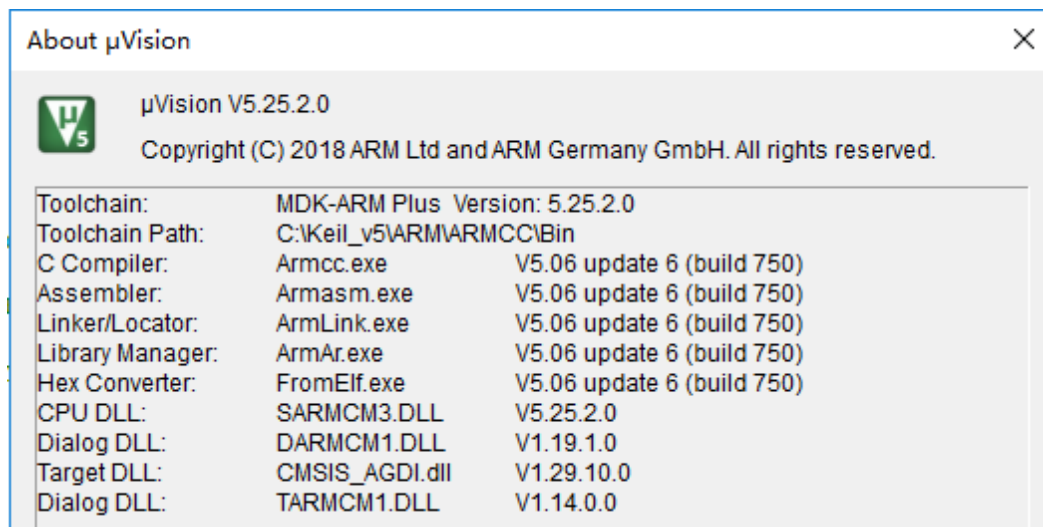


BAT32G135 Pack使用说明

Pack版本： 0.1.x (x:0~9)

使用前提

- 已经安装Keil MDK （本说明是基于V5.25.2.0版本下测试完成）

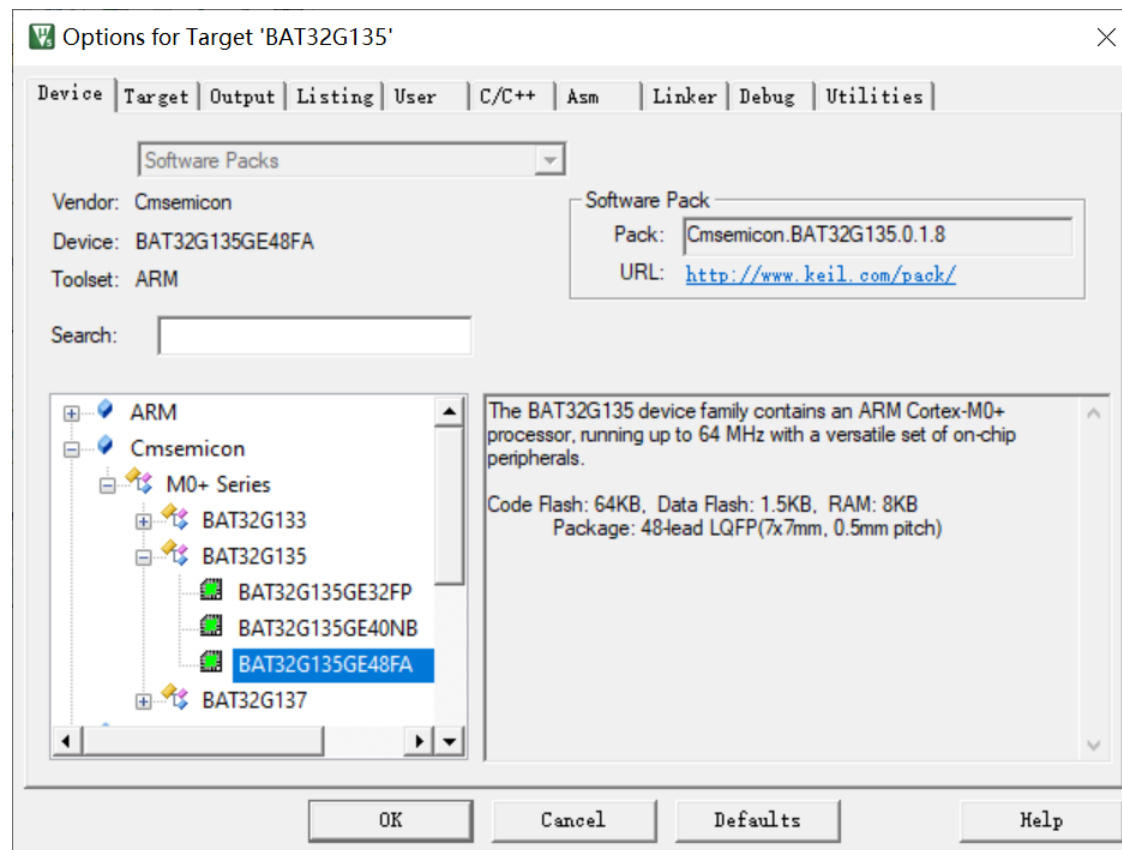


1. 安装BAT32G135 Pack

- 鼠标双击Cmsemicon.BAT32G135.0.1.x.pack即可安装。
- 安装完毕后可以在Keil的安装目录或用户的指定目录看到如下目录：
 - C:\Keil_v5\ARM\PACK\Cmsemicon\BAT32G135\0.1.x
 - 或者
 - C:\Users\Name\AppData\Local\Arm\Packs\Cmsemicon\BAT32G135\0.1.x
- 以上路径后面简称Pack安装目录

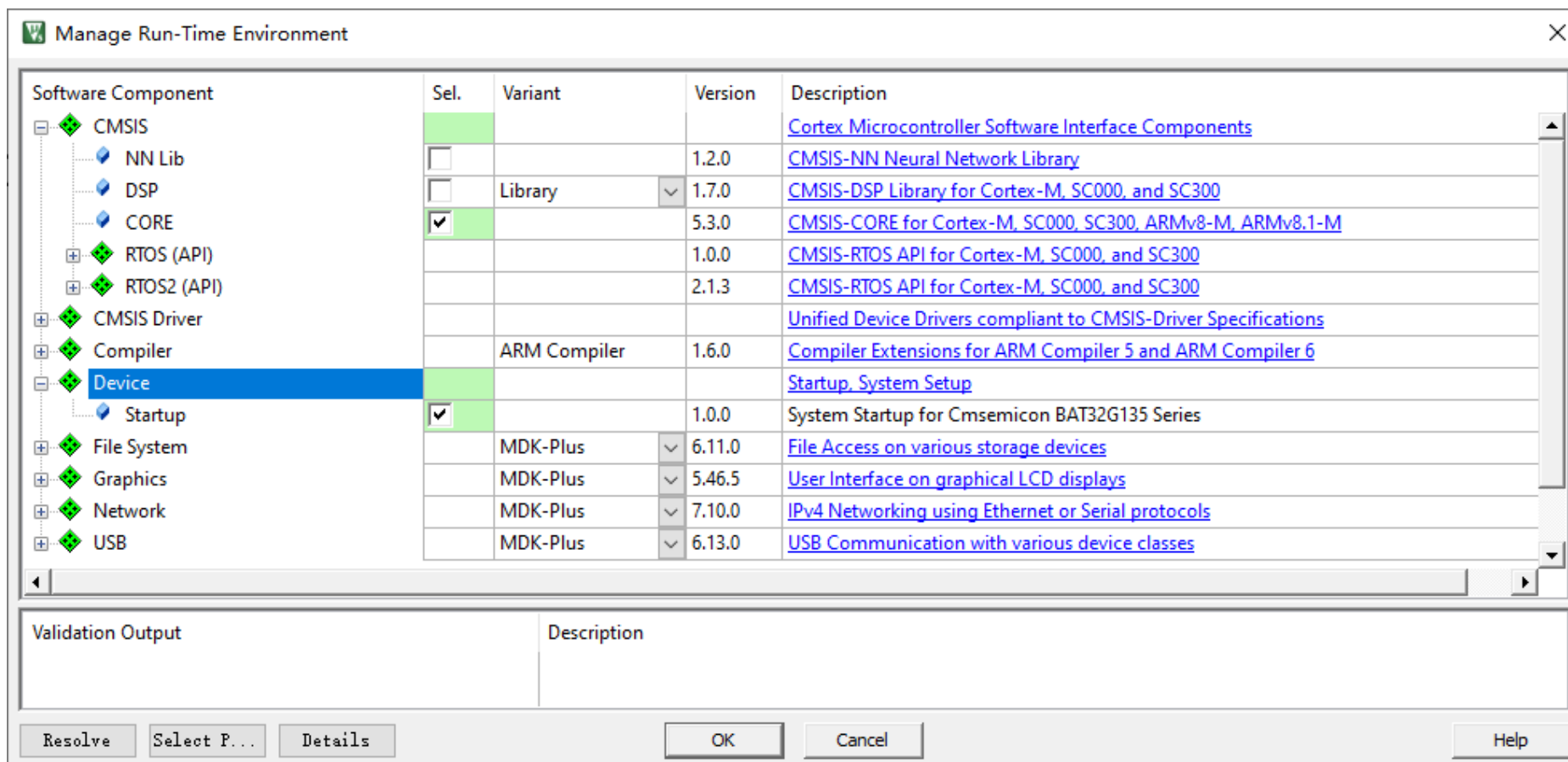
2.新建工程

- 2-1. 选择器件的时候，选择Cmsemicon/M0+ Series/BAT32G135下的相应器件。如下图例



2.新建工程

- 2-2. 运行时环境勾选CMSIS/CORE和Device/Startup。如下图例

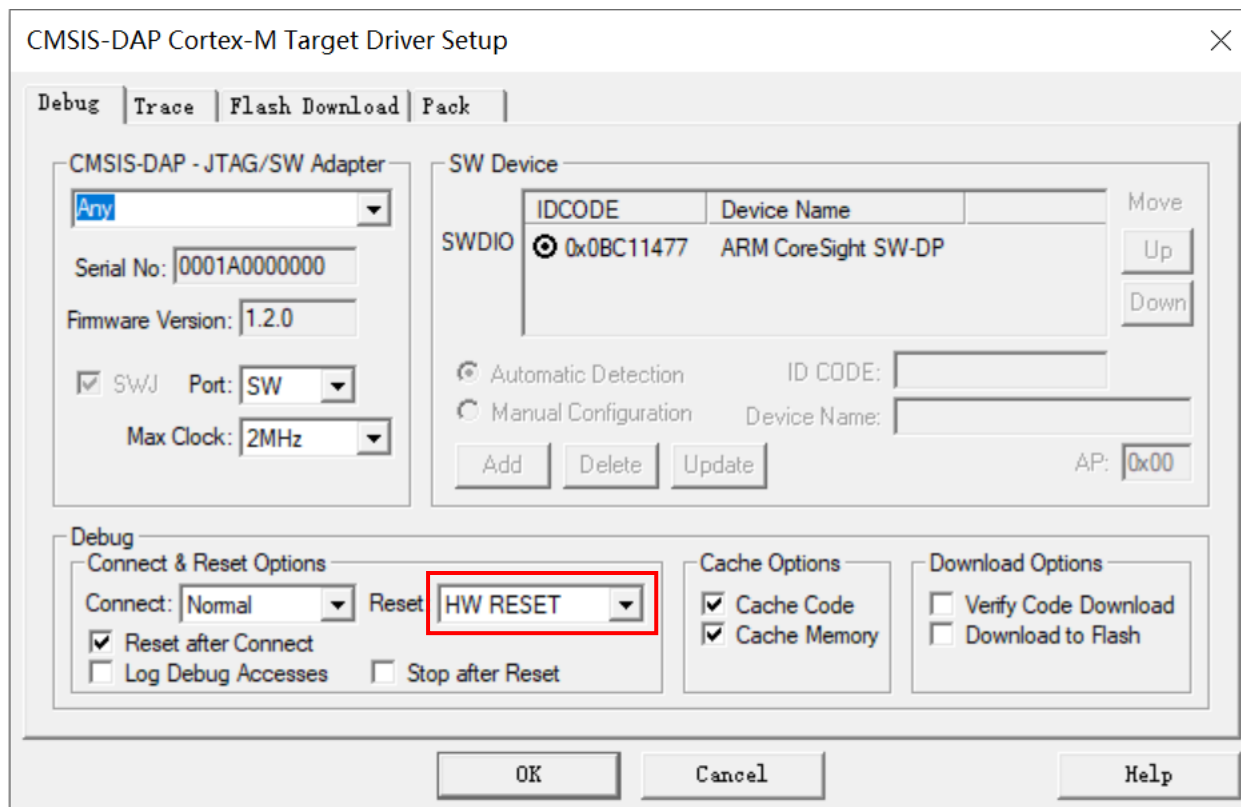
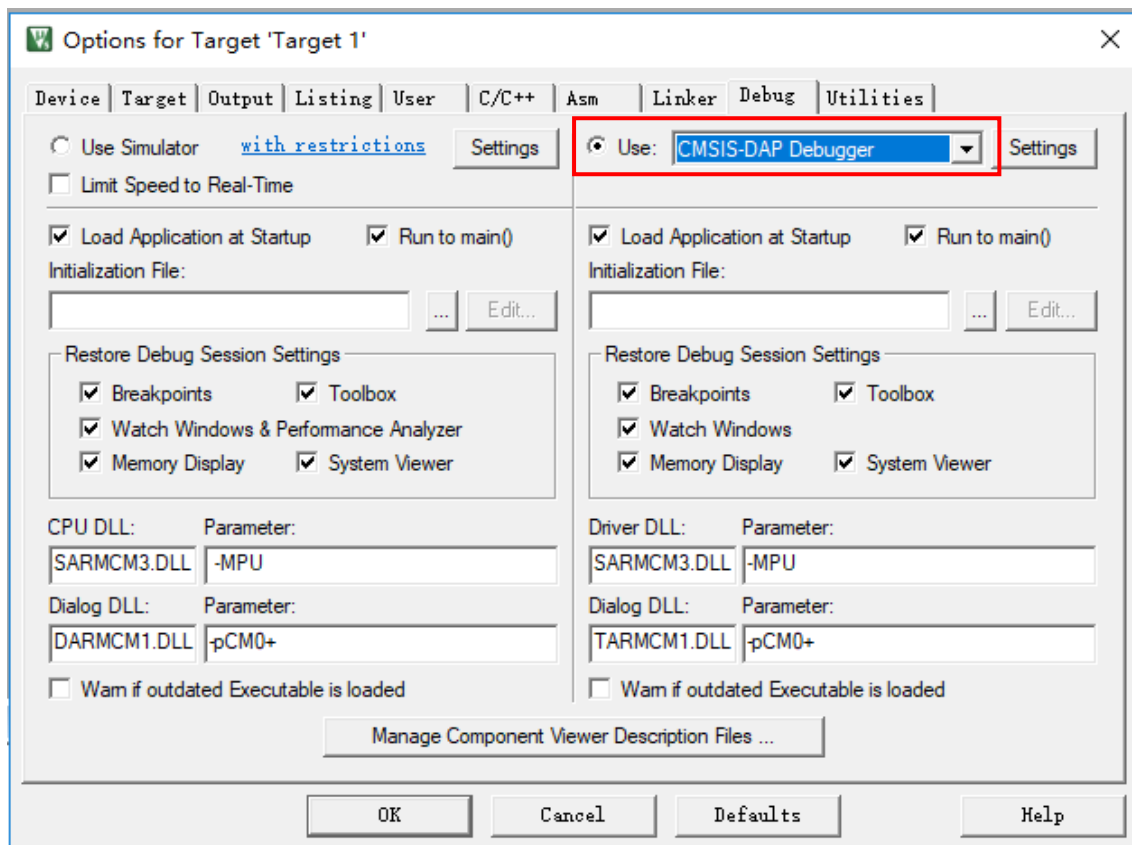


2.新建工程

- 2-3. 添加你自己写的程序到工程中，最简单的情况只需添加main.c程序即可。
- 在Pack安装目录下有Examples可供参考，例如
 - \$Pack安装目录\Examples\Blinky\main.c
- 提示：参考例程时，请把如下Examples和Driver目录拷贝到您自己的目录下，并保持Driver和Examples目录同级。
 - \$Pack安装目录\Driver
 - \$Pack安装目录\Examples

2.新建工程

- 2-4.选择你手里有的Debugger, 比如CMSIS-DAP Debugger



2.新建工程

- 2-5.配置选项字节（可选）

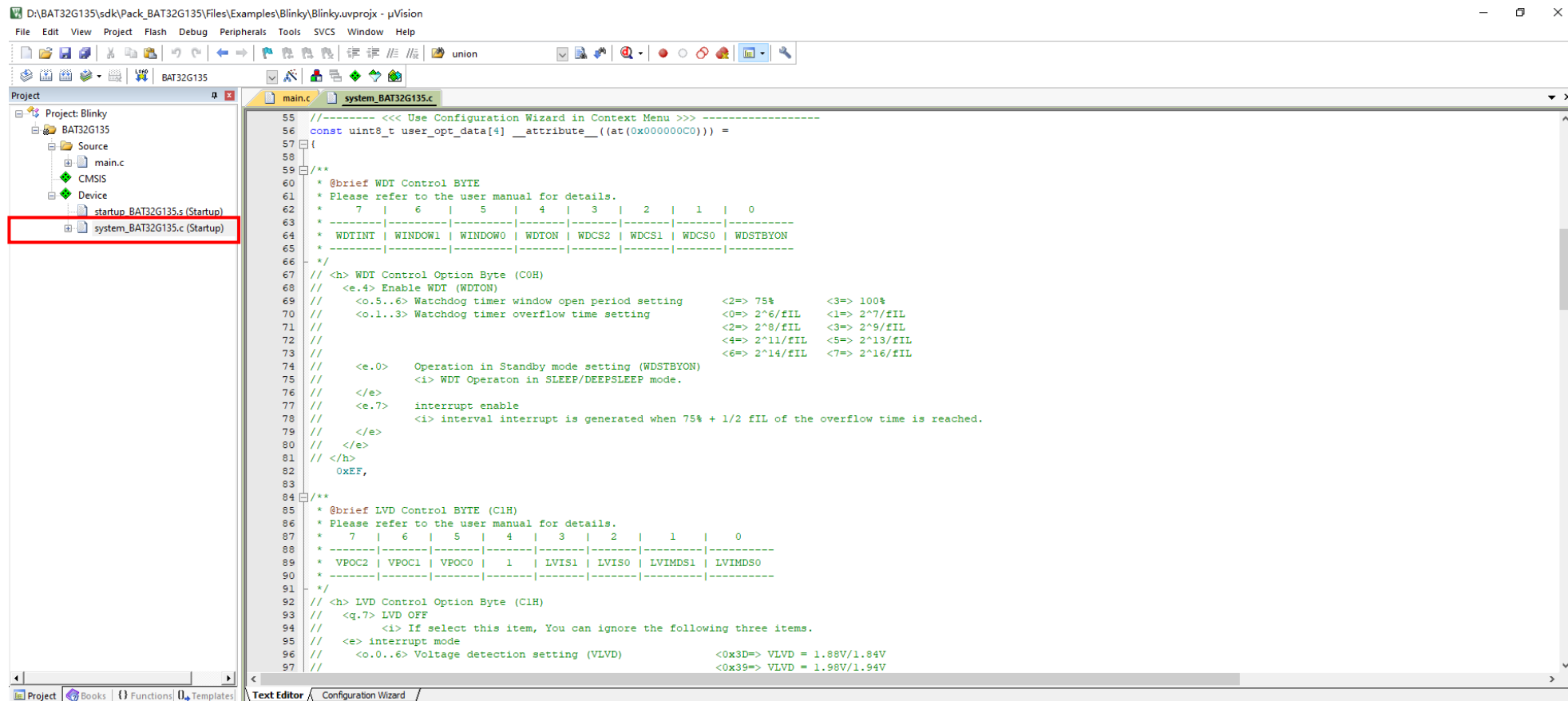
- 选项字节在Device/system_BAT32G135.c文件中设置。默认值为如下所示

- 第082行(C0H): 0xEF // WDT OFF
 - 第132行(C1H): 0xFF // LVD OFF
 - 第156行(C2H): 0xF8 // HOCO 64MHz (fHOCO = 64MHz, fIH = 64MHz)
 - 第167行(C3H): 0xFF // OCD EN

- 即可以直接修改代码中的值，也可以使用Configuration Wizard进行设置。

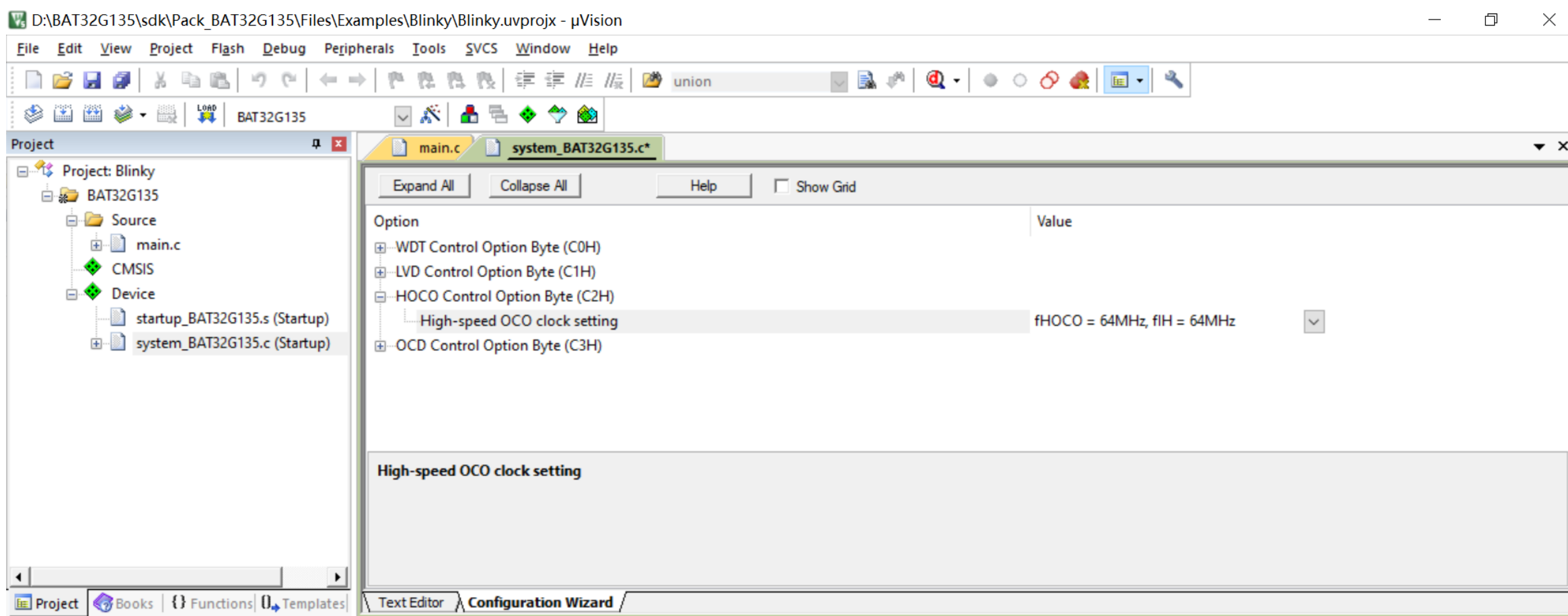
2-5. 配置选项字节 (续)

- 鼠标双击system_BAT32G135.c(Startup), 即可看到如下画面。在【Text Editor】中可以直接修改代码中的值



2-5. 配置选项字节 (续)

- 在【Configuration Wizard】中可以通过GUI方式修改相应的值



3. 分配兼用管脚

- BAT32G135中有些功能兼用只能分配到固定的管脚（后文简称固定兼用），有些功能兼用可以分配到任意管脚（后文简称任意兼用），为了方便用户根据需要分配兼用管脚而无需修改各个driver文件，在userdefine.h头文件中使用宏定义的方式集中定义了兼用管脚设置代码。注释中不带“ToDo”的是固定兼用，带“ToDo”的是任意兼用，请根据需要进行相应修改。

```
73
74 /* =====
75 /* ===== TM40 =====
76 /* =====
77 /**
78  * @brief TM40 TI00~3 and TO00~3 Port Setting
79  */
80
81 #define TI00_PORT_SETTING() do{ \
82     PORT->P00CFG = 0x00; /* P00 default alternate function */ \
83     PORT->PMCO &= ~(1 << 0); /* P00 digital function */ \
84     PORT->PM0 |= (1 << 0); /* P00 is used as TI00 input */ \
85 }while(0)
86
137 /* =====
138 /* ===== TM41 =====
139 /* =====
140
141 /**
142  * @brief TM41 TI10~3 and TO10~3 Port Setting
143  */
144
145 /* ToDo: You can allocate the TI10 to any desired pins with TI10PCFG register */
146 #define TI10_PORT_SETTING() do{ \
147     PORT->TI10PCFG = 0x03; /* allocate TI10 to P10 */ \
148     PORT->PMCl &= ~(1 << 0); /* P10 digital function */ \
149     PORT->PM1 |= (1 << 0); /* P10 is used as TI10 input */ \
150 }while(0)
```

固定兼用

任意兼用

4. 配置SCI的功能

- SCI的每个通道可以有UART、SPI和简易IIC三种功能，三种功能共用同一个中断请求信号。在sci.c和sci_user.c中已经把UART/SPI/IIC的相关函数都定义了，但实际使用中只能使用其中一种功能，因此请在userdefine.h的如下段落定义相应的宏，这里定义的宏在sci_user.c中决定把哪个中断服务程序分配到相应的中断入口。

The image displays two code editors side-by-side, illustrating the configuration of SCI (Serial Communication Interface) modules. The left editor shows the 'userdefine.h' file, and the right editor shows the 'sci_user.c' file.

Left Editor (userdefine.h):

- Macro definitions for SCI usage are provided, stating that each channel has three functions: UART, SPI, and IIC. Only one function can be chosen for each channel.
- For each channel (0, 1, 2), there are three macro definitions:
 - `USE_SCI_UARTx_TX`: Using CHx of SCI0 as UART Transmitter *
 - `USE_SCI_SPIx0`: Using CHx of SCI0 as SPI Transmitter or Receiver *
 - `USE_SCI_IICx0`: Using CHx of SCI0 as IIC Transmitter or Receiver *
- Similar definitions are provided for channels 1 and 2, using CH1 and CH2 respectively.

Right Editor (sci_user.c):

- Pragma directive is used to place the code in the flash memory.
- Interrupt handlers are defined for each channel:
 - `IRQ07_Handler`: Corresponds to channel 0.
 - `IRQ08_Handler`: Corresponds to channel 1.
 - `IRQ10_Handler`: Corresponds to channel 2.
- Each handler function is defined with a `__attribute__((alias(...)))` directive, linking it to the appropriate interrupt service routine (e.g., `uart2_interrupt_send`, `spi20_interrupt`, `iic20_interrupt`).

Annotations:

- Red arrows point from the macro definitions in the left editor to the corresponding interrupt handlers in the right editor.
- Red text "三选一" (Select one) is written next to the macro definitions for each channel, indicating that only one macro should be defined for each channel.

4. 配置SCI的功能

- SCI的每个通道可以有UART、SPI和简易IIC三种功能，三种功能共用同一个中断请求信号。在sci.c和sci_user.c中已经把UART/SPI/IIC的相关函数都定义了，但实际使用中只能使用其中一种功能，因此请在userdefine.h的如下段落定义相应的宏，这里定义的宏在sci_user.c中决定把哪个中断服务程序分配到相应的中断入口。

The image displays two code editors side-by-side, illustrating the configuration of SCI (Serial Communication Interface) modules in a C file. The left editor shows the 'userdefine.h' file, and the right editor shows the 'sci_user.c' file.

Left Editor (userdefine.h):

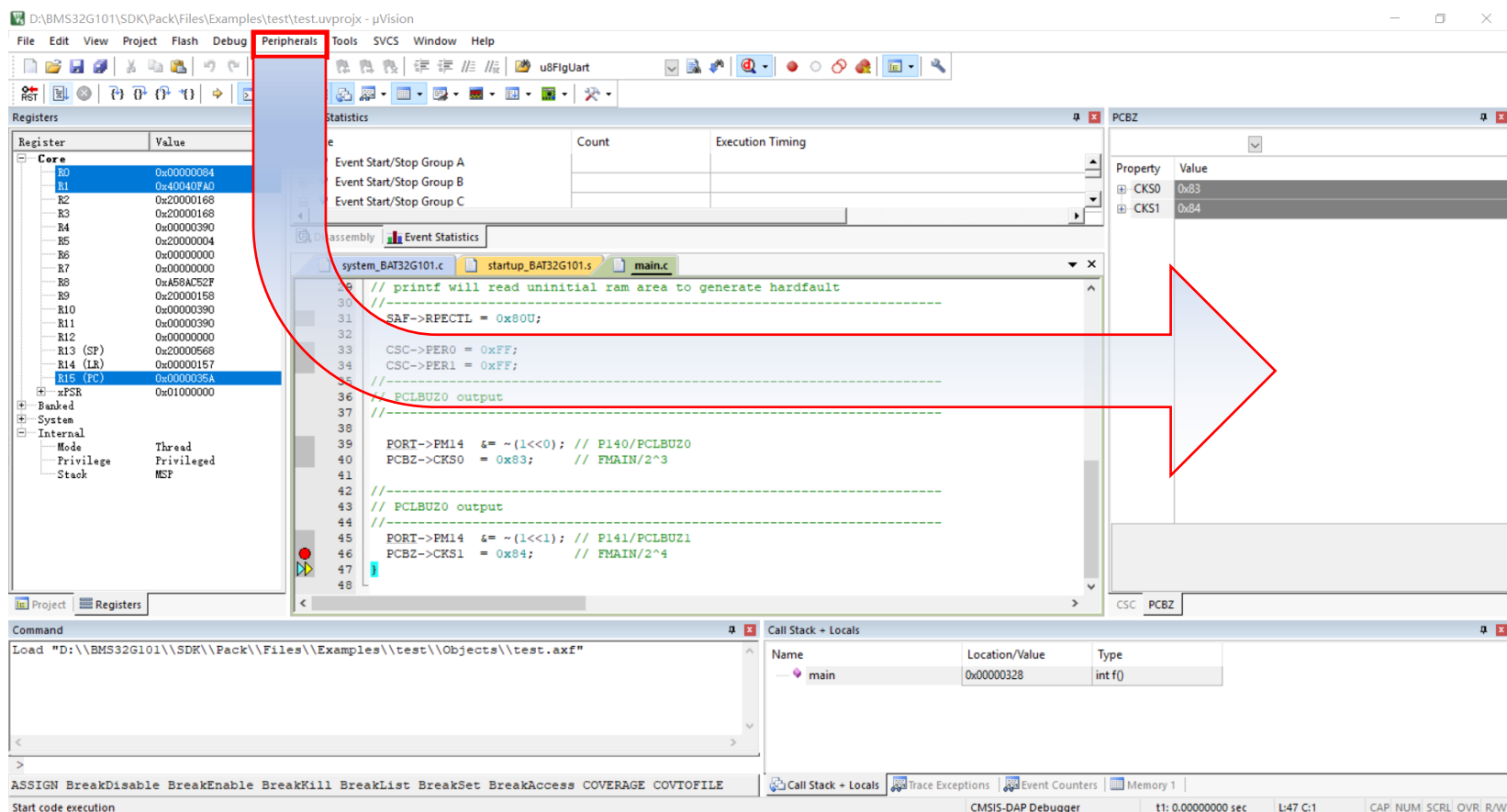
- Macro definitions of SCI usage: Each channel of SCI has three functions: UART, SPI, and IIC. You can only choose one function to use.
- Macro definitions for SCI0 and SCI1 channels, each with three options (UART, SPI, IIC) for Transmitter and Receiver.
- Red annotations with arrows point to the macro definitions, indicating a 'select one' (三选一) choice for each channel configuration.

Right Editor (sci_user.c):

- Pragma directive: #pragma GCC diagnostic ignored "-Wunused-function"
- Interrupt handlers for SCI0 and SCI1 channels, each with three options (UART, SPI, IIC) for Transmitter and Receiver.
- Red annotations with arrows point to the interrupt handlers, indicating a 'select one' (三选一) choice for each channel configuration.

5. 通过SVD窗口查看或修改SFR的值

- 在debugger时，可以通过SVD窗口查看或修改SFR的值



6. 参考文档

- 在Pack的Documents目录下有如下文档
 - BAT32G135_Pack使用说明.pdf // 本文档
 - ARM
 - DUI0662B_cortex_m0p_r0p1_dgug.pdf // Cortex-M0+ 官方用户手册
 - DataSheet
 - BAT32G135_datasheet
 - UserManual
 - BAT32G135用户手册

发现手册或Pack例程
问题欢迎反馈！
谢谢！