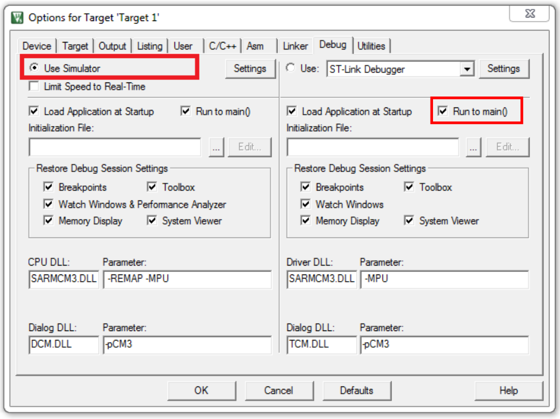
# LAB: GPIO Digital I/O

# - Tutorial: keil uVision 5 Debugging –

## Software *vs* Hardware Debug

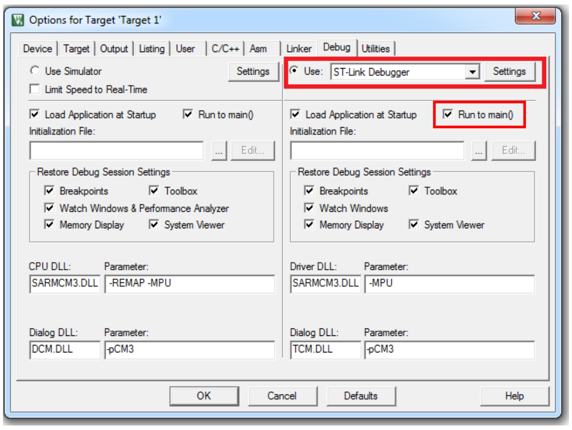
- There are two methods to debug your program: software debug and hardware debug.
  - Software Debug: you do not have to have the hardware board to debug a software program.
  - Hardware Debug: MCU board must be connected to the computer.
- For this tutorial, we will use Hardware Debugging


Options for Target 'Target 1'
Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities
Settings
Edit...
OK
Cancel
Defaults
Help

● Use Simulator  
□ Limit Speed to Real-Time  
☑ Load Application at Startup ☑ Run to main()  
Initialization File:  
Restore Debug Session Settings: ☑ Breakpoints ☑ Toolbox ☑ Watch Windows & Performance Analyzer ☑ Memory Display ☑ System Viewer  
CPU DLL: SARMCM3.DLL Parameter: -REMAP -MPU  
Dialog DLL: DCM.DLL Parameter: -pCM3

○ Use: ST-Link Debugger  
☑ Load Application at Startup ☑ Run to main()  
Initialization File:  
Restore Debug Session Settings: ☑ Breakpoints ☑ Toolbox ☑ Watch Windows ☑ Memory Display ☑ System Viewer  
Driver DLL: SARMCM3.DLL Parameter: -MPU  
Dialog DLL: TCM.DLL Parameter: -pCM3

Selecting **software** debug


Options for Target 'Target 1'
Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities
Settings
Edit...
OK
Cancel
Defaults
Help

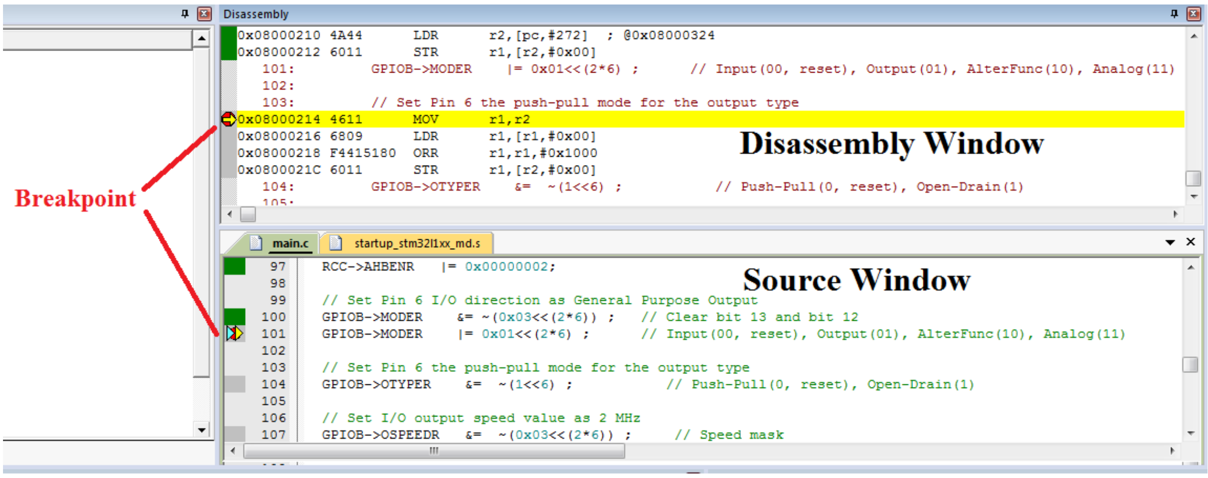
○ Use Simulator  
□ Limit Speed to Real-Time  
☑ Load Application at Startup ☑ Run to main()  
Initialization File:  
Restore Debug Session Settings: ☑ Breakpoints ☑ Toolbox ☑ Watch Windows & Performance Analyzer ☑ Memory Display ☑ System Viewer  
CPU DLL: SARMCM3.DLL Parameter: -REMAP -MPU  
Dialog DLL: DCM.DLL Parameter: -pCM3

● Use: ST-Link Debugger  
☑ Load Application at Startup ☑ Run to main()  
Initialization File:  
Restore Debug Session Settings: ☑ Breakpoints ☑ Toolbox ☑ Watch Windows ☑ Memory Display ☑ System Viewer  
Driver DLL: SARMCM3.DLL Parameter: -MPU  
Dialog DLL: TCM.DLL Parameter: -pCM3

Selecting **hardware** debug

## Debug Control

- **Compile, Debug, and Run**
  - You can program the STM32 flash by clicking the LOAD button .
  - Click the debug button to start the debug and click it again to exit the debug.
  - You can use the breakpoint button to set a break point in either disassembly or source windows.
  - STM32 allows up to six breakpoints during hardware debugging. When a program stops at a breakpoint, the corresponding instruction has not been executed yet.
  - You can choose either debugging *assembly code* or *C-code* line by choosing the disassembly or source window in focus.



```
0x08000210 4A44      LDR      r2,[pc,#272]  ; @0x08000324
0x08000212 6011      STR      r1,[r2,#0x00]
   101:        GPIOB->MODER    |= 0x01<<(2*6) ;      // Input(00, reset), Output(01), AlterFunc(10), Analog(11)
   102:
   103:        // Set Pin 6 the push-pull mode for the output type
0x08000214 4611      MOV      r1,r2
0x08000216 6809      LDR      r1,[r1,#0x00]
0x08000218 F4415180  ORR      r1,r1,#0x1000
0x0800021C 6011      STR      r1,[r2,#0x00]
   104:        GPIOB->OTYPER    &=  ~(1<<6) ;           // Push-Pull(0, reset), Open-Drain(1)
   105:
```

```
RCC->AHBENR    |= 0x00000002;

// Set Pin 6 I/O direction as General Purpose Output
GPIOB->MODER    &= ~(0x03<<(2*6)) ;   // Clear bit 13 and bit 12
GPIOB->MODER    |= 0x01<<(2*6) ;      // Input(00, reset), Output(01), AlterFunc(10), Analog(11)

// Set Pin 6 the push-pull mode for the output type
GPIOB->OTYPER    &=  ~(1<<6) ;           // Push-Pull(0, reset), Open-Drain(1)

// Set I/O output speed value as 2 MHz
GPIOB->OSPEEDR   &=  ~(0x03<<(2*6)) ;     // Speed mask
```

– Control buttons of debugging

| Start Debug | Set a Breakpoint | Run | Stop Debug | Step In | Step Over | Step Out |
|---|---|---|---|---|---|---|

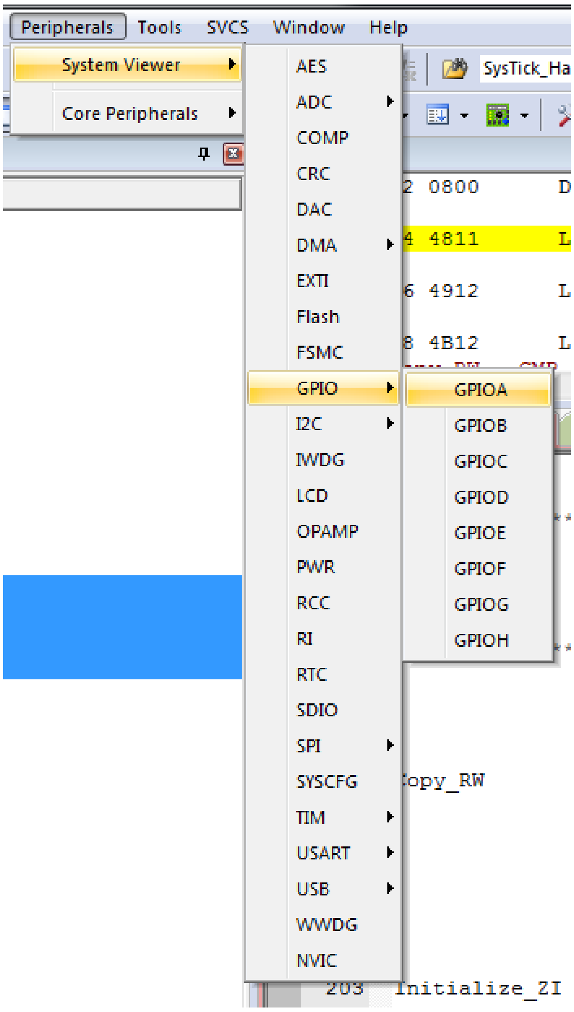
- **Run(F5)**: Continues the execution from the current position until you click ***Stop*** or the program is paused by a breakpoint.
- **Step In(F11)**: Execute one step and enter the function if the current step calls a function.
- **Step Out(Ctrl+F11)**: Execute until the current function returns.
- **Step Over(F10)**: Execute one step and run the function all at once if the current step calls a function.

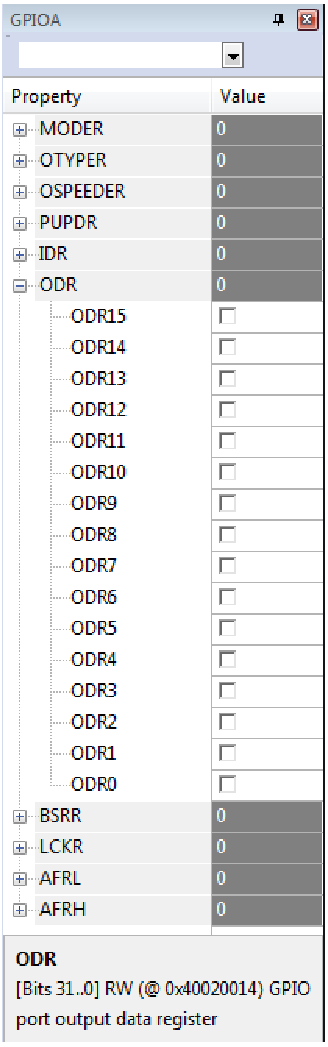
## Peripheral Registers

– Choose the menu: **Peripherals ⟶ System Viewer**

– View and update the control and data registers of all available peripherals

– Example: Registers for GPIO Port A. Can view specific registers such as Output Data Register (ODR) for debugging

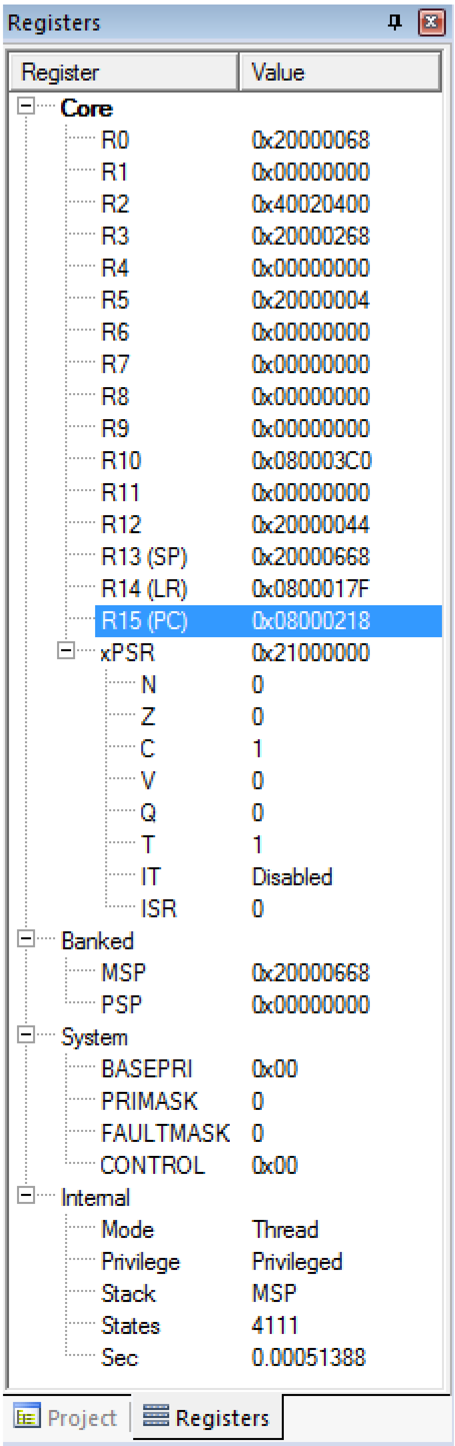




| Property | Value |
|---|---|
| MODER | 0 |
| OTYPER | 0 |
| OSPEEDER | 0 |
| PUPDR | 0 |
| IDR | 0 |
| ODR | 0 |
| ODR15 | ☐ |
| ODR14 | ☐ |
| ODR13 | ☐ |
| ODR12 | ☐ |
| ODR11 | ☐ |
| ODR10 | ☐ |
| ODR9 | ☐ |
| ODR8 | ☐ |
| ODR7 | ☐ |
| ODR6 | ☐ |
| ODR5 | ☐ |
| ODR4 | ☐ |
| ODR3 | ☐ |
| ODR2 | ☐ |
| ODR1 | ☐ |
| ODR0 | ☐ |
| BSRR | 0 |
| LCKR | 0 |
| AFRL | 0 |
| AFRH | 0 |

**ODR**
[Bits 31..0] RW (@ 0x40020014) GPIO port output data register

## Processor Registers



| Register | Value |
|---|---|
| Core | |
| R0 | 0x20000068 |
| R1 | 0x00000000 |
| R2 | 0x40020400 |
| R3 | 0x20000268 |
| R4 | 0x00000000 |
| R5 | 0x20000004 |
| R6 | 0x00000000 |
| R7 | 0x00000000 |
| R8 | 0x00000000 |
| R9 | 0x00000000 |
| R10 | 0x080003C0 |
| R11 | 0x00000000 |
| R12 | 0x20000044 |
| R13 (SP) | 0x20000668 |
| R14 (LR) | 0x0800017F |
| R15 (PC) | 0x08000218 |
| xPSR | 0x21000000 |
| N | 0 |
| Z | 0 |
| C | 1 |
| V | 0 |
| Q | 0 |
| T | 1 |
| IT | Disabled |
| ISR | 0 |
| Banked | |
| MSP | 0x20000668 |
| PSP | 0x00000000 |
| System | |
| BASEPRI | 0x00 |
| PRIMASK | 0 |
| FAULTMASK | 0 |
| CONTROL | 0x00 |
| Internal | |
| Mode | Thread |
| Privilege | Privileged |
| Stack | MSP |
| States | 4111 |
| Sec | 0.00051388 |

Core Registers:

- Program counter (PC) r15 holds the memory address (location in memory) of the next instruction to be fetched from the instruction memory.
- Stack point (SP) r13 holds a memory address that points to the top of the stack. SP is a shadow of either MSP or PSP.
- xPSR (Special-purpose program status registers) is a combination of the following three processor status registers:
    - Application PSR
    - Interrupt PSR
    - Execution PSR

| | |
|---|---|
| N | Negative or less than flag (1 = result negative) |
| Z | Zero flag (1 = result 0) |
| C | Carry or borrow flag (1 = Carry true or borrow false) |
| V | Overflow flag (1 = overflow) |
| Q | Q Sticky saturation flag |
| T | Thumb state bit |
| IT | If-Then bits |
| ISR | ISR Number ( 6 bits ) |