MPLAB® Harmony 3 之基础篇（09）

-- 如何使用 Harmony I2C 驱动开发应用程序

Microchip Technology Inc.
MCU32 产品部

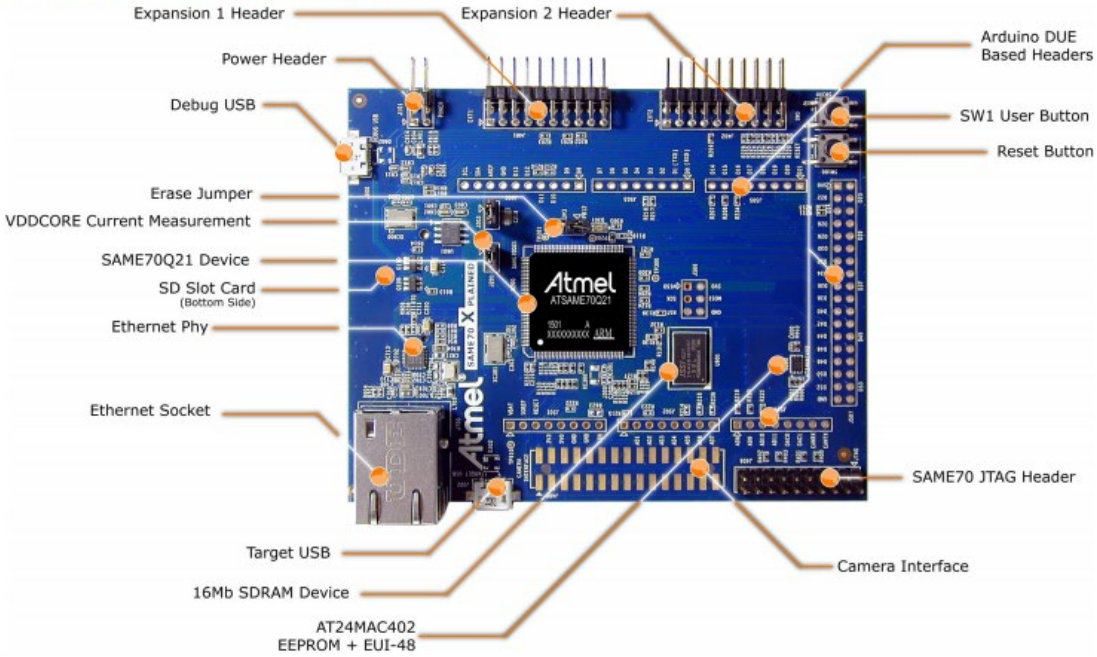
一、 简介

MPLAB® Harmony 是 MPLAB 开发工具生态系统的重要组成部分，MPLAB® Harmony 3 适用于是 Harmony 系列开发工具的换代升级，增加了对 SAM®系列微处理器的支持，是 Microchip®32 位 SAM®和 PIC®微控制器的嵌入式系统的重要软件方案。本文主要介绍如何利用 MPLAB X IDE 创建一个工程，利用 MPLAB Harmony 3 Configurator(MHC)添加 IC 外设驱动到工程文件，并利用 MHC 的配置工具(CLOCK, PIN 等)完成 IC 外设的配置。通过调用 I2C PLIB API 实现对 AT24 系列 EEPROM 的读写操作。

二、 硬件工具和软件平台

硬件：SAM E70 Xplained Board
http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-44050-Cortex-M7-Microcontroller-SAM-E70-XPLD-Xplained_User-guide.pdf

Figure 1-2 SAME70-XPLD Board Overview



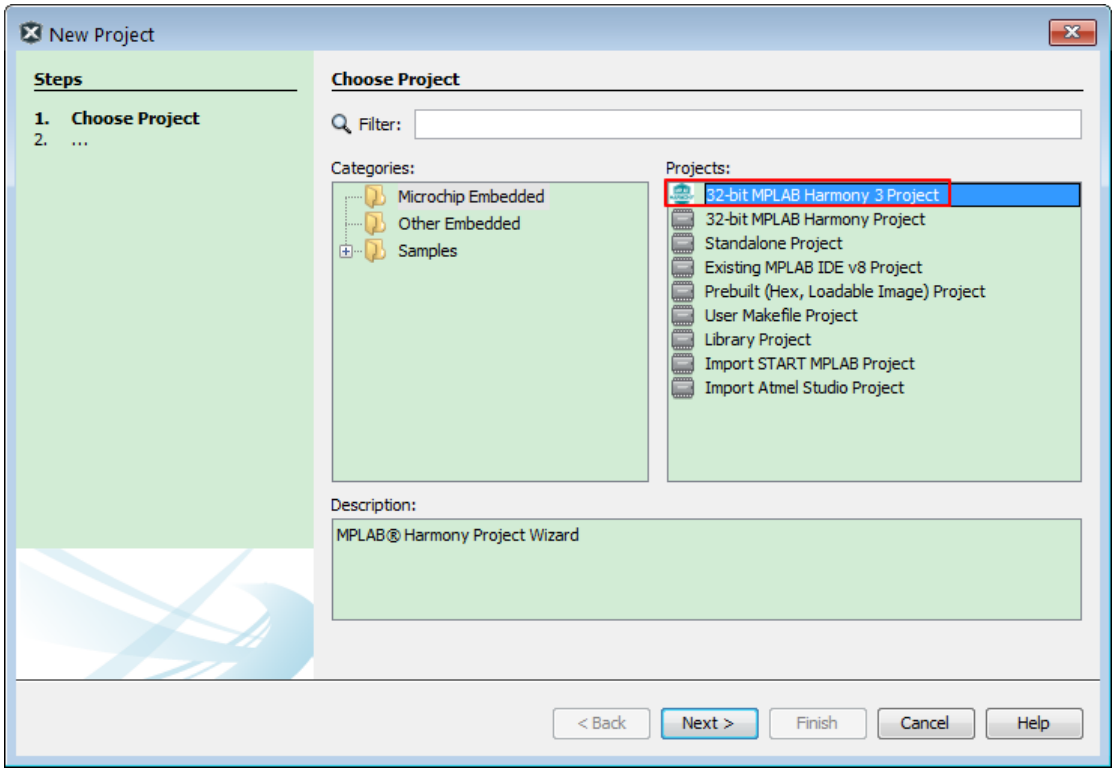
软件(开发工具和环境的安装和使用，见
“MPLAB® Harmony 3 之基础篇（01） -- Harmony 3 开发环境搭建”
“MPLAB® Harmony 3 之基础篇（02） -- 了解 MHC”

MPLAB® X IDE: v5.10 或者更新
XC32： v2.10 或者更新
Harmony 3： v3.10 或者更新

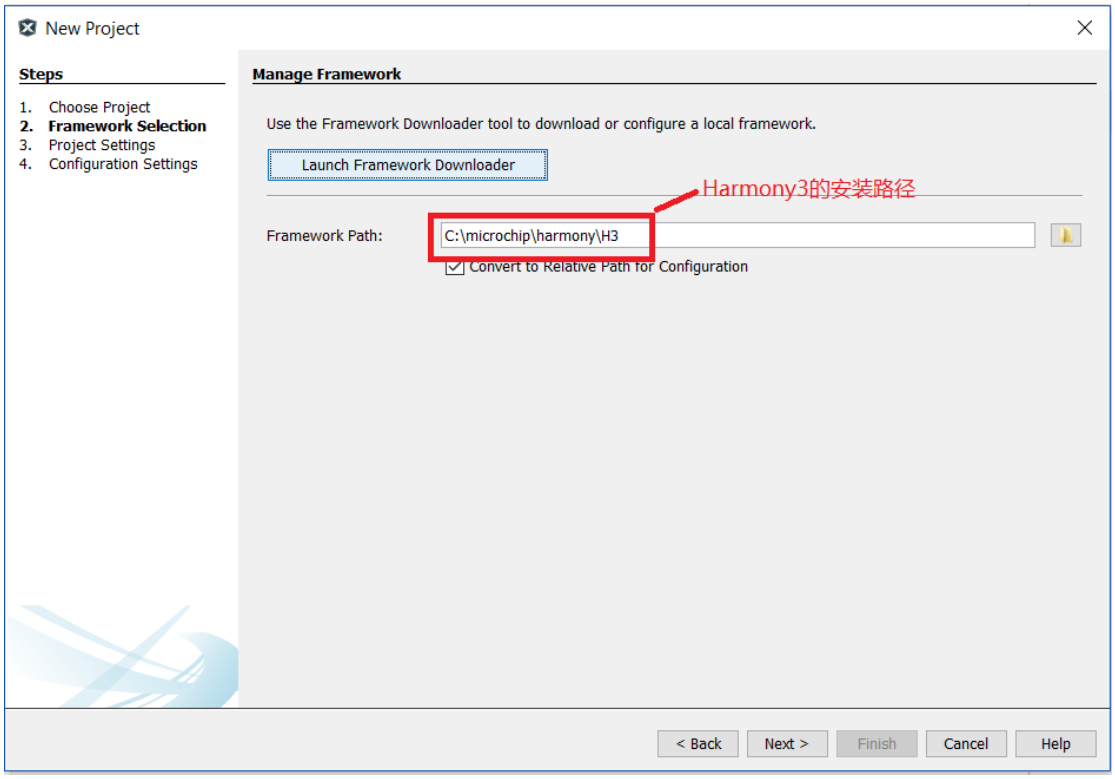
三、 详细步骤

接下来我们就可以用 MPLAB X IDE 和 MHC 一步步地创建和配置 I2C 外设驱动的程序。
注：以下 MHC 配置里没有特别标注出来的地方，说明使用的是默认选项。

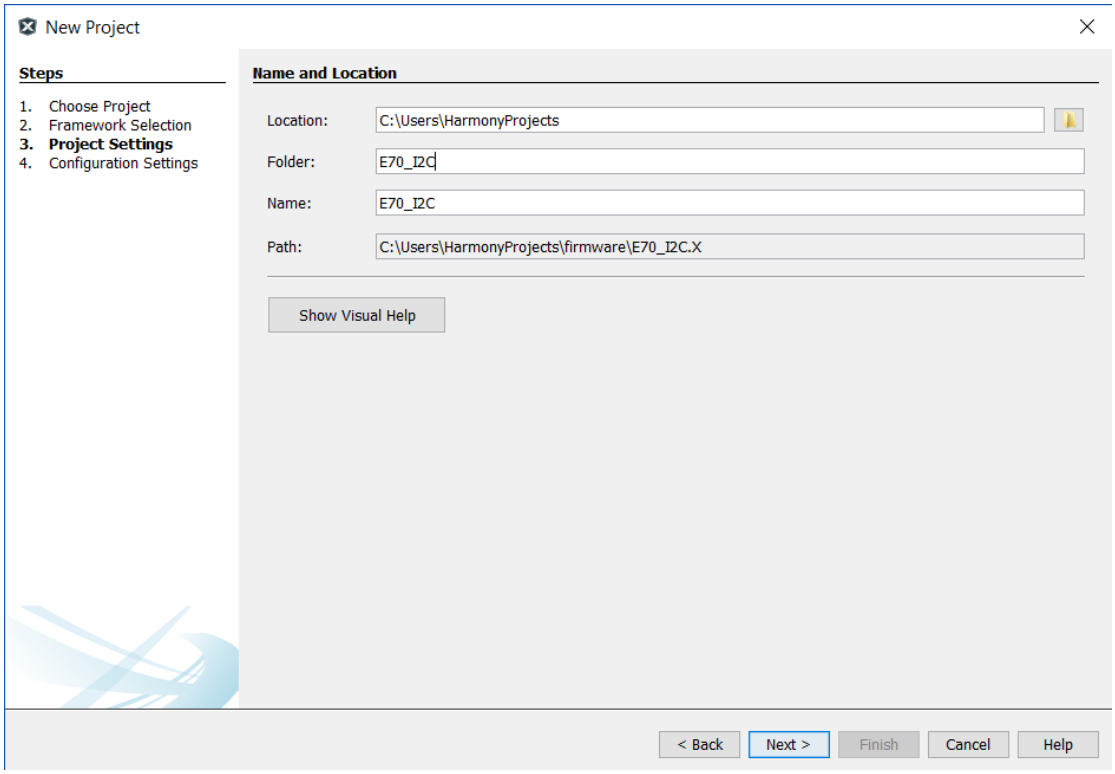
(一) 在 MPLAB X 里新建一个 Harmony 3 项目
在 MPLAB X IDE 里点击 File > New Project：



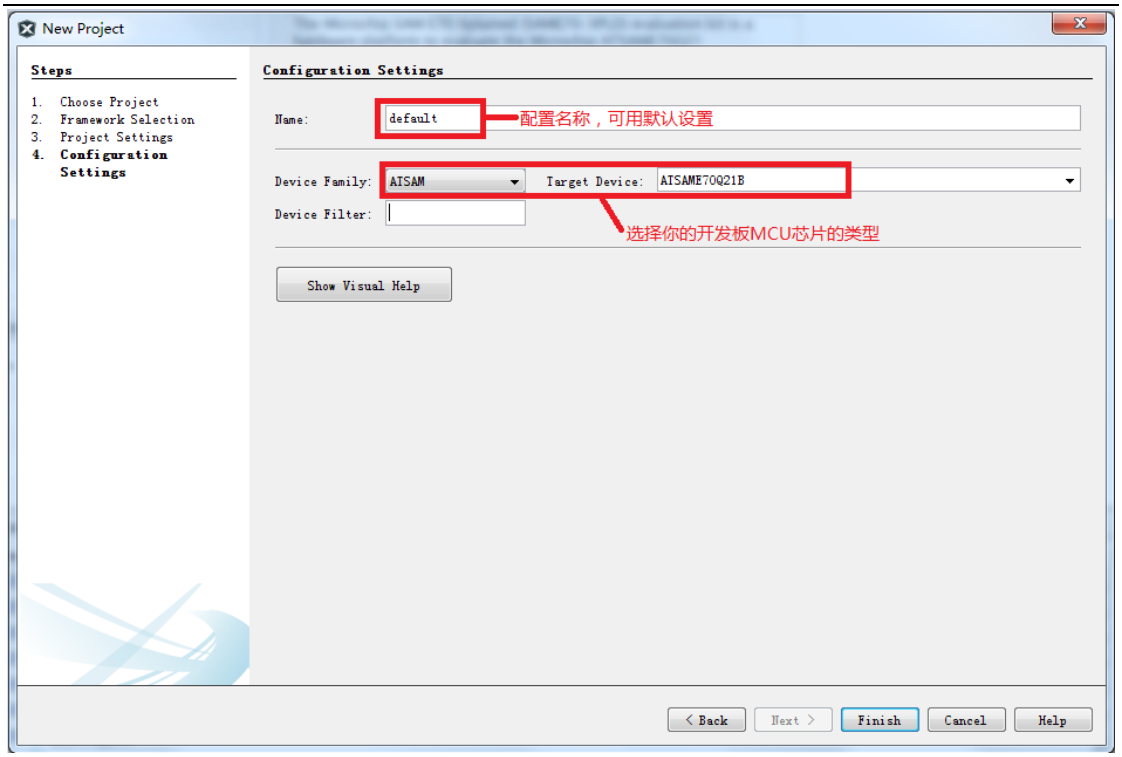
选择“32-bit MPLAB Harmony Porject”，然后点击“Next”按钮。



选择“Harmony Framework”路径，然后点击“Next”按钮。



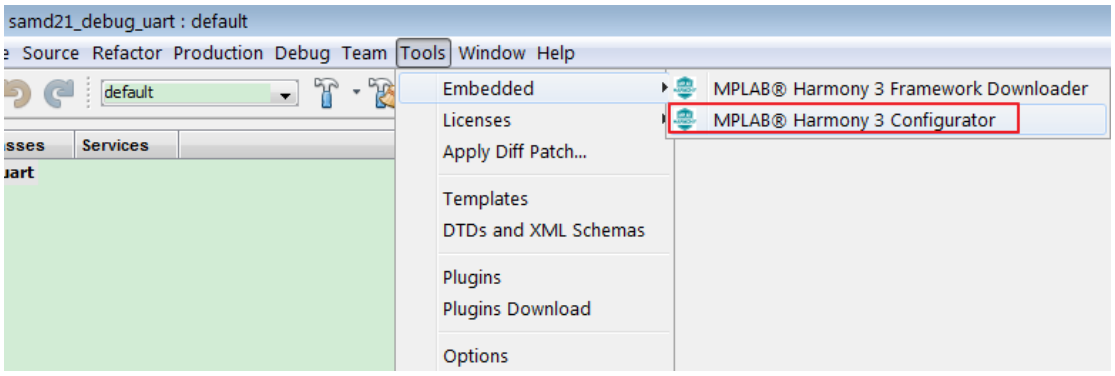
填写项目名称，本示例使用“E70_I2C”，然后点击“Next”按钮。



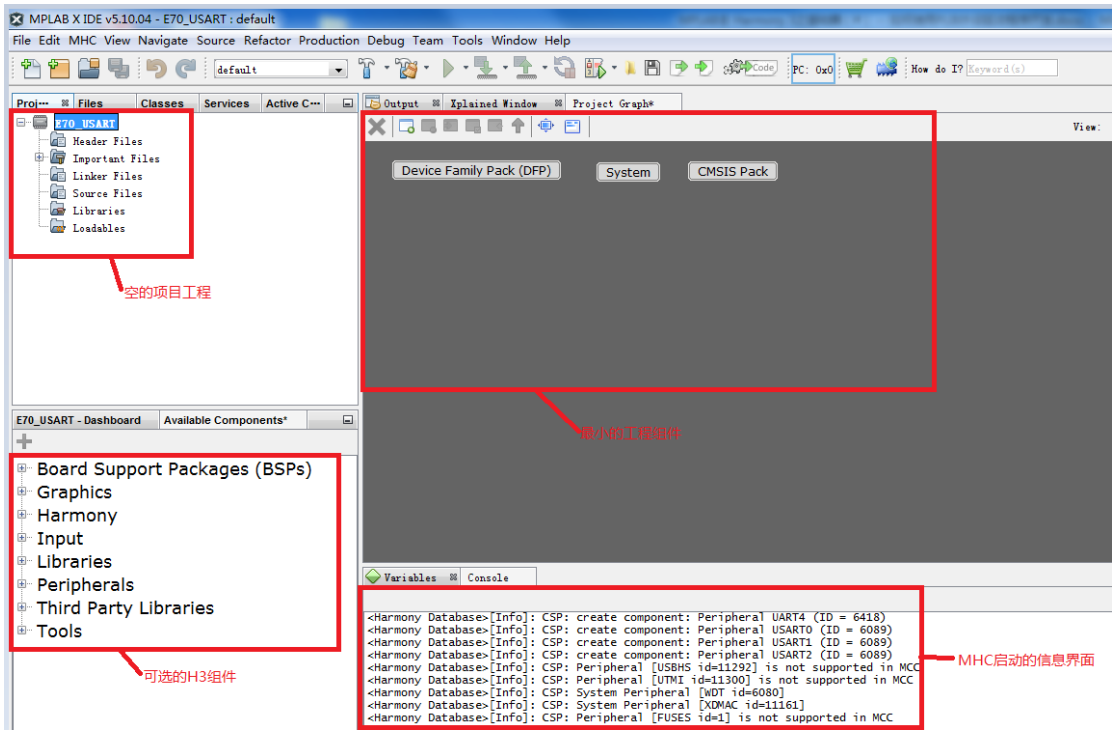
选择芯片类型“ATSAME70Q21B”，最后点击“Finish”按钮启动 MHC 配置界面。

(二) 启动 MHC

第一次创建项目时，MHC 配置界面会自动启动。或者手动在 MPLAB X 里点击 Tools > Embeded > MPLAB Harmony 3 Configurator 启动 MHC：

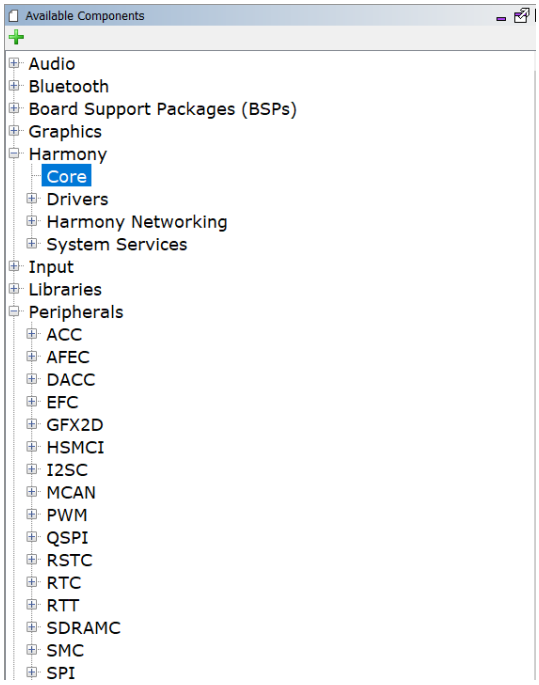


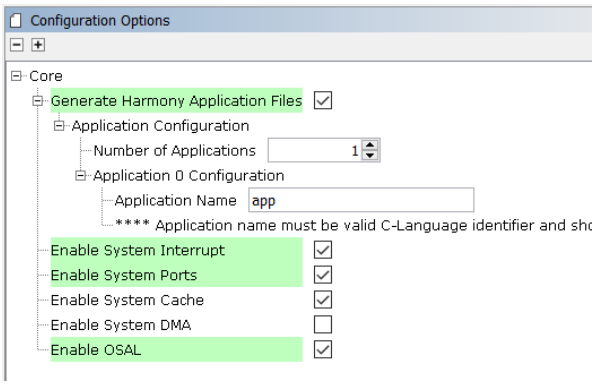
启动完成后的，主界面如下图：



(三)添加 Harmony Core，I2C0 外设驱动，AT24 EEPROM 驱动和 E70 Xplained Board (BSP)支持

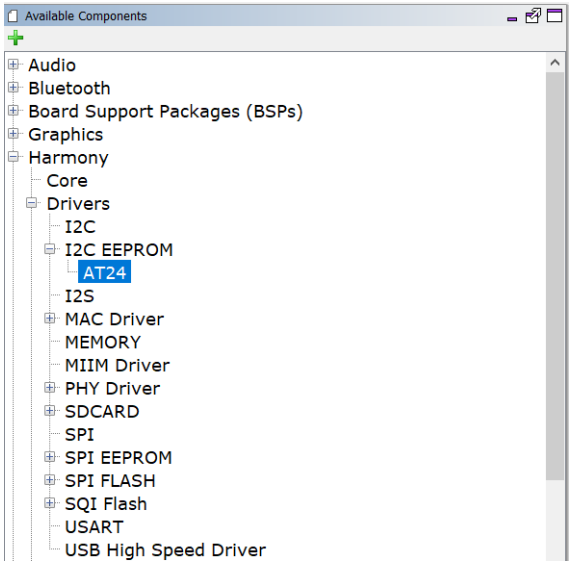
从 Harmony 中选择 Core，并选择合理的 Core 配置。



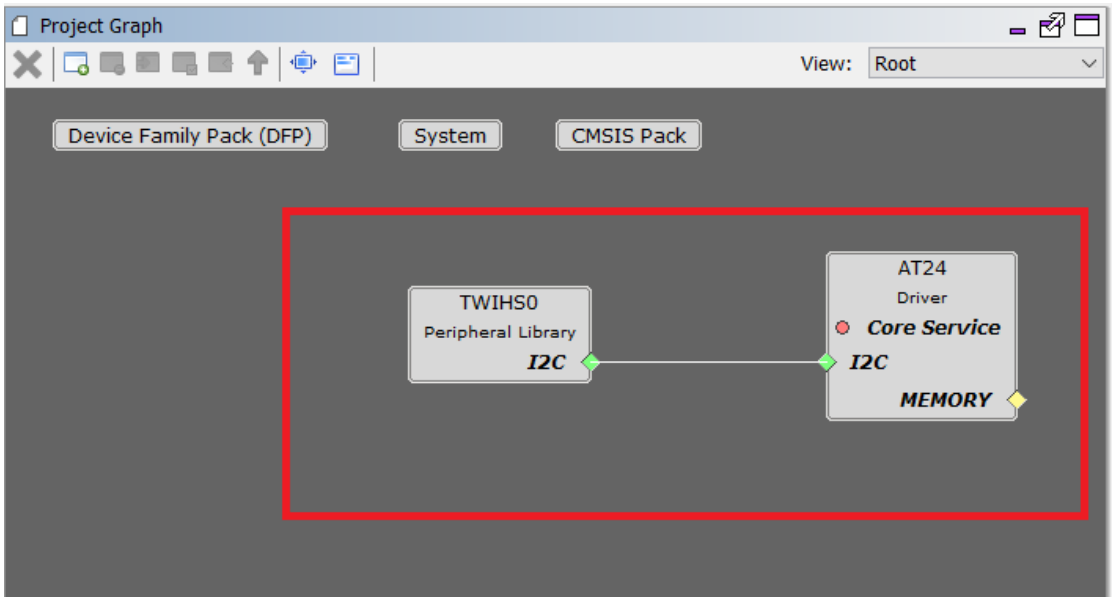


从 Peripherals 选择 TWIHS0 外设驱动，双击，添加到 Project Graph，USART 采用默认的设置。

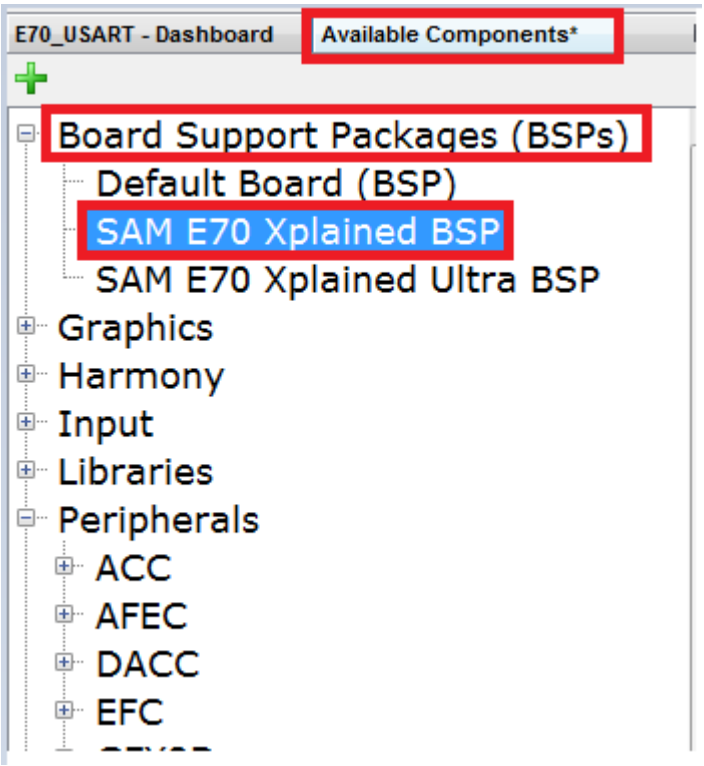
从 Harmony 中选择 AT24 的驱动程序。

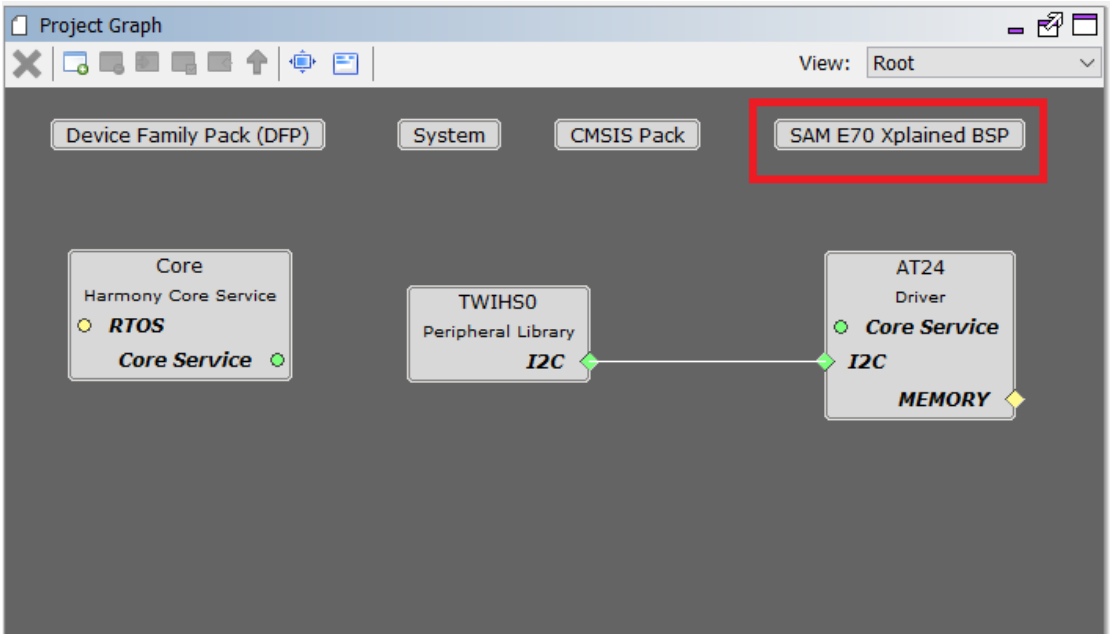


选好 I2C 和 AT24 的驱动以后，在 Project Graph 中用直线连接连接两者的 I2C 接口。



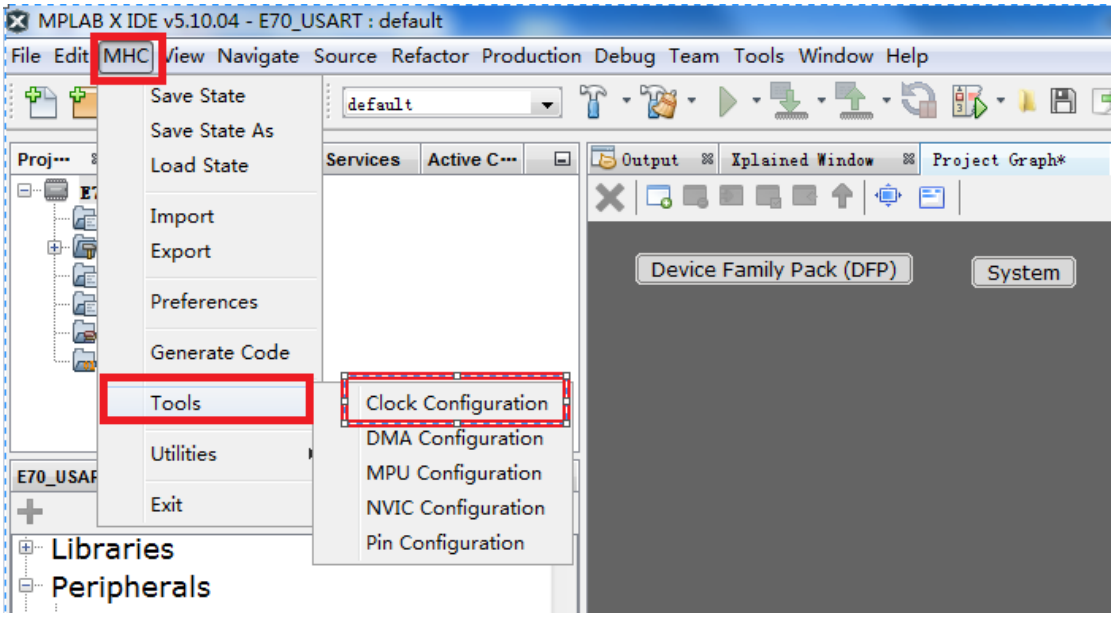
从 Board Support Packages 选择 E70 Xplained BSP 的支持，双击添加到 Project Graph.



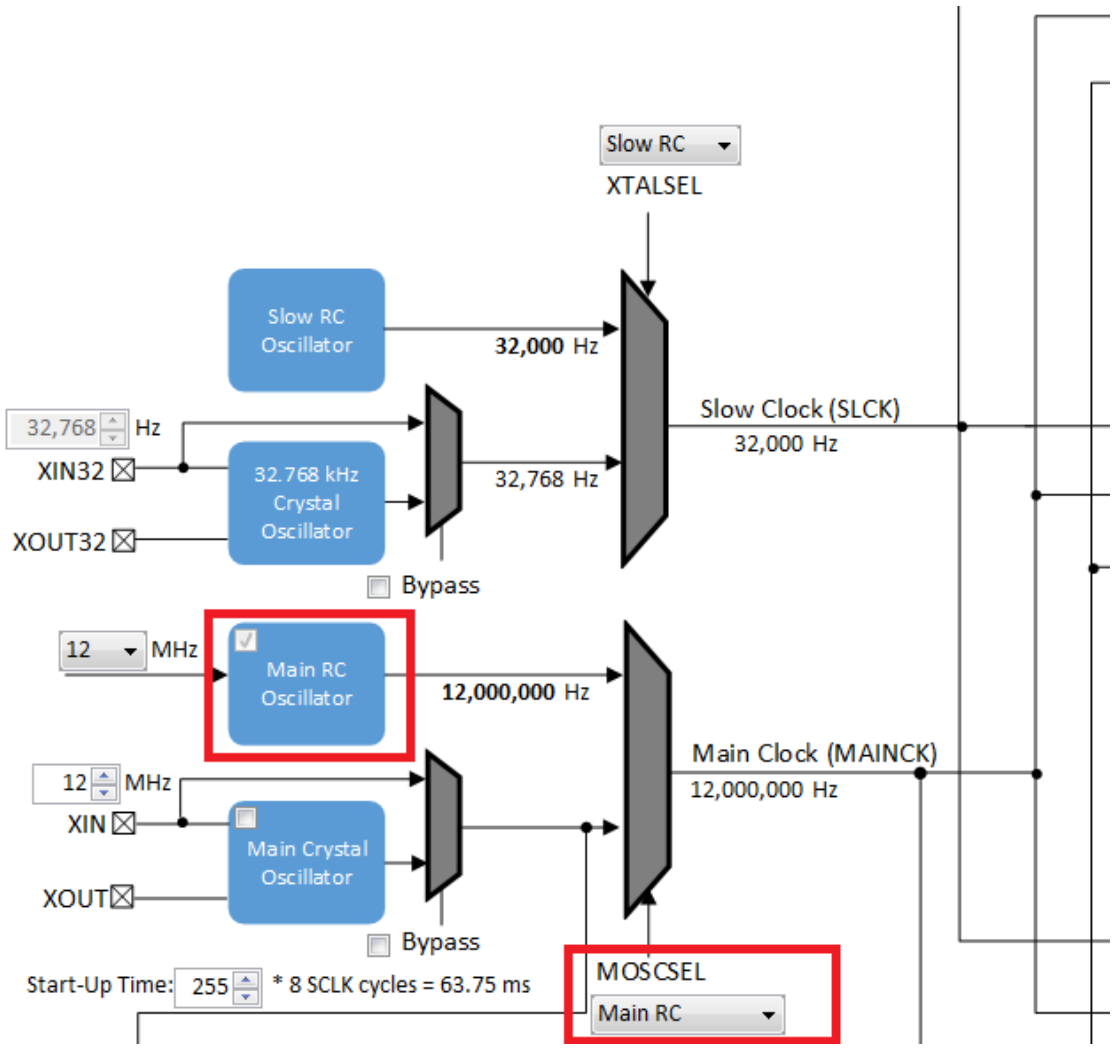


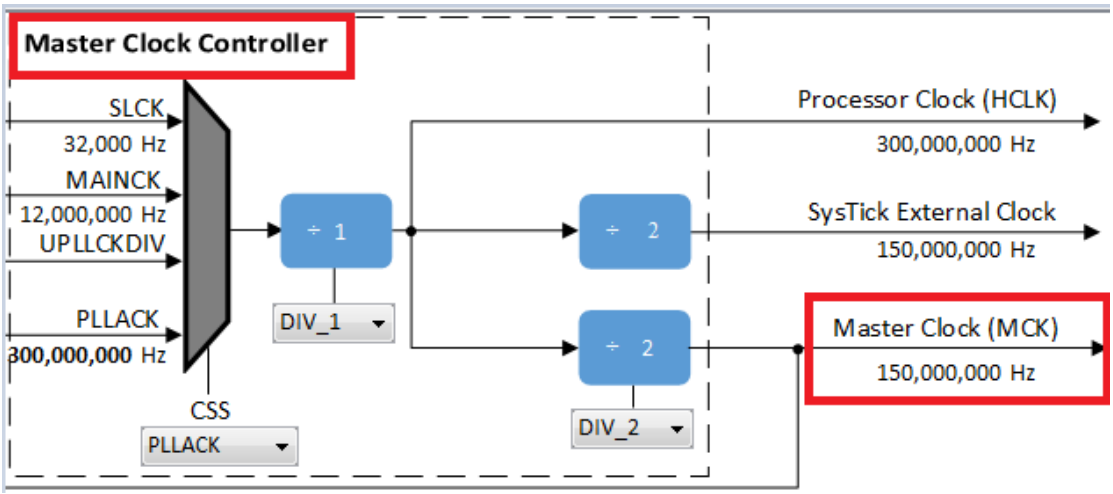
(四) 选择 MHC>Tools>Clock Configuration 菜单，启动时钟配置

E70 的时钟配置界面，确认时钟源和 I2C 的时钟是我们需要的配置。

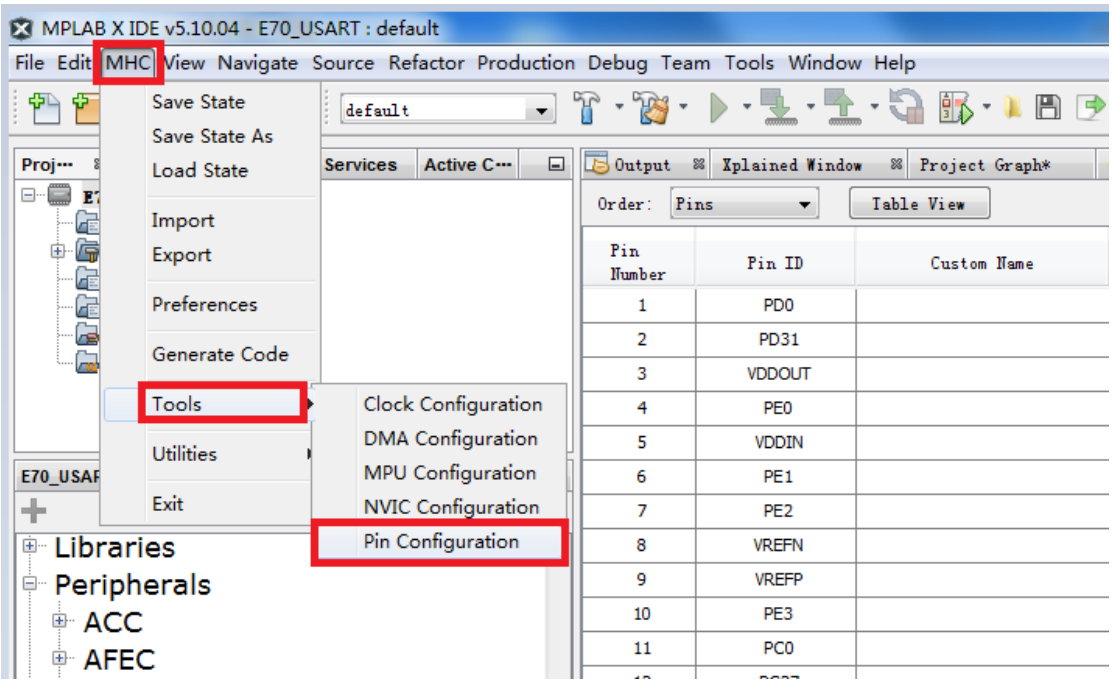


这里我们使用默认配置即可：





(五) 使用 MHC>Tools>Pin Configuration，配置 I2C 和 LED0 的引脚分配



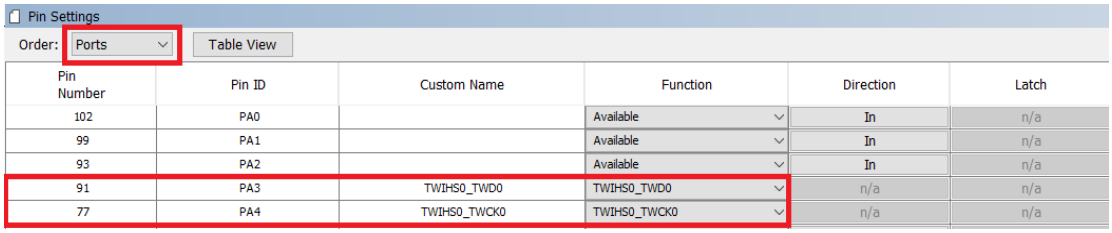
I2C 的管脚分配可以参考以下 E70 Xplained Board User guide 文档 (Atmel-44050-Cortex-M7-Microcontroller-SAM-E70-XPLD-Xplained_User-guide.pdf)

11 [I2C_SDA]	PA3	TWD0	Camera Connector, EXT2 Header, J500 Header (Arduino Shield), AT24MAC402, Embedded Debugger
12 [I2C_SCL]	PA4	TWCK0	Camera Connector, EXT2 Header, J500 Header (Arduino Shield), AT24MAC402, Embedded Debugger

Table 4-25 Virtual COM Port Connections

SAM E70 Pin	Function	Shared Functionality
PB4	TXD1 (SAM E70 UART TX Line)	EXT2 Header, J507 Header, Embedded Debugger
PA21	RXD1 (SAM E70 UART RX Line)	EXT2 Header, J507 Header, Embedded Debugger

E70 XPlained board 上分配 PA3 和 PA4 到 TWD0 和 TWCK0。

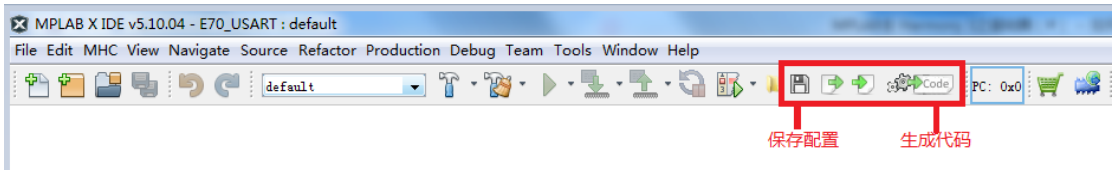


E70 XPlained board 上分配 PA23 到 LED0。

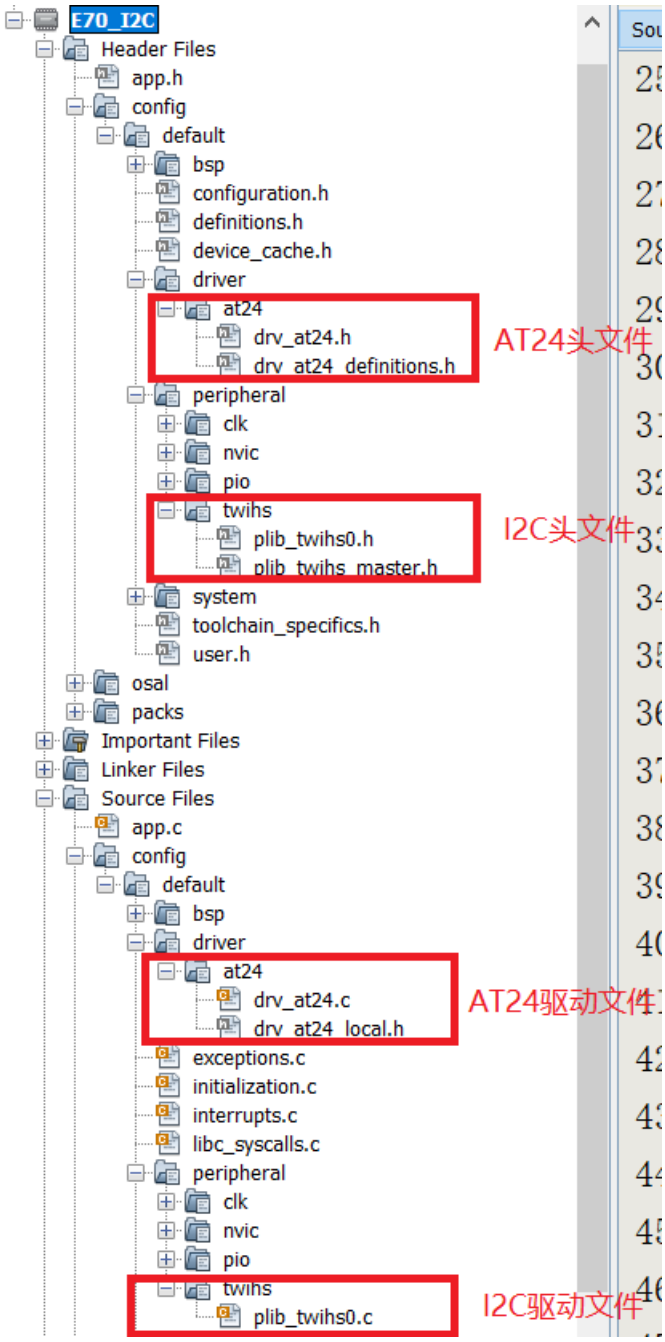
46	PA23	LED0	LED_AL	Out	High

(六)“保存”项目配置和“代码生成”

在任务栏点击“保存”和“代码生成”按钮

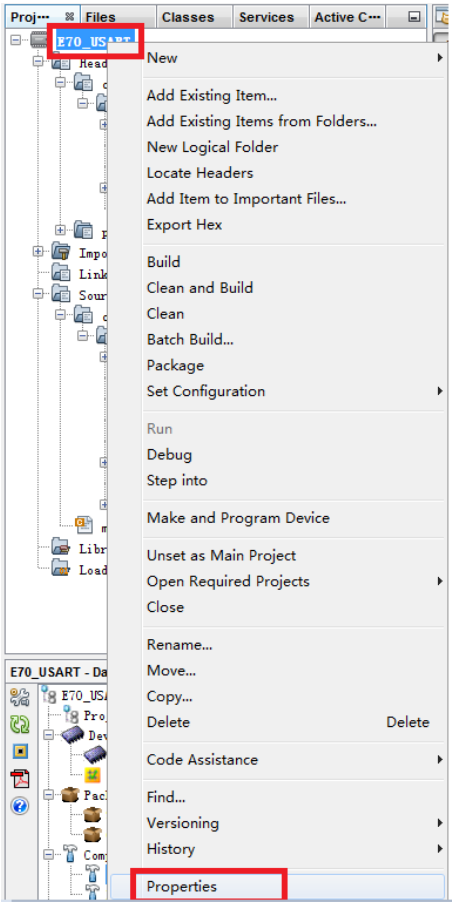


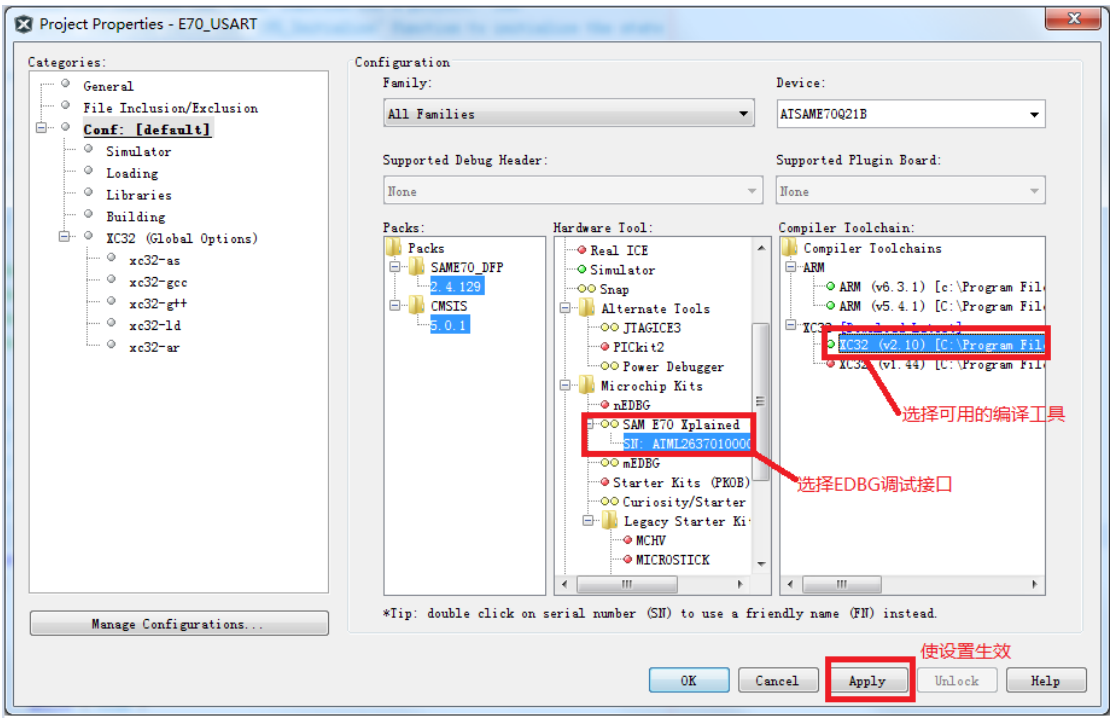
展开左侧的工程项目管理目录树，可以看到相关的 I2C 头文件和源代码已经生成了。



(七) 设置项目的调试接口和编译器

选择项目 E70_I2C，右键选择 Properties





(八) app.h 里增加如下测试代码

```
// ******************************************************************************
```

```
// *****************************************************************************
// Section: Type Definitions
// *****************************************************************************
// *****************************************************************************
#define BUFFER_SIZE                 256

// *****************************************************************************
/* Application states

  Summary:
    Application states enumeration

  Description:
    This enumeration defines the valid application states.   These states
    determine the behavior of the application at various times.
*/

typedef enum
{
    /* Application's state machine's initial state. */
    APP_STATE_INIT=0,
    APP_STATE_WRITE,
    APP_STATE_WAIT_WRITE_COMPLETE,
    APP_STATE_READ,
    APP_STATE_WAIT_READ_COMPLETE,
    APP_STATE_VERIFY,
    APP_STATE_ERROR,
    APP_STATE_IDLE,
    /* TODO: Define states used by the application state machine. */

} APP_STATES;


// *****************************************************************************
/* Application Data

  Summary:
    Holds application data

  Description:
    This structure holds the application's data.
```

```
Remarks:
    Application strings and buffers are be defined outside this structure.
 */

typedef struct
{
    /* The application's current state */
    APP_STATES state;

    DRV_HANDLE drvHandle;

    uint8_t writeBuffer[BUFFER_SIZE];

    uint8_t readBuffer[BUFFER_SIZE];

    volatile bool isTransferDone;

    /* TODO: Define any additional data used by the application. */

} APP_DATA;
```

(九) app.c 里增加如下测试代码

```
// *****************************************************************************
// *****************************************************************************
// Section: Global Data Definitions
// *****************************************************************************
// *****************************************************************************
#define AT24_EEPROM_MEM_ADDR                    0x00
// *****************************************************************************
/* Application Data

  Summary:
    Holds application data

  Description:
    This structure holds the application's data.

  Remarks:
    This structure should be initialized by the APP_Initialize function.

    Application strings and buffers are be defined outside this structure.
*/

APP_DATA appData;

// *****************************************************************************
// *****************************************************************************
// Section: Application Callback Functions
// *****************************************************************************
// *****************************************************************************

/* TODO:  Add any necessary callback functions.
*/
void APP_EEPROM_EventHandler(DRV_AT24_TRANSFER_STATUS event, uintptr_t context)
{
    switch(event)
    {
        case DRV_AT24_TRANSFER_STATUS_COMPLETED:
            appData.isTransferDone = true;
            break;
        case DRV_AT24_TRANSFER_STATUS_ERROR:
        default:
```

```
                appData.isTransferDone = false;
                appData.state = APP_STATE_ERROR;
                break;
        }
}
// *****************************************************************************
// *****************************************************************************
// Section: Application Local Functions
// *****************************************************************************
// *****************************************************************************


/* TODO:  Add any necessary local functions.
*/


// *****************************************************************************
// *****************************************************************************
// Section: Application Initialization and State Machine Functions
// *****************************************************************************
// *****************************************************************************

/*******************************************************************************
  Function:
    void APP_Initialize ( void )

  Remarks:
    See prototype in app.h.
 */

void APP_Initialize ( void )
{
    /* Place the App state machine in its initial state. */
    appData.state = APP_STATE_INIT;

    appData.isTransferDone = false;

}

/*******************************************************************************
  Function:
    void APP_Tasks ( void )
```

```
  Remarks:
    See prototype in app.h.
 */

void APP_Tasks ( void )
{
    /* Check the application's current state. */
    switch ( appData.state )
    {
        /* Application's initial state. */
        case APP_STATE_INIT:

            appData.drvHandle = DRV_AT24_Open(DRV_AT24_INDEX,
DRV_IO_INTENT_READWRITE);

            if (appData.drvHandle != DRV_HANDLE_INVALID)
            {
                DRV_AT24_EventHandlerSet(appData.drvHandle, APP_EEPROM_EventHandler, 0);

                appData.state = APP_STATE_WRITE;
            }
            else
            {
                appData.state = APP_STATE_ERROR;
            }
            break;

        case APP_STATE_WRITE:

            /* Fill up the write buffer */
            for (uint32_t i = 0; i < BUFFER_SIZE; i++)
            {
                appData.writeBuffer[i] = i;
            }

            appData.state = APP_STATE_WAIT_WRITE_COMPLETE;

            if (DRV_AT24_Write(appData.drvHandle,
                    appData.writeBuffer,
                    BUFFER_SIZE,
                    AT24_EEPROM_MEM_ADDR) == false)
            {
                appData.state = APP_STATE_ERROR;
```

```
        }
        break;

    case APP_STATE_WAIT_WRITE_COMPLETE:

        if (appData.isTransferDone == true)
        {
            appData.isTransferDone = false;
            appData.state = APP_STATE_READ;
        }
        break;

    case APP_STATE_READ:

        appData.state = APP_STATE_WAIT_READ_COMPLETE;

        if (DRV_AT24_Read(appData.drvHandle,
                appData.readBuffer,
                BUFFER_SIZE,
                AT24_EEPROM_MEM_ADDR) == false)
        {
            appData.state = APP_STATE_ERROR;
        }
        break;

    case APP_STATE_WAIT_READ_COMPLETE:

        if (appData.isTransferDone == true)
        {
            appData.isTransferDone = false;
            appData.state = APP_STATE_VERIFY;
        }
        break;

    case APP_STATE_VERIFY:

        if (memcmp(appData.writeBuffer, appData.readBuffer, BUFFER_SIZE ) == 0)
        {
            appData.state = APP_STATE_IDLE;
        }
        else
        {
            appData.state = APP_STATE_ERROR;
```

```
            }
            break;

        case APP_STATE_IDLE:
            /* Turn on the LED to indicate success */
            LED_On();
            break;

        case APP_STATE_ERROR:
        default:
            LED_Off();
            break;
    }
}
```

(十)编译下载测试

用 Micro USB 线通过 EDBG 调试口将 SAM E70 Xplained Board 开发板连接到电脑编译并下载程序：

如果程序正常运行，你将看到 LED0 指示灯被点亮。

四、 总结

本文展示了如何通过 MPLAB X IDE 和 MHC 一步步完成了一个通过 I2C 的程序，开发人员可以从这个过程了解到 Harmony 配置开发的全过程。