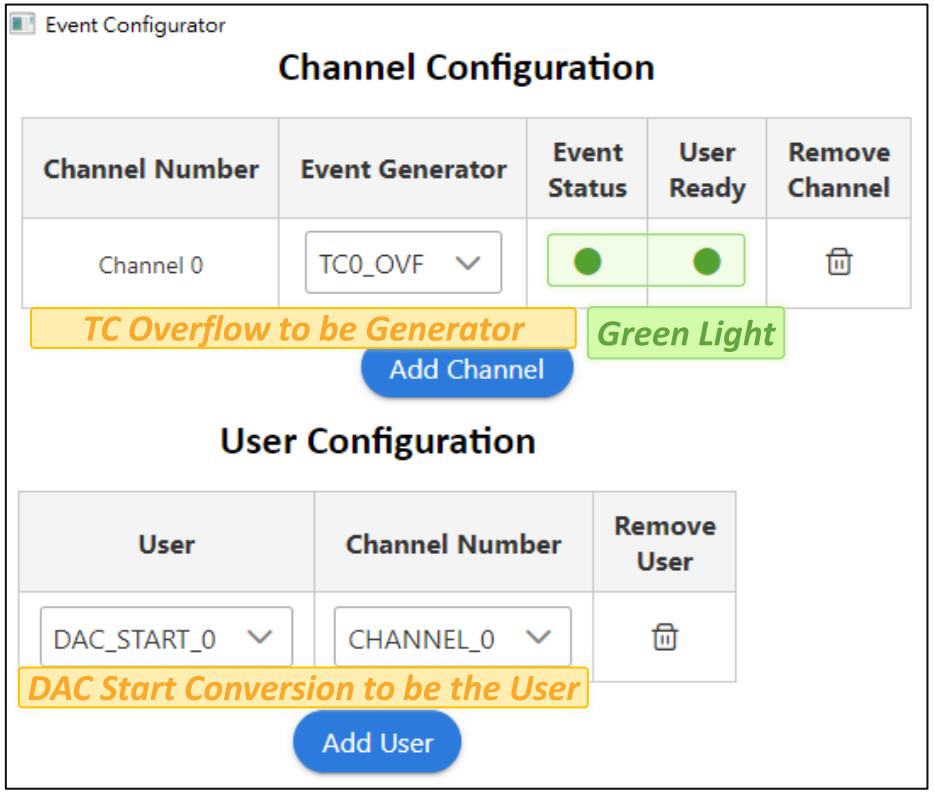
PA02



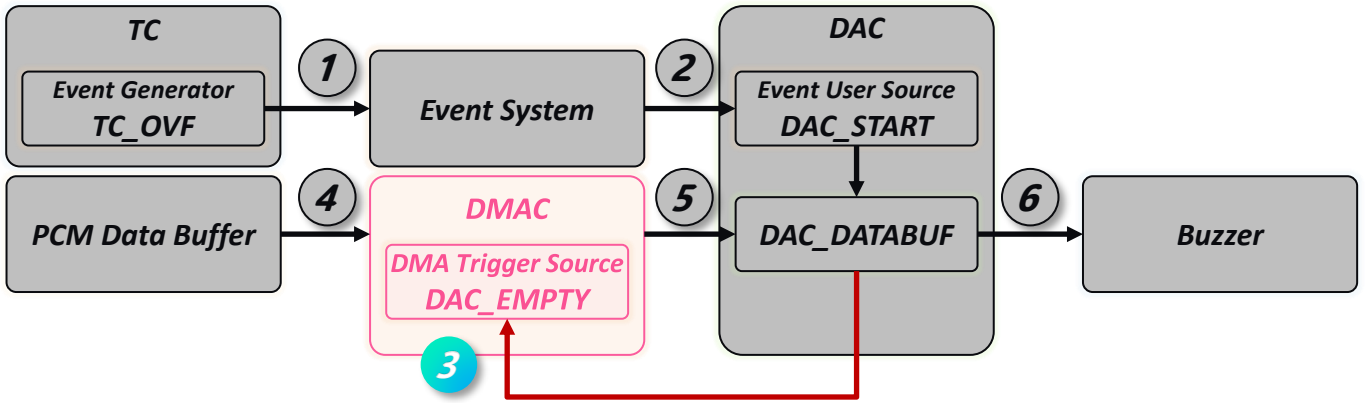
PA02/VOUT0

The MCC Configuration





The MCC Configuration



DMA Configuration

Use Linked List Mode

Active Channels List

Channel Number	Trigger
DMAC Channel 0	DAC_EMPTY0

DAC Empty is Trigger Source

Add Channel | Remove Selected Channel

DMA Channel 0 Settings

Enable Interrupt	☑
Trigger Action	One Beat Transfer per DMA Request
Source Address Mode	Increment Address After Every Transfer
Destination Address Mode	Fixed Address Mode
Beat Size	16-bit bus transfer
Burst Length	SINGLE
FIFO Threshold	1BEAT

Source Address Increment

Destination Address Fixed

16-bit Transfer

MICROCHIP

Application Implement

Application Implement

TSS_Voice.h

```
#ifndef TTS_VOICE_H
#define TTS_VOICE_H

// Welcome - Sample Length: 25431 samples, Sample Rate: 8000Hz
const uint16_t Welcome[25431] = {
0x043D, 0x043D, 0x043D, 0x043D, 0x043D, 0x043D, 0x043D, 0x043D, 0x043D, 0x043D, 0x043D, 0x043D, 0x043D, 0x043D,
0x0401, 0x0404, 0x040A, 0x0412, 0x041B, 0x0425, 0x042E, 0x0437, 0x043F, 0x0445, 0x0449, 0x044B, 0x044C, 0x044C,
0x044A, 0x0448, 0x0445, 0x0442, 0x043F, 0x043D, 0x043B, 0x043A, 0x043A, 0x043B, 0x043B, 0x043D, 0x043E, 0x043F,
0x0442, 0x0449, 0x0451, 0x045B, 0x0467, 0x0475, 0x0480, 0x0489, 0x0491, 0x0495, 0x0496, 0x0493, 0x048E, 0x0485,
0x0479, 0x046B, 0x045B, 0x044B, 0x043B, 0x042D, 0x041F, 0x0415, 0x040D, 0x0407, 0x0405, 0x0406, 0x0409, 0x040E,
0x0416, 0x041E, 0x0427, 0x042F, 0x0438, 0x043E, 0x0444, 0x0448, 0x044B, 0x044C, 0x044C, 0x044A, 0x0448, 0x0445,
0x0441, 0x043E, 0x043B, 0x0439, 0x0437, 0x0437, 0x0437, 0x0438, 0x0439, 0x043B, 0x043E, 0x0445, 0x044C, 0x0454,
0x045C, 0x0464, 0x046A, 0x0470, 0x0476, 0x0478, 0x0478, 0x0475, 0x0472, 0x046B, 0x0464, 0x045B, 0x0452, 0x0447,
0x043E, 0x0435, 0x042E, 0x0427, 0x0422, 0x041E, 0x041D, 0x041D, 0x041E, 0x0420, 0x0424, 0x0427, 0x042C, 0x042F,
0x0433, 0x0436, 0x0439, 0x043B, 0x043C, 0x043D, 0x043D, 0x043D, 0x043D, 0x043C, 0x043C, 0x043C, 0x043B, 0x043B,
0x043B, 0x043B, 0x043C, 0x043C, 0x043D, 0x043E, 0x043F, 0x0442, 0x0448, 0x044D, 0x0452, 0x0456, 0x045B, 0x045F,
0x0463, 0x0467, 0x0467, 0x0467, 0x0465, 0x0463, 0x0460, 0x045B, 0x0456, 0x044F, 0x0449, 0x0443, 0x043E, 0x0438,
0x0434, 0x042F, 0x042C, 0x0429, 0x0428, 0x0428, 0x0428, 0x0428, 0x042A, 0x042D, 0x0430, 0x0433, 0x0437, 0x043A,
0x043D, 0x0441, 0x0444, 0x0446, 0x0449, 0x044A, 0x044C, 0x044C, 0x044D, 0x044C, 0x044B, 0x0449, 0x0449, 0x0446,
0x0444, 0x0442, 0x0440, 0x043D, 0x043E, 0x0440, 0x0441, 0x0441, 0x0443, 0x0444, 0x0447, 0x044A, 0x044C, 0x044C,
0x044D, 0x044E, 0x044F, 0x0450, 0x044F, 0x044D, 0x044C, 0x044B, 0x044A, 0x0448, 0x0446, 0x0443, 0x0441, 0x043F,
0x043F, 0x043C, 0x043B, 0x0439, 0x0439, 0x0438, 0x0438, 0x0438, 0x0438, 0x0438, 0x0439, 0x0439, 0x043A, 0x043B,
…

};// End of Welcome

#define NUM_VOICE 1

typedef struct {
    uint32_t length;
    const uint16_t *data;
} TTS_VOICE;

TTS_VOICE TTS_Voice[NUM_VOICE] = {
    { 25431, Welcome }
};

#endif // TTS_VOICE_H
```

MICROCHIP

Application Implement

main.c

```
#include <stddef.h>                     // Defines NULL
#include <stdbool.h>                    // Defines true
#include <stdlib.h>                     // Defines EXIT_FAILURE
#include "definitions.h"                // SYS function prototypes
#include "TTS_Voice.h"

volatile uint8_t  PCM_Voice  = 0;
volatile uint32_t DMA_BlockSize = 0;
volatile uint32_t DMA_BlockNums = 0;
volatile uint32_t DMA_BlockIndx = 0;

void DMAC0_CompleteCallback(DMAC_TRANSFER_EVENT event, uintptr_t contextHandle)
{
    if( event==DMAC_TRANSFER_EVENT_COMPLETE )
    {
        if( DMA_BlockIndx < DMA_BlockNums )
        {
            uint16_t *srcAddr;
            size_t   blockSize;

            DMA_BlockIndx++;
            blockSize = (TTS_Voice[PCM_Voice].length*2)-(DMA_BlockIndx*DMA_BlockSize);
            if( blockSize>DMA_BlockSize ) blockSize = DMA_BlockSize;
            srcAddr = (uint16_t *)(((uint8_t *)TTS_Voice[PCM_Voice].data)+(DMA_BlockIndx*DMA_BlockSize));
            // Continue remain TSS Wave data
            DMAC_ChannelTransfer( DMAC_CHANNEL_0,                           // channel
                                  (const void *) srcAddr,                   // srcAddr
                                  (const void *) &(DAC_REGS->DAC_DATABUF),  // destAddr
                                  blockSize                                 // blockSize (n Bytes)
                                );
        }
        else
        {
            // Event System Channel 0 User Disable (DAC_VOUT0)
            EVSYS_UserDisable(61U);
        }
    }
}
```

Application Implement

main.c

```
int main ( void )
{
    SYS_Initialize ( NULL );

    // Register DMAC Complete callback to continue next block transfer
    DMAC_ChannelCallbackRegister( DMAC_CHANNEL_0, DMAC0_CompleteCallback, 0 );

    // Start TC timer
    TC0_CompareStart();

    // Given the DMA block size
#if 0
    DMA_BlockSize = (64*1024*2); // DMA support maximum 64K beats per block = 64K x 16bits(beat size) = 128K Bytes
#else
    // The plib_damc has a bug to convert blocksize to 16bit before divided,
    // We use maximum 64Kbyte(16bits) as maximum block size for workaround but not modify bug in plib_damc
    DMA_BlockSize = (64*1024)-1;
#endif

    // Calculate number of blocks (maximum block size)
    DMA_BlockIndx = 0;
    DMA_BlockNums = (TTS_Voice[PCM_Voice].length*2)/DMA_BlockSize;
    if( DMA_BlockNums==0 )
    {
        // Block size less than maximum block size
        DMA_BlockSize = (TTS_Voice[PCM_Voice].length*2);
    }

    // Start DMA Transfer
    DMAC_ChannelTransfer( DMAC_CHANNEL_0,                          // channel
                          (const void *) TTS_Voice[PCM_Voice].data, // srcAddr
                          (const void *) &(DAC_REGS->DAC_DATABUF),  // destAddr
                          DMA_BlockSize                             // blockSize (n Bytes)
                        );

    while ( true )
    {
```

The bug of DMAC_ChannelTransfer () in plib_dmac

plib_dmac.c

```
bool DMAC_ChannelTransfer( DMAC_CHANNEL channel, const void *srcAddr, const void *destAddr, size_t blockSize )
{
    uint8_t beat_size = 0U;
    bool returnStatus = false;
    bool isBusy = dmacChannelObj[channel].isBusy;
    const uint32_t* pu32srcAddr = (const uint32_t*)srcAddr;
    const uint32_t* pu32dstAddr = (const uint32_t*)destAddr;

    if (((DMAC_REGS->CHANNEL[channel].DMAC_CHINTFLAG & (DMAC_CHINTENCLR_TCMPL_Msk | DMAC_CHINTENCLR_TERR_Msk)) != 0U) || (!isBusy) )
    {
        /* Clear the transfer complete flag */
        DMAC_REGS->CHANNEL[channel].DMAC_CHINTFLAG = DMAC_CHINTENCLR_TCMPL_Msk | DMAC_CHINTENCLR_TERR_Msk;

        dmacChannelObj[channel].isBusy = true;

        ...

        /*Calculate the beat size and then set the BTCNT value */
        beat_size = (uint8_t)((dmacDescReg->DMAC_BTCTRL & DMAC_BTCTRL_BEATSIZE_Msk) >> DMAC_BTCTRL_BEATSIZE_Pos);

        /* Set Block Transfer Count */
        dmacDescReg->DMAC_BTCNT = ((uint16_t)blockSize / ((uint16_t)1U << beat_size));

        /* Enable the channel */
        DMAC_REGS->CHANNEL[channel].DMAC_CHCTRLA |= DMAC_CHCTRLA_ENABLE_Msk;

        /* Verify if Trigger source is Software Trigger */
        if ((((DMAC_REGS->CHANNEL[channel].DMAC_CHCTRLA & DMAC_CHCTRLA_TRIGSRC_Msk) >> DMAC_CHCTRLA_TRIGSRC_Pos) == 0x00U)
                                && (((DMAC_REGS->CHANNEL[channel].DMAC_CHEVCTRL & DMAC_CHEVCTRL_EVIE_Msk)) != DMAC_CHEVCTRL_EVIE_Msk))
        {
            /* Trigger the DMA transfer */
            DMAC_REGS->DMAC_SWTRIGCTRL |= ((uint32_t)1U << channel);
        }
        returnStatus = true;
    }

    return returnStatus;
}
```

MICROCHIP

Microchip Trademarks

https://www.microchip.com/en-us/about/legal-information/microchip-trademarks

Trademark	Description/Appropriate Nouns	Notes
ClockWorks®	configurator, device, family, product, series	US only
CodeGuard™	security	
CryptoAuthentication™	development library, device, family, IC, library	
CryptoAutomotive™	development kit, security IC, technology	
CryptoCompanion™	chip	
CryptoController™	solution, TPM, trusted platform module	
CryptoMemory®	cryptographic security IC, IC	
CryptoRF®	device, transponder	
dsPIC®	DSC, digital signal controller	
dsPIC®	dsPIC® logo	
dsPICDEM™	demonstration board, development board	
dsPICDEM.net™	board	
Dynamic Averaged Matching™ (DAM™)	architecture	
ECAN™	solution, technology	
Espresso T1S™	device	
EtherGREEN™	family, platform, technology	
EtherSynch®	family, platform, technology	US only
EyeOpen™	cable compensation	
Flashtec®	controller, family, product	US only
flexPWR®	technology	
Frequency on Demand®		Registered in other countries (not US)
GestIC®	controller, sensing, technology	
GridTime™	device, GNSS time server, time server	
HELDO®	family, regulator	
Hyper Speed Control®	architecture, family	US only
HyperLight Load®	mode	US only
In-Circuit Serial Programming™ (ICSP™)	programming capability	
IdealBridge™	dual MOSFET-bridge rectifier, rectifier	
IGaT™	board, demonstration board	
IGLOO®	device, family, FPGA, series	
INICnet™	sniffer, technology	

Note: *Always check with the Legal Department for the most current trademarks.

Trademark	Description/Appropriate Nouns	Notes
MICROCHIP	Microchip name and logo	This is the vertical representation; the horizontal representation is also available.
Microsemi	Microsemi logo	Do not use
Microsemi®		Do not use
Mindi™	analog simulator, analog simulator tool	
MiWi™	application, coordinator, mesh, network, protocol, software, stack, wireless networking protocol	
MOST®	board, cooperation, design, device, evaluation kit, NetServices, network, platform, product, specification, system, technology	
MOST®	MOST® logo	
motorBench®	development suite	US only
MPASM™	assembler	
MPI™	device, product	
MPLAB®	**Nouns:** C compiler, compiler, development ecosystem, ICD, ICE, IDE, IPE, in-circuit debugger, in-circuit emulator, integrated development environment, integrated programming environment, library, programmer, server, starter kit, universal device programmer, XC compiler **Tool Names:** AI Edge, Analog Designer, Analysis Tool Suite, Code Configurator, Connect Configurator, Data Visualizer, Discover, Extensions, Harmony, Mindi™, Network Creator, PICkit™, Programmer-to-Go, Programmer-to-Go Mobile App, PTG, SiC Power Simulator, Snap, Xpress	
MPLAB® CERTIFIED	MPLAB® Certified logo	
MPLIB™	librarian, object librarian	

Note: *Always check with the Legal Department for the most current trademarks.

Trademark	Description/Appropriate Nouns	Notes
Ripple Blocker™	attenuator, family of linear regulators, product line, technology	
REAL ICE™	in-circuit emulator	
RTAX™	FPGA	
RTG4™	FPGA	
SAM-BA®	applet, application, host, in-system programmer, ISP, monitor, software	
SAM-ICE™	emulator	
SenGenuity®	product, sensor	
Serial Quad I/O™ (SQI™)	family, Flash device, product	
Silicon Storage Technology®		Japan only
simpleMAP™	protocol	
SimpliPHY™	device, family, product	
SmartBuffer™	device, IC	
SmartFusion®	evaluation kit, FPGA, SoC, System-on-Chip	US only
SmartHLS™	compiler, compiler software, IDE, software	
SMART-I.S.™	technology	
SpyNIC®	device	
SST®	Acronym for Silicon Storage Technology	
SST	SST logo	
STK®	starter kit	
storClad™	encryption technology, technology	
SuperFlash®	device, kit, macro, memory, technology	
SuperSwitcher™	buck regulator, family, regulator	
SuperSwitcher II™	family	
Switchtec™	family, product line, switch, technology	
Symmcom®		China only
Symmetricom®		
SynchroPHY™	device, family, product	
SyncServer®	clock, device, instrument, oscillator, server	

Note: *Always check with the Legal Department for the most current trademarks.

Trademark	Description/Appropriate Nouns	Notes
Adaptec®	Adaptec "a" logo	Registered in other countries (not US)
Adaptec®	adapter, cable, controller, driver, family, product, series, software, solution	
Adjacent Key Suppression™	technology	
AgileSwitch®	device, digital programmable gate driver, driver, family, gate driver	US only
AKS™	technology	
Analog-for-the-Digital Age™	AIPD family tagline	
Any Capacitor™	stable	
Anyin™	capability, feature, structure	
AnyOut™	capability, feature	
Augmented Switching™	technology	
AVR®	architecture, CPU, device, family, MCU, microcontroller	
AVR®	AVR® logo	
AVR Freaks®	forum	
BesTime®	technology	
BitCloud®	SDK, software development kit	
BlueSky®	firewall, subscription service, technology	
BodyCom™	development kit, system, technology	
chipKIT®	I/O shield, board, hardware, network, platform, software	
chipKIT®	chipKIT® logo	
Clockstudio™	application, program, software, utility	

Note: *Always check with the Legal Department for the most current trademarks.

Trademark	Description/Appropriate Nouns	Notes
InitDrive™	driver board, integrated solution, power module	
Intelligent Paralleling™	technology	
IntelliMOS®	family, power stage family	US only
Inter-Chip Connectivity™	technology	
JitterBlocker™	attenuator, family	
JukeBlox®	platform, technology	
jukeblox®	JukeBlox® registered word mark with a design	Image is not registered; this is a graphic treatment with the registered word mark
KeeLoq®	protocol, security IC, technology	
Kleer®	technology	
KLEER®	Kleer® registered word mark with a design	Image is not registered; this is a graphic treatment with the word mark
Knob-on-Display™ (KoD™)	device, family, HMI, knob designer, position wizard, solution, technology, touch controller, user interface, widget configurator	
LANCheck®	online design review, online review, review	
Libero®	design file, designer, IDE, SoC design suite, software, tool	US only
LinkMD®	cable diagnostics, diagnostics engine, family, technology	
MarginLink™	signal integrity testing	
maxCrypto™	controller-based encryption, encryption	
maXStylus®	solution	
maXTouch®	family, studio, technology, touchscreen controller	
maxView™	driver, storage manager, tool	
MediaLB®	bus, controller, device	
megaAVR®	device, MCU, microcontroller	
memBrain™	neuromorphic memory product, product, solution, technology	
	Microchip logo	Please do not use this logo; always use the Microchip name and logo.

Note: *Always check with the Legal Department for the most current trademarks.

Trademark	Description/Appropriate Nouns	Notes
MPLINK™	linker, object linker	
mSiC™	bare die, diode, gate driver, module, MOSFET, product, solution, technology	
mTouch®	development kit, evaluation kit, library, sensing solution, solution, technology	US only
MultiTRAK™	technology	
NetDetach™	technology	
Omniscient Code Generation™ (OCG)	compilation technology	
OptoLyzer®	network analysis tool, studio, tool	
PIC®	assembler, device, MCU, microcontroller	
PIC32	PIC32 logo	
PIC64	PIC64 logo	
PICDEM™	demonstration board	
PICDEM.net™	Internet/Ethernet demonstration board	
PICkit™	debugger, development tool, in-circuit debugger, on-board, programmer, starter kit, tool	
picoPower®	technology	
PICSTART®	development programmer	
PICtail™	board, daughter board	
PolarFire®	device, family, FPGA, kit, SoC, System-on-Chip	
PowerSmart™	designer, development suite, digital control library	
Precision Edge®	family, product family	US only
ProASIC®	architecture, board, family, FPGA, kit	US only
ProASIC Plus®	architecture, device, family, FPGA, switch	US only
Prochip Designer®	software suite	
PureSilicon™	oscillator, technology	
QMatrix™	device, sensor	
QTouch®	board, composer, configurator, device, development platform, library, project builder, sensor, solution, technology, tool	
Quiet-Wire®	enhanced EMI technology, family, technology	

Note: *Always check with the Legal Department for the most current trademarks.

Trademark	Description/Appropriate Nouns	Notes
SyncWorld®	ecosystem program, timing	US only
Tachyon®	controller, family, platform, product, technology	
The Embedded Control Solutions Company®		US only; should be in italics
TimeCesium®	primary frequency reference, product	US only
TimeHub®	platform, system	US only
TimePictra®	platform, software suite, synchronization management system	US only
TimeProvider®	clock, enhanced PRTC, ePRTC, extension, system, grandmaster, kit, module, product, series, server, shelf	US only
TimeSource®	PRC, PRS, primary reference clock, primary reference source, shelf, system	
tinyAVR®	MCU, microcontroller	
Total Endurance™	software model	
TSHARC™	controller, touchscreen controller	
Turing™	hardware security module, programmer	
UNI/O®	bus, communication protocol, firmware, hardware port, memory chip	
USBCheck™	design review	
VariSense™	technology	
VectorBlox™	accelerator, intellectual property, IP, platform, SDK, software development kit, solution, tool, tool chain, tool flow	
Vectron®	oscillators, products	
VelocityDRIVE™	CT, configuration tool, product, solution, software platform, SP	
VeriPHY™	cable diagnostics algorithm, cable diagnostics software suite	
ViewSpan™	technology	
WiperLock™	technology	
XMEGA®	Custom Logic (XCL) device, family, MCU, microcontroller, series	
XpressConnect™	retimer	
ZENA™	analyzer, software, wireless adapter	
ZL®		US only

Note: *Always check with the Legal Department for the most current trademarks.

MICROCHIP