香港中文大學
The Chinese University of Hong Kong

CENG2400 Embedded System Design

Tutorial 01:
IDE Installation & HelloWorld

Chenchen ZHAO

cczhao@cse.cuhk.edu.hk

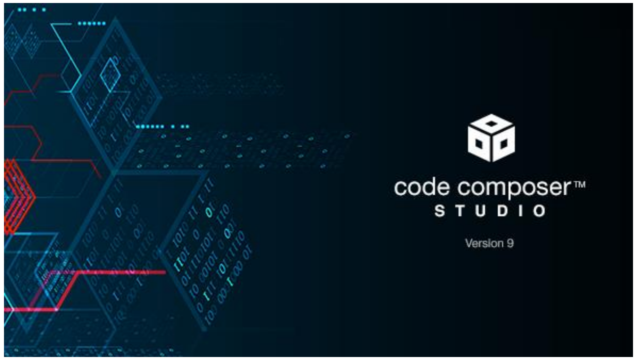
Outline

- Step 1: installing and checking IDE & SDK
- Step 2: getting familiar with the Tiva LaunchPad
- Step 3: building and running a HelloWorld program

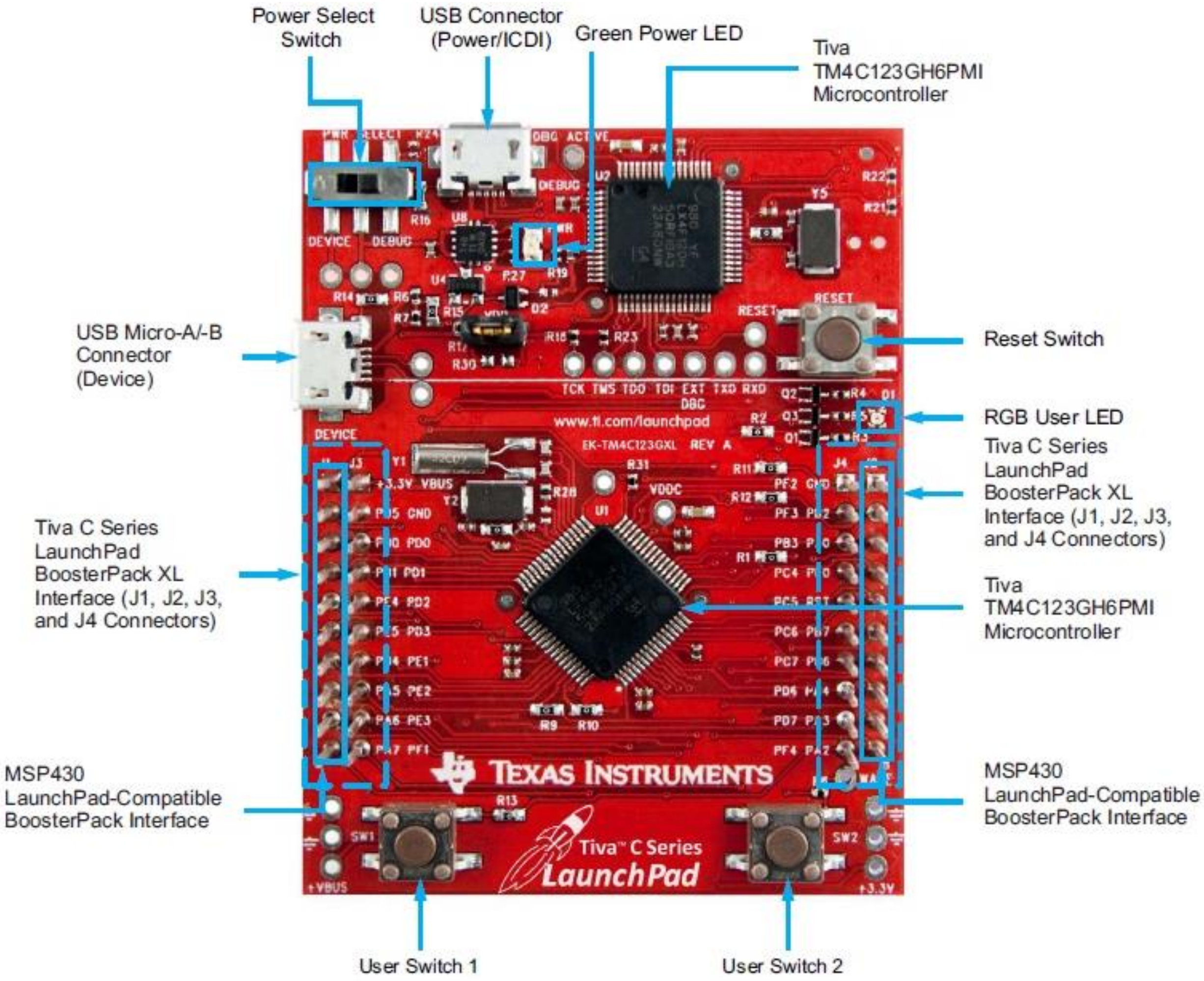
Step 1: installing and checking IDE & SDK

- IDE: Code Composer Studio (CCS)
  - https://www.ti.com.cn/tool/EN/CCSTUDIO
- SDK: Tivaware SW-TM4C
  - https://www.ti.com/tool/SW-TM4C

- *CCS and Tivaware are already installed on the computers in Lab102*



Step 2: getting familiar with the Tiva LaunchPad


Power Select Switch
USB Connector (Power/ICDI)
Green Power LED
Tiva TM4C123GH6PMI Microcontroller
USB Micro-A/-B Connector (Device)
Reset Switch
RGB User LED
Tiva C Series LaunchPad BoosterPack XL Interface (J1, J2, J3, and J4 Connectors)
Tiva C Series LaunchPad BoosterPack XL Interface (J1, J2, J3, and J4 Connectors)
Tiva TM4C123GH6PMI Microcontroller
MSP430 LaunchPad-Compatible BoosterPack Interface
MSP430 LaunchPad-Compatible BoosterPack Interface
User Switch 1
User Switch 2

Tiva C Series TM4C123G LaunchPad Evaluation Board

Step 2: getting familiar with the Tiva LaunchPad

- Tiva-TM4C123GH6PM:
  - Device & Debug processing systems
    - Two identical systems for different use cases
  - Device / Debug activation switch
  - Device & Debug micro-USB ports
  - 3 buttons (1 reset + 2 inputs)
    - The program is stored in the board memory **even when powered-off**. By pressing RESET, the program restarts from its initial stage
  - 1 RGB LED (PF1 + PF2 + PF3)
  - …

Step 3: building and running a HelloWorld program

- Step 3.1: initializing and configuring a CCS project
- Step 3.2: importing the HelloWorld code
- Step 3.3: building and running the program
- Step 3.4: running a debug session of the program

Step 3.1: initializing and configuring a CCS project

1. Run the CCS software

2. Specify the CCS workspace (directory to save project files) (if not done yet)

3. Create a new CCS project
   - File -> New -> CCS Project

4. Configure the project as follows

Step 3.1: initializing and configuring a CCS project

Target:

Tiva TM4C123GH6PM

Connection:

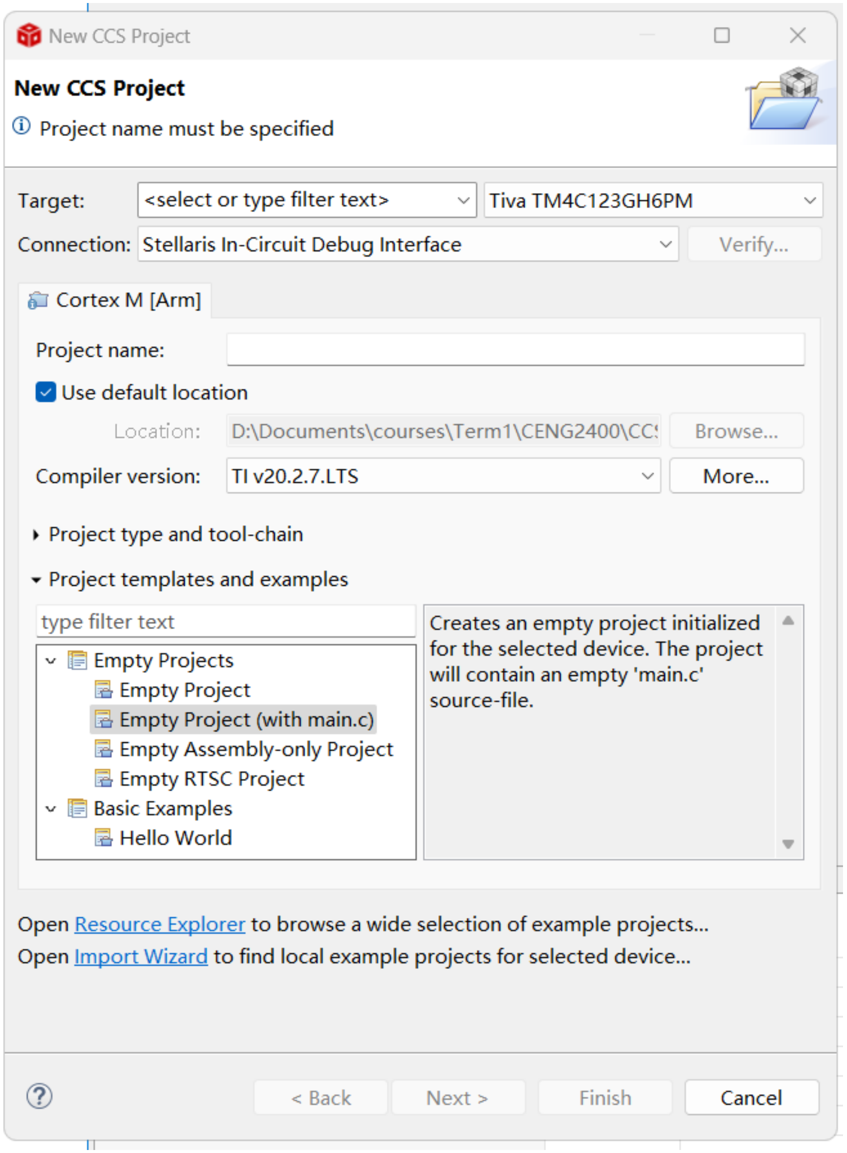
Stellaris In-Circuit Debug Interface

Compiler version:

TI vxxx *(your CCS version)*

Project templates and examples:

Empty Project (with main.c)



New CCS Project

New CCS Project

ⓘ Project name must be specified

Target: <select or type filter text> Tiva TM4C123GH6PM

Connection: Stellaris In-Circuit Debug Interface Verify...

Cortex M [Arm]

Project name:

☑ Use default location

Location: D:\Documents\courses\Term1\CENG2400\CC! Browse...

Compiler version: TI v20.2.7.LTS More...

▸ Project type and tool-chain

▾ Project templates and examples

type filter text

- Empty Projects
  - Empty Project
  - Empty Project (with main.c)
  - Empty Assembly-only Project
  - Empty RTSC Project
- Basic Examples
  - Hello World

Creates an empty project initialized for the selected device. The project will contain an empty 'main.c' source-file.

Open Resource Explorer to browse a wide selection of example projects...

Open Import Wizard to find local example projects for selected device...

Step 3.1: initializing and configuring a CCS project

Configuration: Project Explorer -> right click project -> Properties:

- Build -> ARM Compiler -> Include Options: add the Tivaware installation path
  - E.g., *C:\ti\TivaWare_C_Series-2.1.4.178*
- Build -> ARM Linker -> File Search Path: add the Tivaware **driverlib** path
  - E.g., *C:\ti\TivaWare_C_Series-2.1.4.178\driverlib\ccs\Debug\driverlib.lib*

Step 3.2: importing the HelloWorld code

In file main.c:

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"

uint8_t magic_number=0;

int main(void)
{
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    while(1)
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, magic_number);
        SysCtlDelay(2000000);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00);
        if(magic_number==16) {magic_number=0;} else {magic_number+=2;}
    }
}
```

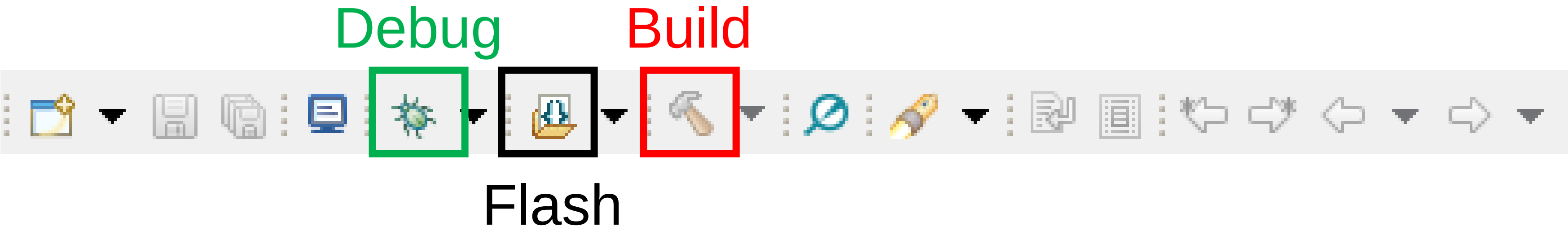
Step 3.3: building and running the program

1. Build the program
2. Connect the board to the computer
3. Run the program

Build: compile the program on the computer

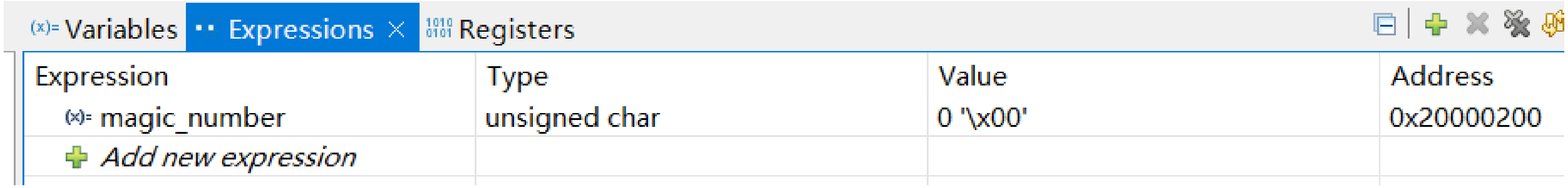
Flash: send the executable files to the board and run

Debug: send the executable files to the board and wait for debug commands



Step 3.4: running a debug session of the program

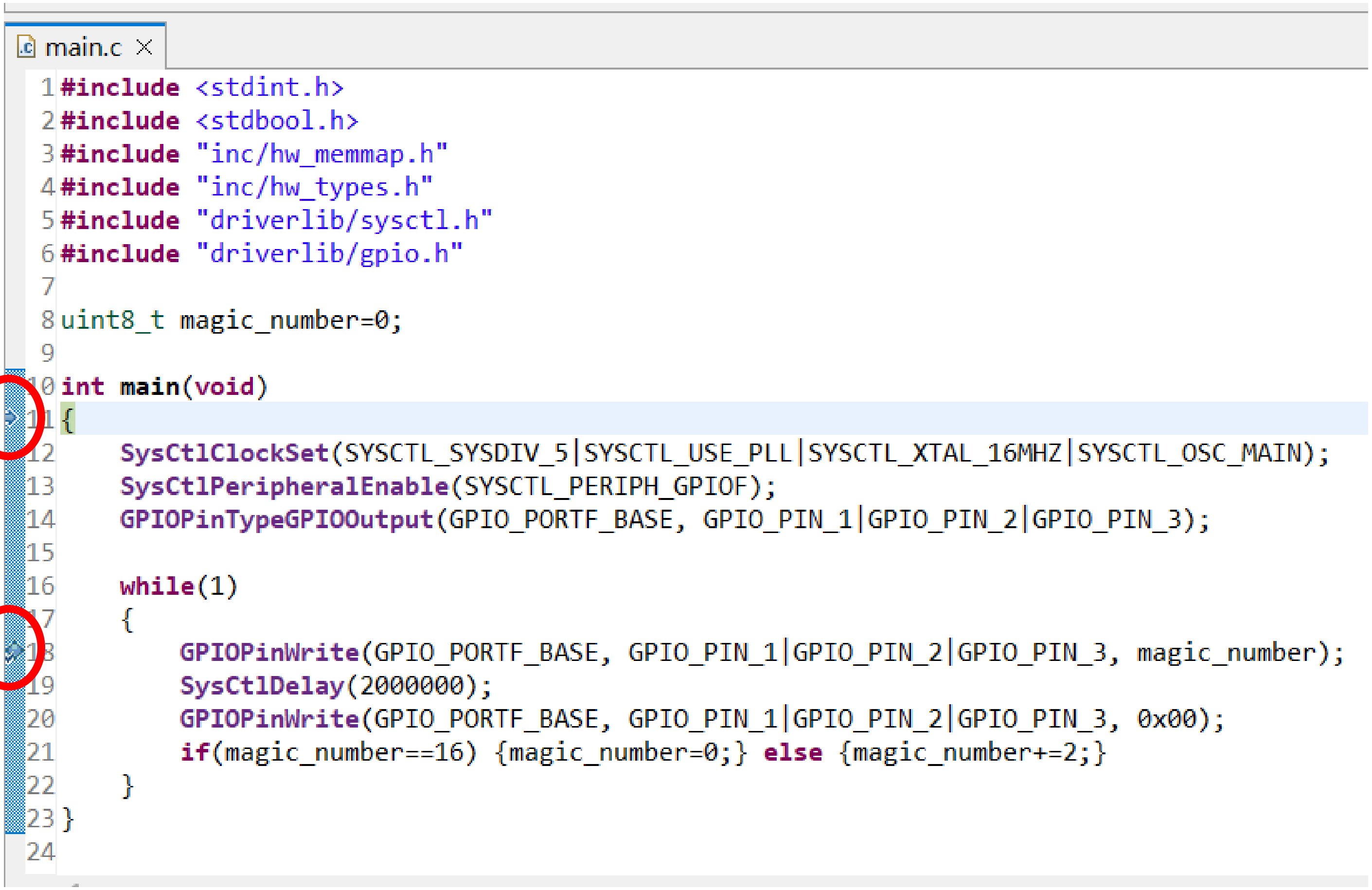
1. Build the program
2. Connect the board to the computer
3. Run “Debug” of the program
4. Add an expression “magic_number”



Expression	Type	Value	Address
magic_number	unsigned char	0 '\x00'	0x20000200
Add new expression			

Step 3.4: running a debug session of the program

5. Double click line18 at the blue zone to add a breakpoint



```
 1 #include <stdint.h>
 2 #include <stdbool.h>
 3 #include "inc/hw_memmap.h"
 4 #include "inc/hw_types.h"
 5 #include "driverlib/sysctl.h"
 6 #include "driverlib/gpio.h"
 7
 8 uint8_t magic_number=0;
 9
10 int main(void)
11 {
12     SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
13     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
14     GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
15
16     while(1)
17     {
18         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, magic_number);
19         SysCtlDelay(2000000);
20         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00);
21         if(magic_number==16) {magic_number=0;} else {magic_number+=2;}
22     }
23 }
24
```

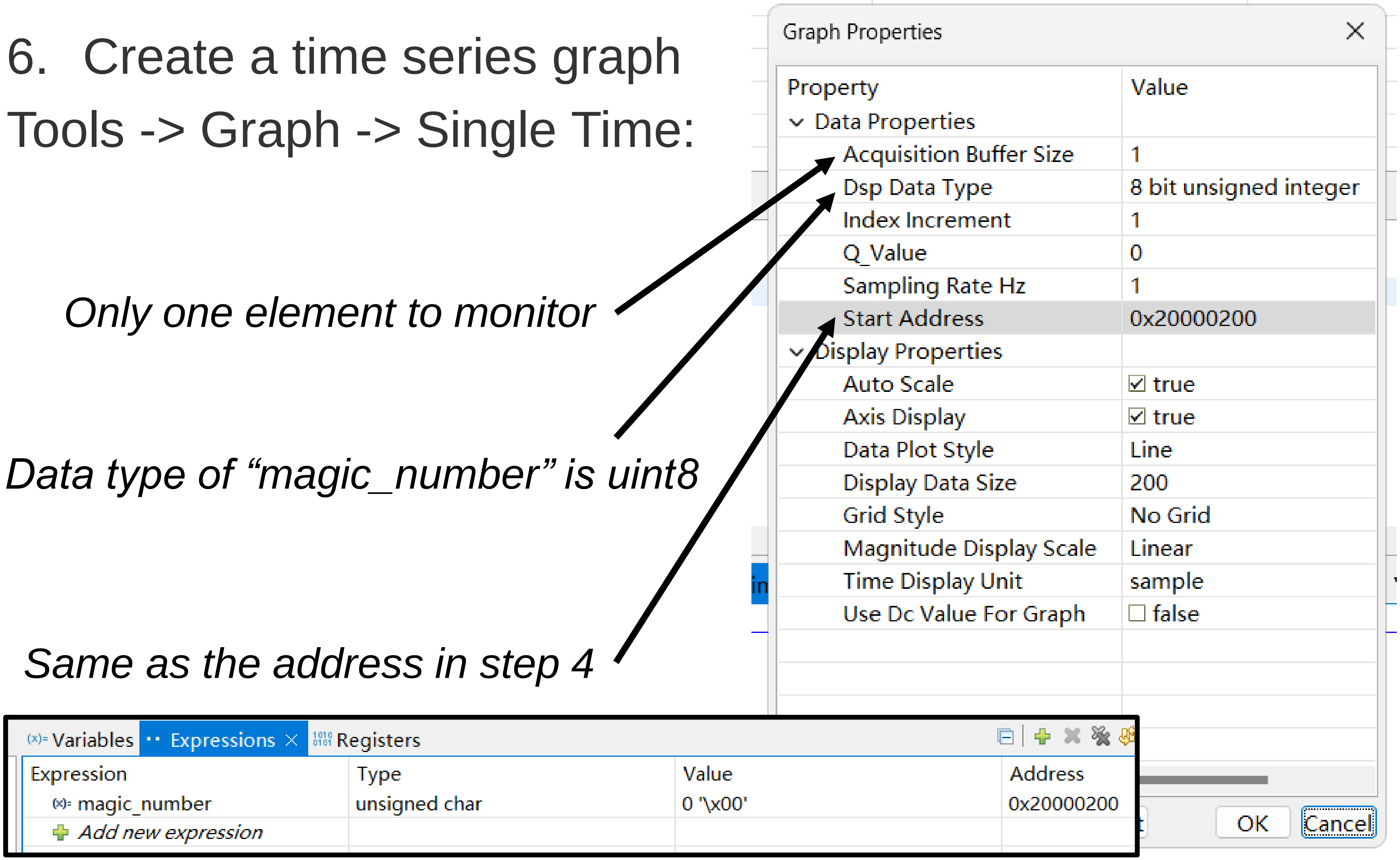
Current location

Breakpoint location

Step 3.4: running a debug session of the program

6. Create a time series graph

Tools -> Graph -> Single Time:



Only one element to monitor

Data type of "magic_number" is uint8

Same as the address in step 4

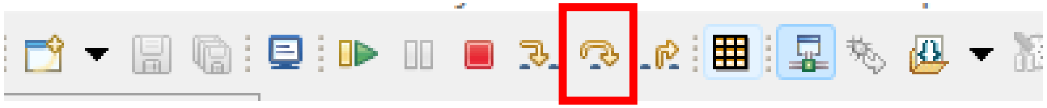
Graph Properties

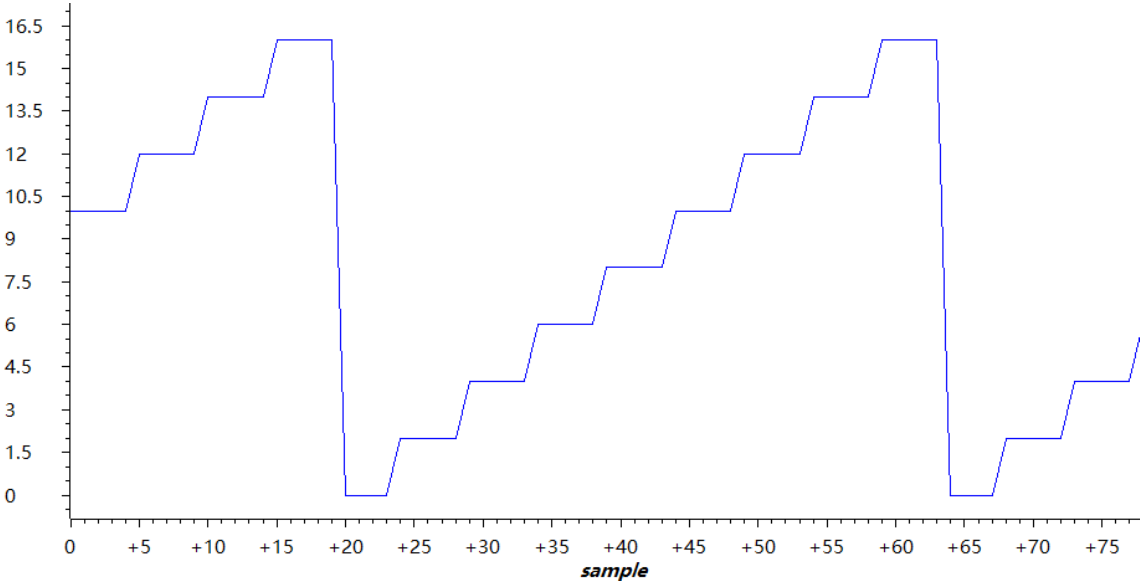
Property	Value
⌄ Data Properties	
Acquisition Buffer Size	1
Dsp Data Type	8 bit unsigned integer
Index Increment	1
Q_Value	0
Sampling Rate Hz	1
Start Address	0x20000200
⌄ Display Properties	
Auto Scale	☑ true
Axis Display	☑ true
Data Plot Style	Line
Display Data Size	200
Grid Style	No Grid
Magnitude Display Scale	Linear
Time Display Unit	sample
Use Dc Value For Graph	□ false

Expression	Type	Value	Address
magic_number	unsigned char	0 '\x00'	0x20000200
Add new expression			

Step 3.4: running a debug session of the program

7. Run “Step Over” and check the results
   - Don’t press too fast!


Step Over



Summary

1. Install and check IDE & SDK
2. Create a new project with the given code
3. Run and debug the program
4. **Submit a video of your results to Blackboard before the next lab (next Tuesday)**

- Switch off your computer before leaving
- Take all your belongings before leaving – make it clean

Summary

- Do **just the same** as shown in the slides
- Carefully check all the settings of the program
  - Project settings
  - Debug settings
  - …

Q&A

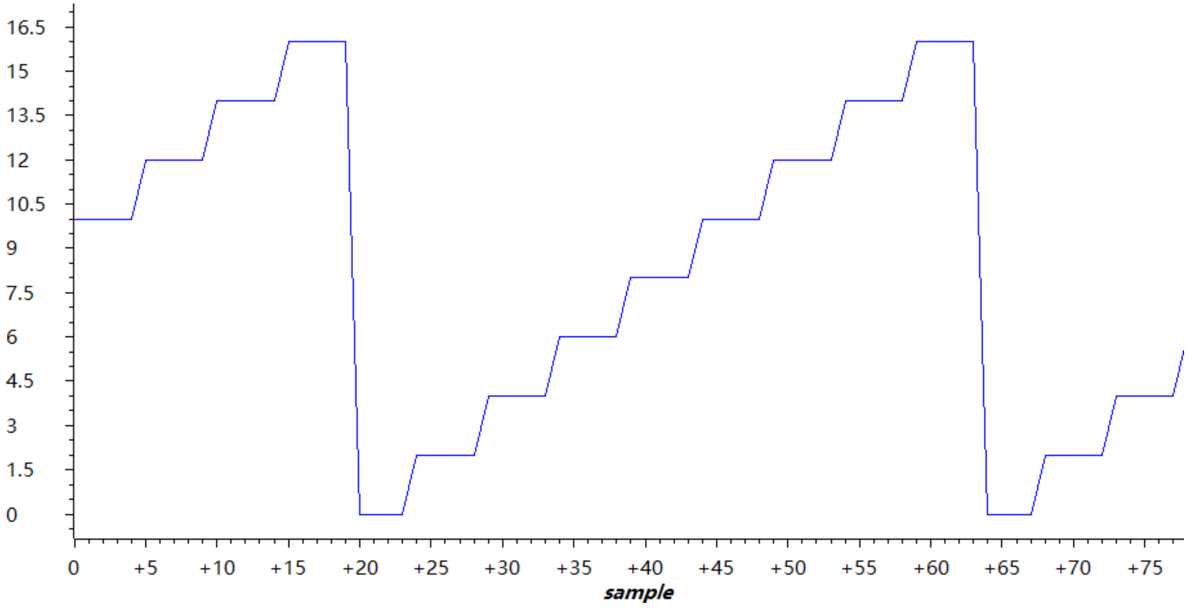
Thanks for listening!

Q & A

Explanations

Why is the graph like that?

- “magic_number” increases by 2 in each loop, and is reset after reaching 16



Explanations

Why is the LED blinking like that?

- The Tiva board has multiple groups of ports, named bases, such as port A base, port B base, …
- Each group of ports have multiple pins, named GPIO pins, such as GPIO pin 1, GPIO pin 2, …
- For example, pin number 1 in group F is named GPIO_PIN_1 in GPIO_PORTF_BASE (PF1)

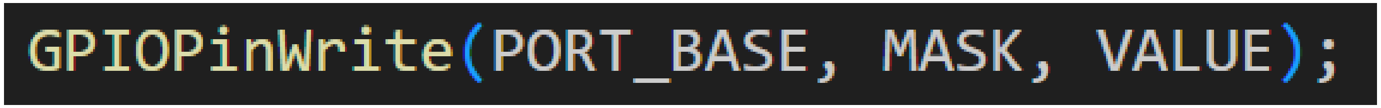
Explanations

Why is the LED blinking like that?

- Each pin can be set as input or output. In this lab, we set the pins of the LED as output
- The R, G and B of the LED correspond to PF1, PF3 and PF2
- If PF1 is set high, then red is on; if PF3 is set high, then green is on; if both PF1 and PF3 are on, then yellow (red + green) is on

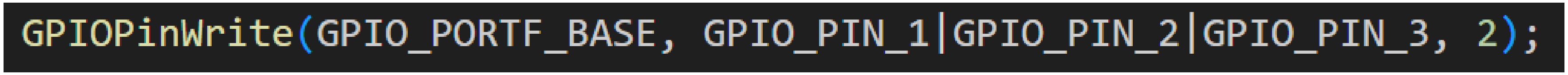
Explanations

Why is the LED blinking like that?



```
GPIOPinWrite(PORT_BASE, MASK, VALUE);
```

- To **set the binary VALUE to PORT_BASE, and the setting only affects MASK**



```
GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 2);
```

- Port base F has 8 pins. This command sets value **00000010** to the base, and it only affects pin 1-3
- This means the values of pin 1-3 are 100 (0000**[001]**0)

Explanations

Why is the LED blinking like that?

					G	B	R		
Value	**PF7**	**PF6**	**PF5**	**PF4**	**PF3**	**PF2**	**PF1**	**PF0**	**LED**
0	0	0	0	0	0	0	0	0	Off
2	0	0	0	0	0	0	1	0	Red
4	0	0	0	0	0	1	0	0	Blue
6	0	0	0	0	0	1	1	0	Purple (Red + Blue)
8	0	0	0	0	1	0	0	0	Green
10	0	0	0	0	1	0	1	0	Yellow (Red + Green)
12	0	0	0	0	1	1	0	0	Cyan (Green + Blue)
14	0	0	0	0	1	1	1	0	White (Red + Green + Blue)
16	0	0	0	1	0	0	0	0	Off