Introduction

This document is used to describe the installation and use of the PY32F0xx_DFP device package. This device package is suitable for the MDK integrated development environment. Before installing this device package, you need to install the MDK tool.

This device package cooperates with MDK software and emulator PY-LINK to realize the development and debugging of PY32F030 series MCU. It supports four functions: new project, compilation, download, and simulation debugging.

Contents

1 Installation

Before installing this device package, you need to install the MDK integrated development environment.
MDK download path: https://www2.keil.com/mdk5.

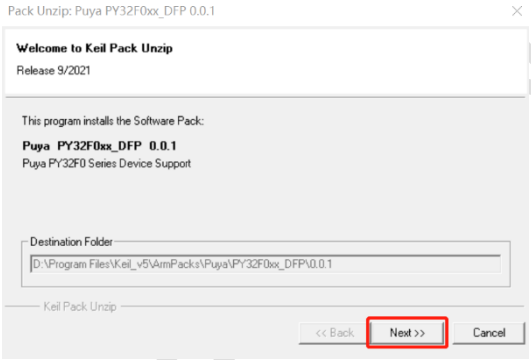
1.1 Install PY32F0xx_DFP

(1) Double-click the package

Figure 1-1 pack installation package

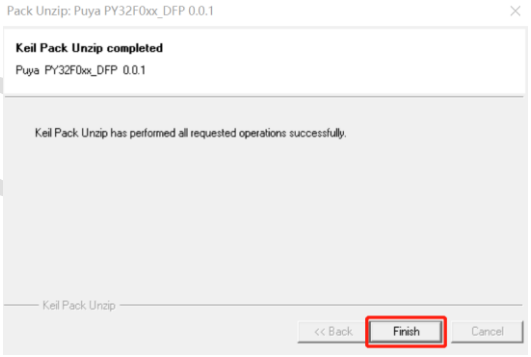
(2) Click the Next button

Figure 1-2 Installation path



(3) Click the Finish button

Figure 1-3 Installation successful reminder



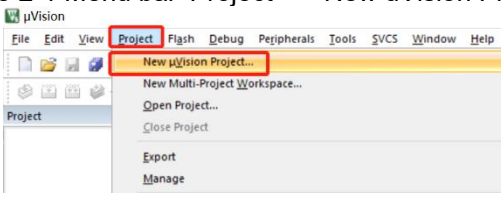
2 Use of DFP

This chapter include the flow of create new project, compile, download and debugging.

2.1 Create new project

(1) Create a new folder, name the file according to your preference, and open the MDK software

Figure 2-1 Menu bar 'Project'--> 'New uVision Project'



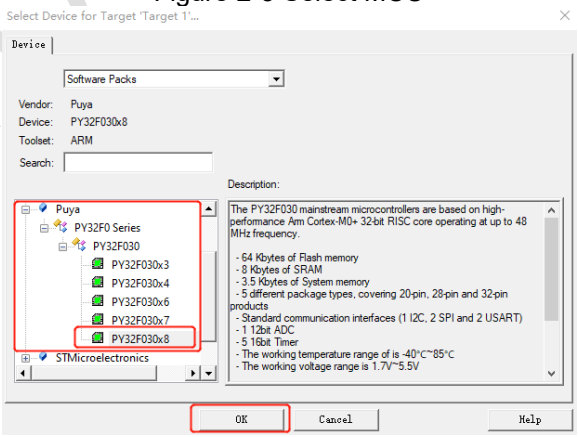
(2) Create New Project, save it in the new folder

Figure 2-2

(3) Select MCU

Select MCU according to the Start Kit.

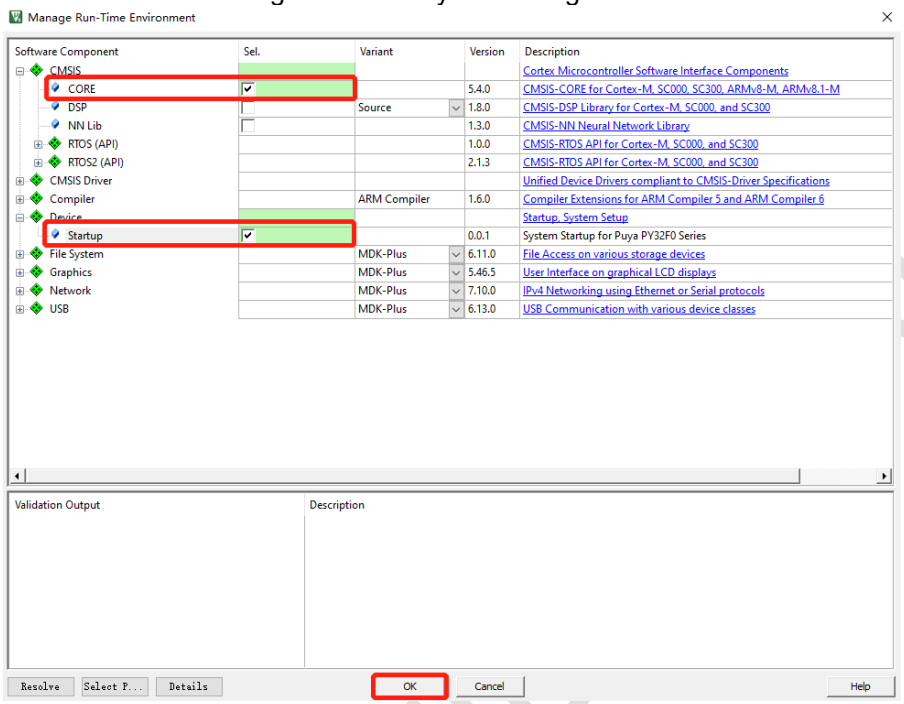
Figure 2-3 Select MCU



(4) Manage Run-Time Environment

Check 'CMSIS -- > CORE ' to add the core file, check ' Device --> Startup' to add the startup file.

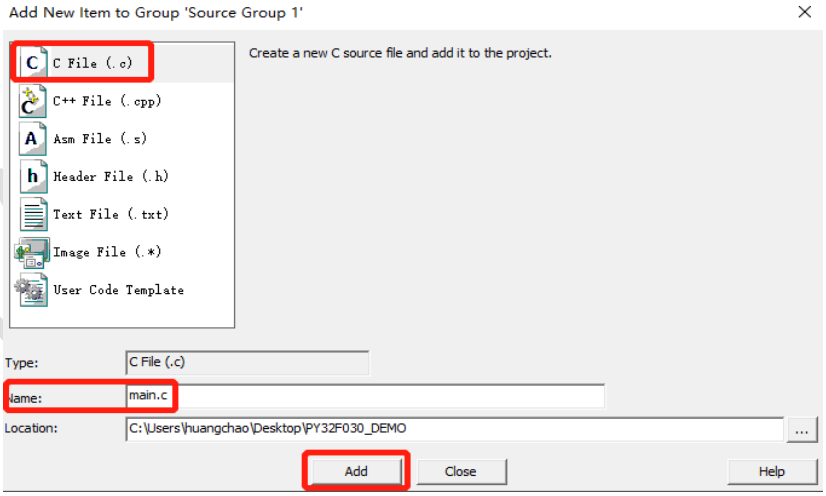
Figure 2-4 library file management



2.2 Add main.c file

- Right-click the 'Source Group1' folder in 'Target 1' in the Project window, and select " Add New Item to Group 'Source Group1'..." in the pop-up window
- Select C File (.c) in the pop-up dialog, fill in the main.c file name

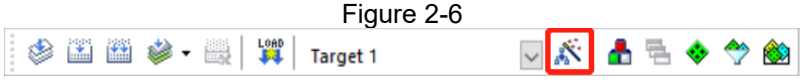
Figure 2-5 Add main.c file



2.3 Add PY32F030x8 macro

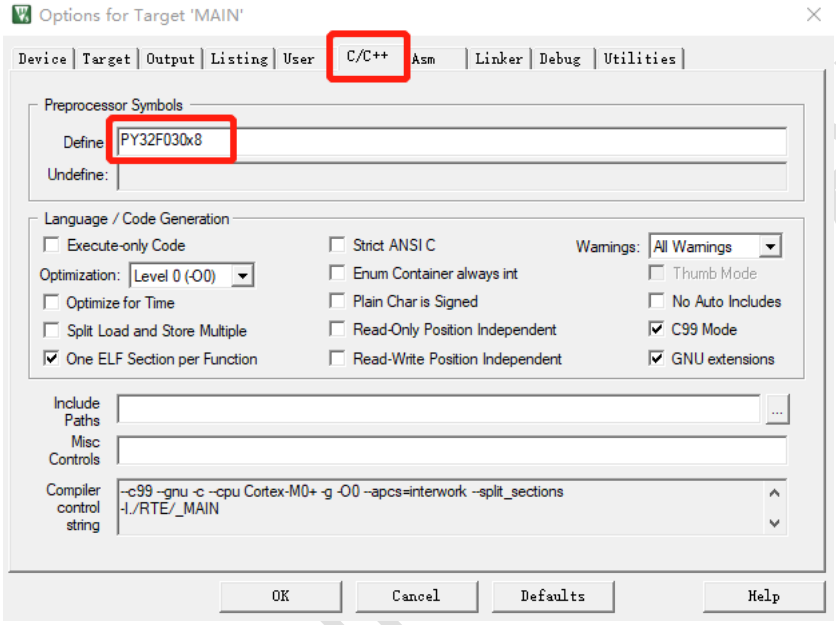
(1) Click the magic wand button in the toolbar to open the Options for Target window

Figure 2-6



(2) Enter the C / C++ page, add PY32F030x8 macro, as shown below

Figure 2-7



2.4 Edit main.c file

Note: Leave at least one blank line at the end of the file, otherwise a warning will be reported

Figure 2-8 main.c file

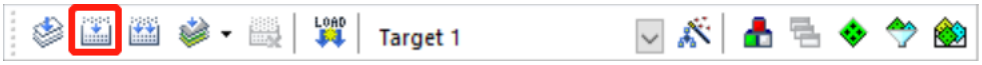
main.c

```
1  #include <py32f0xx.h>
2
3  void delay(uint32_t nTime);
4
5  int main(void)
6  {
7    SET_BIT(RCC->IOPENR, RCC_IOPENR_GPIOAEN);//GPIOA时钟使能
8
9    //设置PA11为通用输出模式
10   MODIFY_REG(GPIOA->MODER, GPIO_MODER_MODE11_1, GPIO_MODER_MODE11_0);
11
12   while (1)
13   {
14     SET_BIT(GPIOA->BSRR, GPIO_BSRR_BS11);//PA11输出高电平
15     delay(0x3FFFF);//软件延时
16
17     SET_BIT(GPIOA->BSRR, GPIO_BSRR_BR11);//PA11输出低电平
18     delay(0x3FFFF);//软件延时
19   }
20 }
21
22 void delay(uint32_t nTime)
23 {
24   while (nTime--);
25 }
26
27
```

2.5 Compile

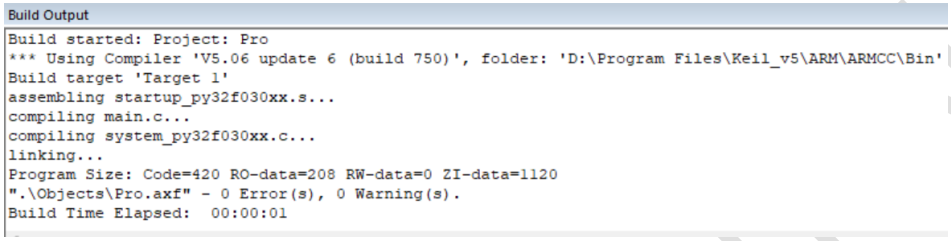
Click the toolbar ' build ' button to start compiling.

Figure 2-9



When ".\Objects\Pro.axf" - 0 Error(s), 0 Warning(s) appears, it means the program is compiled successfully.

Figure 2-10 Build Output



2.6 Download

(1) Connect the emulator with the PC through USB, and connect MCU with SWD, as shown in Figure 2-18

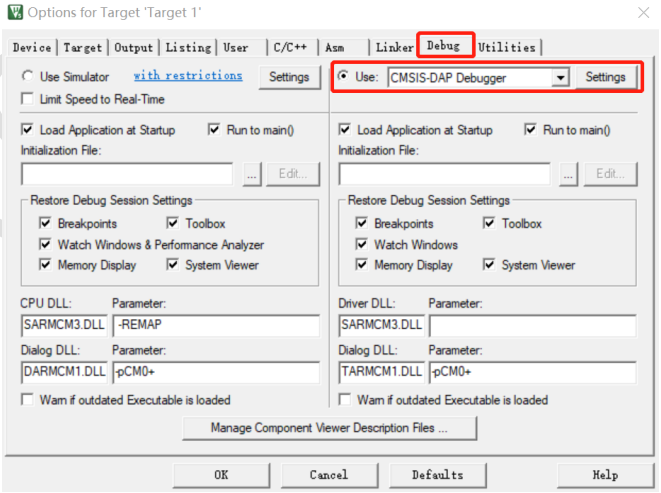
PY32F030 uses SWD to download and debug. Please connect TVC, SWDIO, GND, SWCLK, RST of the emulator to VDD, SWDIO(PA13), GND, SWCLK(PA14), NRST(PF2) of MCU/Start Kit in sequence.

(2) Click the magic wand button in the toolbar to open the Options for Target window

Figure 2-11

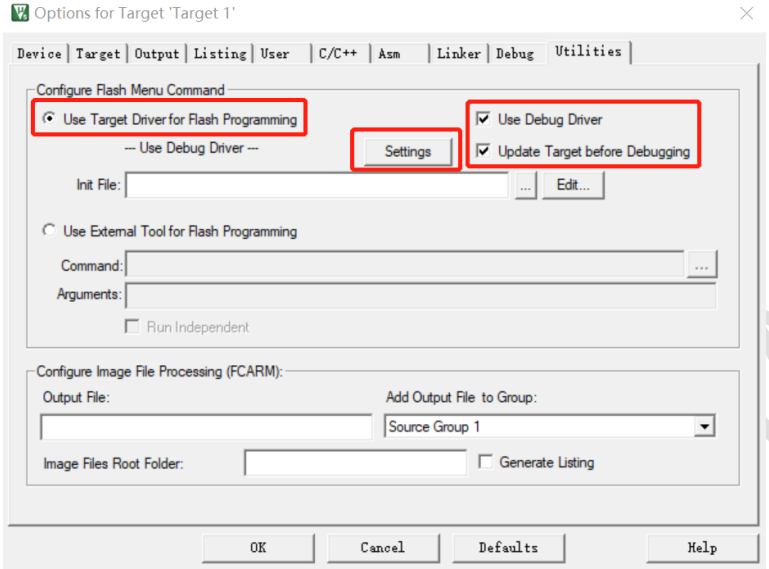
(3) Enter the Debug page, the emulator PY-LINK selects CMSIS-DAP Debugger

Figure 2-12 Debug option configuration



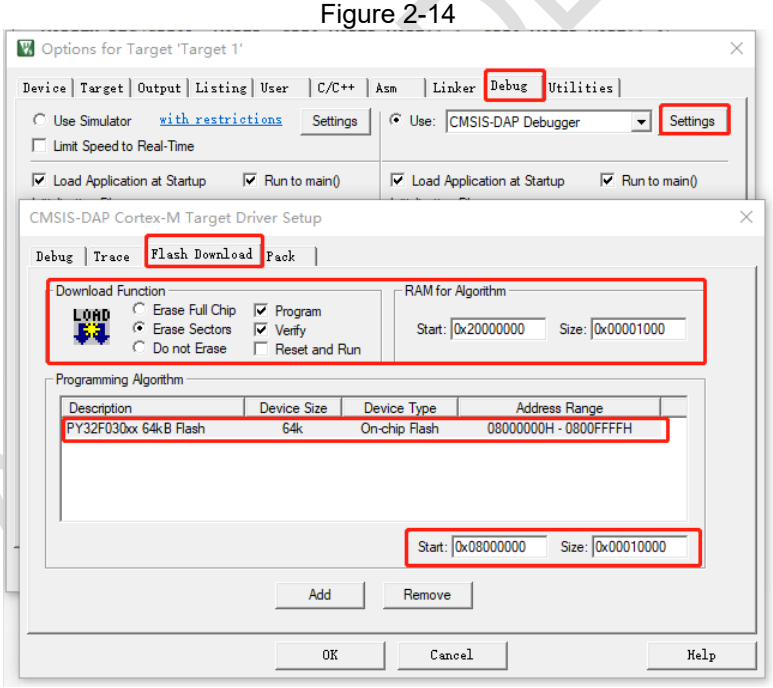
(4) Enter the Utilities page, please confirm whether the configuration is correct

Figure 2-13 Utilities option configuration



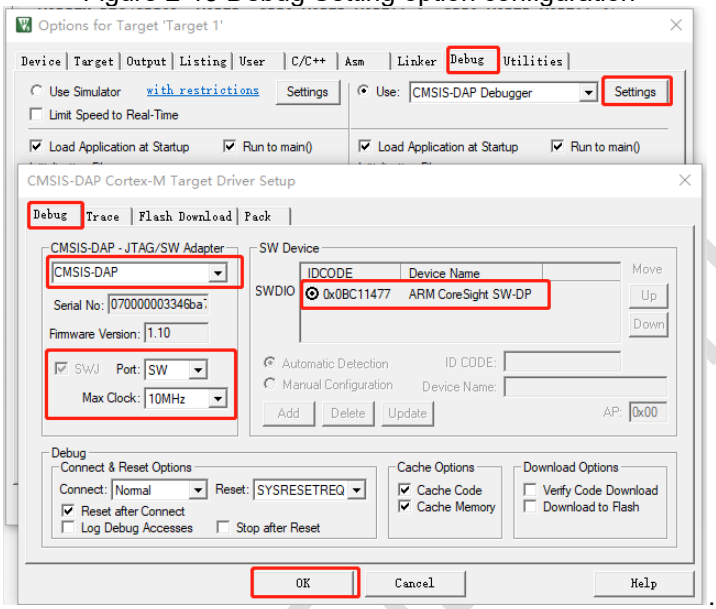
(5) Enter the Flash Download page and select the Programming Algorithm according to the target chip

Figure 2-14



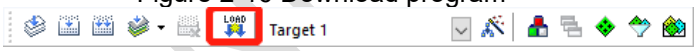
(6) Enter the Debug page and complete the emulator configuration
If the emulator is connected to the computer, MDK will recognize the emulator in 'CMSIS-DAP-JTAG/SW Adapter', and if the development board is powered on at the same time, the chip of the Start Kit will be recognized in 'SW Device' and displayed. Select the SW interface and configure the Max Clock according to the actual situation (can be configured to 10MHZ, if the download fails, you can reduce the Clock here).

Figure 2-15 Debug Setting option configuration



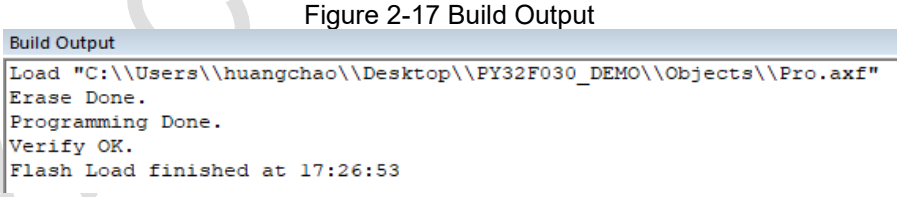
(7) Click the Download button on the toolbar to start the download

Figure 2-16 Download program



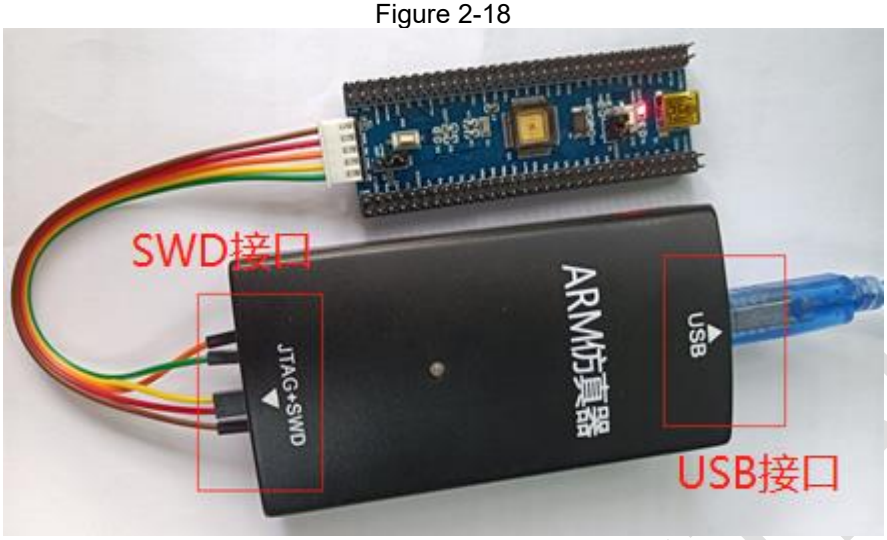
After the program is downloaded, the 'Build Output' tab prints the result as shown in Figure 2-17
Erase Done: Erase Done
Programing Done: Programming is complete
Verify OK: Verification is successful

Figure 2-17 Build Output



(8) After pressing the reset button of the development board, the LED light connected to the PA11 flashes

Figure 2-18



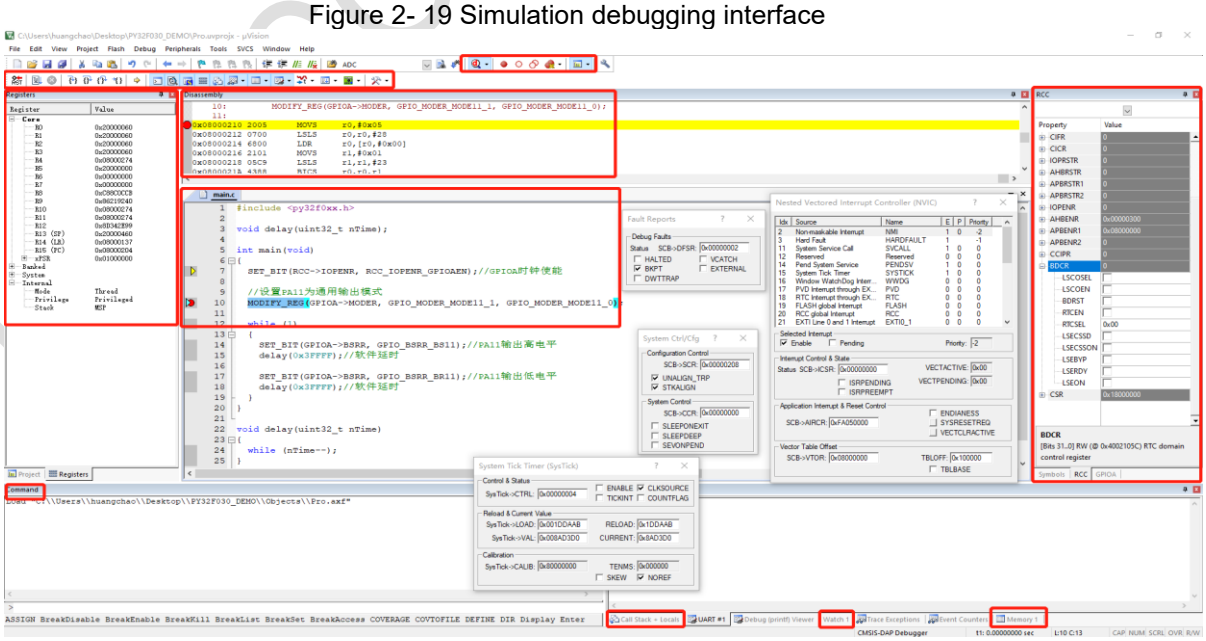
2.7 Debugging

(1) Click the 'Start/Stop Debug Session' button on the toolbar to enter the debugging interface. The leftmost part of the emulation debugging page shows the current values and system information of some registers inside the microcontroller. The above is the code that keil converts C language into assembly. The following is the C program we wrote. Or set the value of all peripheral registers of the MCU, at the bottom is 'Watch 1' to view the value of any variable, which is convenient for us to track and find errors. There is a yellow arrow in the C language and assembly language windows, which represents the current position of the program.

In the toolbar at the top of the debugging page, there are several buttons: the first one marked with RST is reset, after clicking, the program will run to the starting position; then the second button is to run at full speed, after clicking, the program will run at full speed; then the third button is the stop button. When the program runs at full speed, click the stop button and the program will stop immediately, and you can observe where the program is running; Single-step operation, please refer to the diagram for specific entry or exit.

After clicking Reset, you can see that the left side of the C language program window is gray or maintains the original color, and the gray area is where we can set breakpoints.

Figure 2- 19 Simulation debugging interface



(2) General debugging functions supported

- Start/Stop Debug Session (Enter or leave a debug session)
- Insert/Remove Breakpoint (Insert or remove a breakpoint at the current line)
- Disable All Breakpoints in current Target
- Kill All Breakpoints in current/active Target

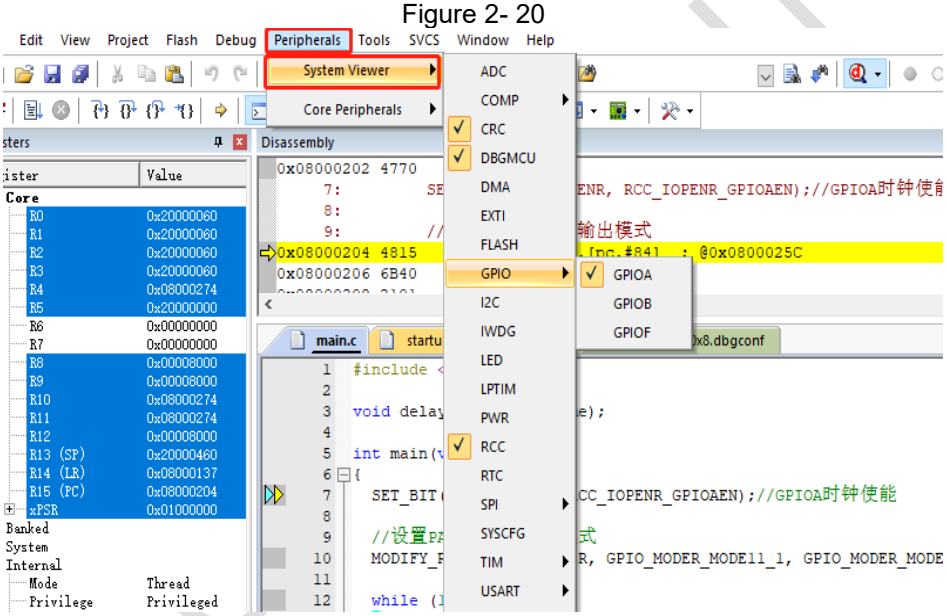
- Reset (Reset the CPU)
- Run (Start code execution)
- Step (Step one line)
- Step Over (Step over the current line)
- Step Out (Step out of the current function)
- Run to Cursor Line (Run to the current cursor line)
- Command Window
- Disassembly Window
- Registers Window
- Call Stack Window
- Watch Windows
- Memory Windows

For the usage of these general debugging functions, please refer to the Keil MDK user manual, which will not be repeated here.

(3) Peripheral register viewing window

Click on the toolbar 'Peripherals' --> 'System Viewer' to view or set the values of all peripheral registers of the MCU.

Figure 2- 20



(4) Preset DBGMCU register

Click 'Edit' to set the value of the DBGMCU register before entering debugging.

In order to debug after the MCU enters STOP /SLEEP mode, the device support package has set the value of the DBGMCU_CR register to 0x00000002 by default, that is, the DBG_STOP bit is 1, corresponding to the DBG_STOP bit on the right side of Figure 2-23 being checked.

the *.dbgconf file in Figure 2-21 according to actual needs, just modify the three places marked in the red box, which correspond to the DBGMCU_CR, DBG_APB_FZ1 and DBG_APB_FZ2 registers of the P Y32F030 respectively. Please refer to the P Y32F030 data sheet for the definitions of the registers and bits.

Figure 2-21

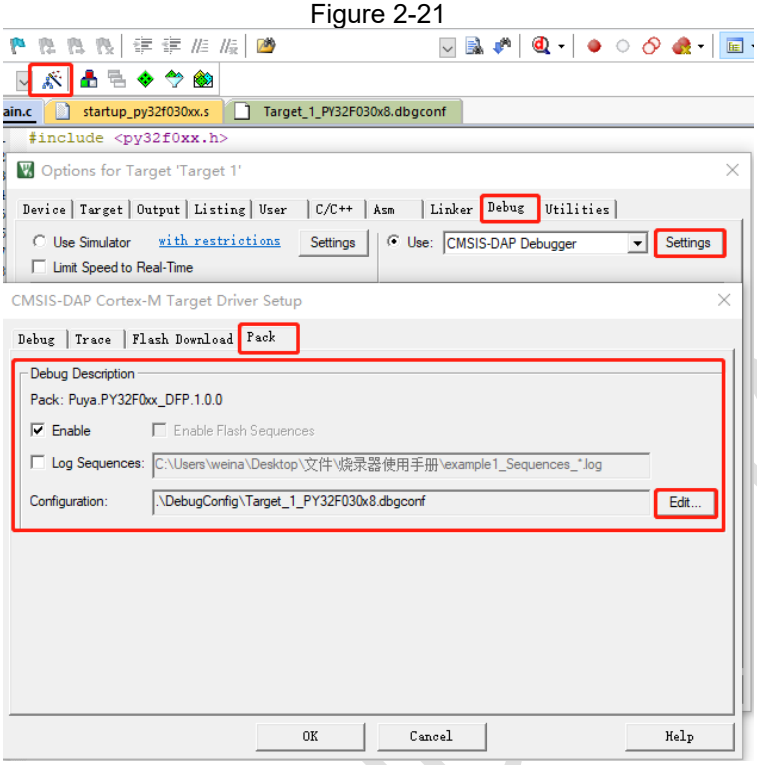
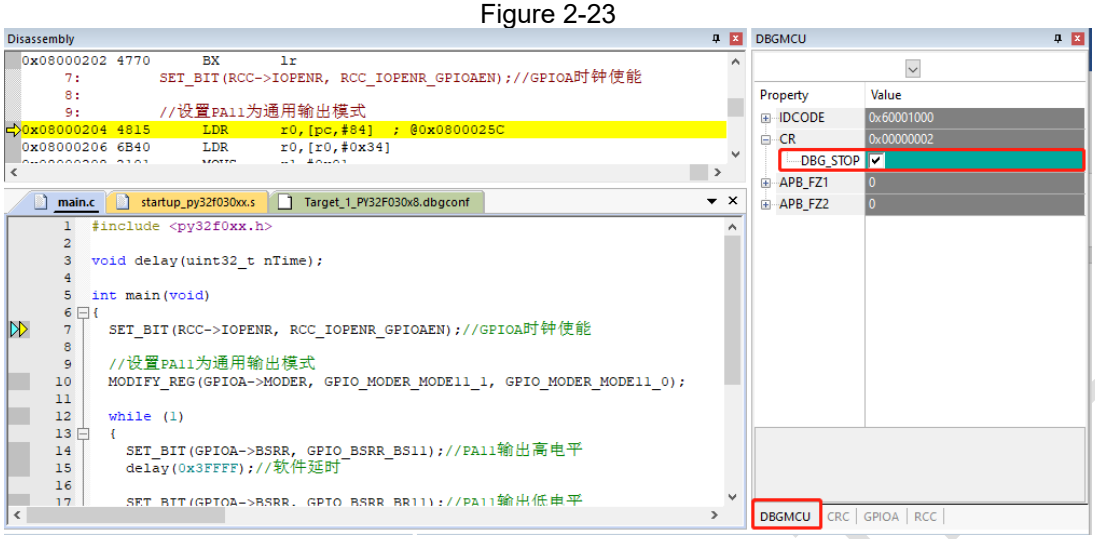


Figure 2-22

Target_1_PY32F030x8.dbgconf

```
// File: PY32F030xx.dbgconf
// Version: 1.0.0

// <<< Use Configuration Wizard in Context Menu >>>

// <h> Debug MCU configuration register (DBGMCU_CR)
//   <o.1>  DBG_STOP                <i> Debug stop mode
// </h>
DbgMCU_CR = 0x00000002;

// <h> Debug MCU APB freeze1 register (DBG_APB_FZ1)
//                                  <i> Reserved bits must be kept at reset value
//   <o.31> DBG_LPTIM_STOP          <i> LPTIM stopped when core is halted
//   <o.12> DBG_IWDG_STOP           <i> Independent watchdog stopped when core is halted
//   <o.11> DBG_WWDG_STOP           <i> Window watchdog stopped when core is halted
//   <o.10> DBG_RTC_STOP            <i> RTC stopped when core is halted
//   <o.4>  DBG_TIM6_STOP           <i> TIM6 counter stopped when core is halted
//   <o.1>  DBG_TIM3_STOP           <i> TIM3 counter stopped when core is halted
// </h>
DbgMCU_APB_Fz1 = 0x00000000;

// <h> Debug MCU APB freeze2 register (DBG_APB_FZ2)
//                                  <i> Reserved bits must be kept at reset value
//   <o.18> DBG_TIM17_STOP          <i> TIM17 counter stopped when core is halted
//   <o.17> DBG_TIM16_STOP          <i> TIM16 counter stopped when core is halted
//   <o.15> DBG_TIM14_STOP          <i> TIM14 counter stopped when core is halted
//   <o.11> DBG_TIM1_STOP           <i> TIM1 counter stopped when core is halted
// </h>
DbgMCU_APB_Fz2 = 0x00000000;

// <<< end of configuration section >>>
```

Figure 2-23



3 Version history

Version	date	Date
Rev 0.1	Initial Release	2021-09-18
Rev 1.0	Add use of DFP	2022-07-05

Puya Semiconductor Co., Ltd.