BASIC GUIDE TO PROGRAMMING NUVOTON NUC140

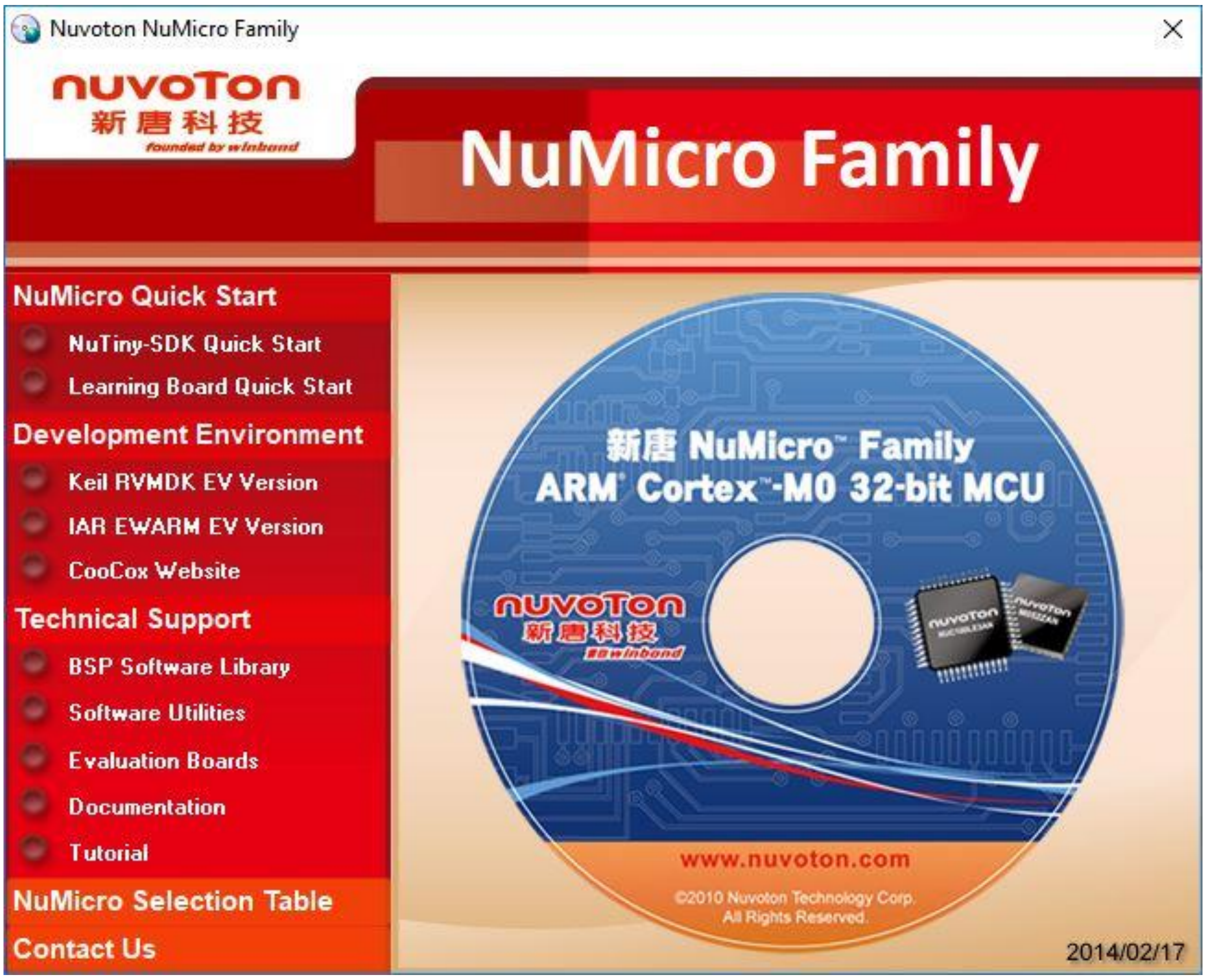
Prepared by: Ahmad Shahril Bin Mohd Ghani

Date: 17th February 2018

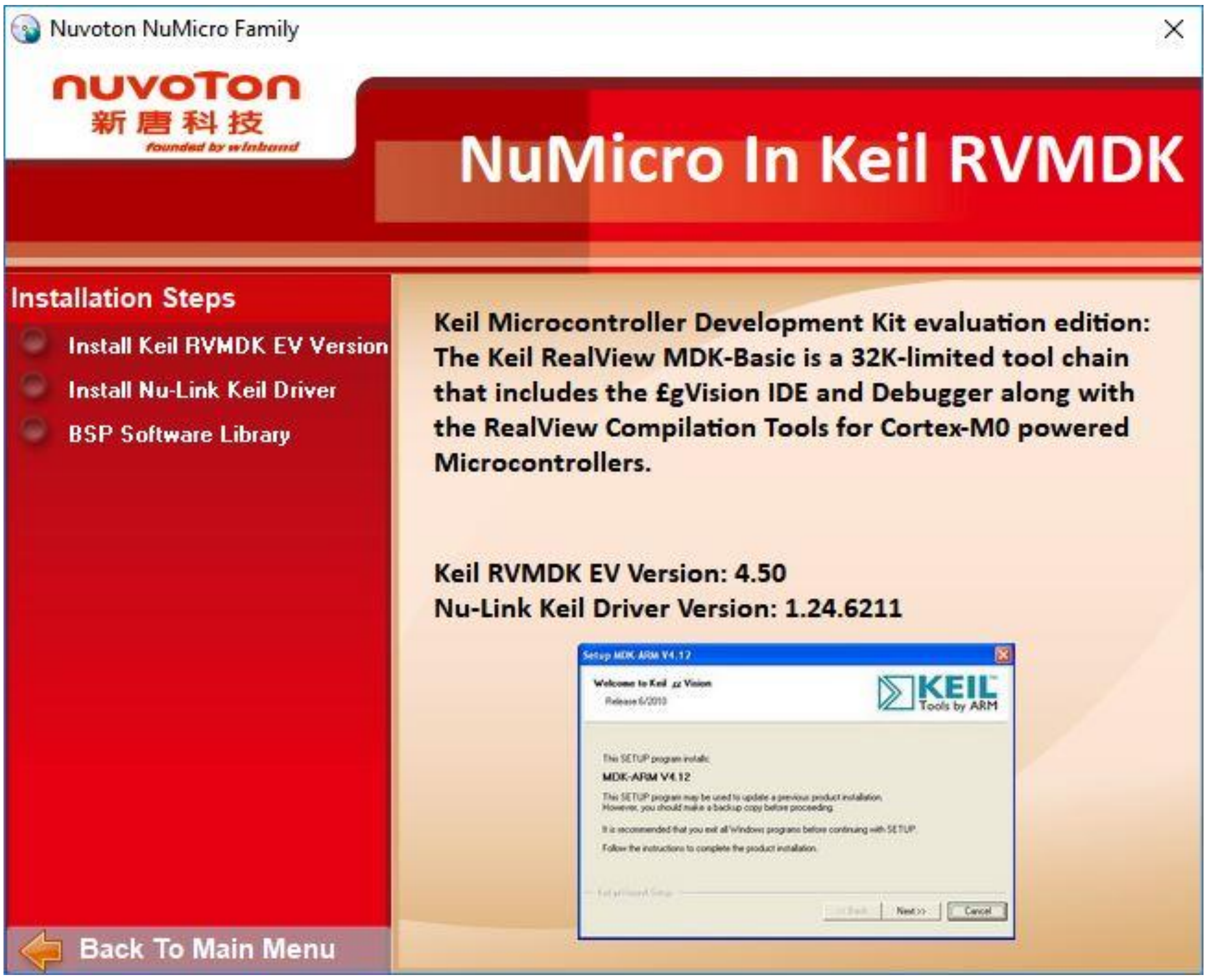
TABLE OF CONTENTS

INSTALLATION

1. Run the installer that can be found in the CD provided.



2. Under *Development Environment*, select **Keil RVMDK EV Version**. Install both **Keil RVMDK EV Version** and **Nu-Link Keil Driver**.



Keil Microcontroller Development Kit evaluation edition: The Keil RealView MDK-Basic is a 32K-limited tool chain that includes the £gVision IDE and Debugger along with the RealView Compilation Tools for Cortex-M0 powered Microcontrollers.

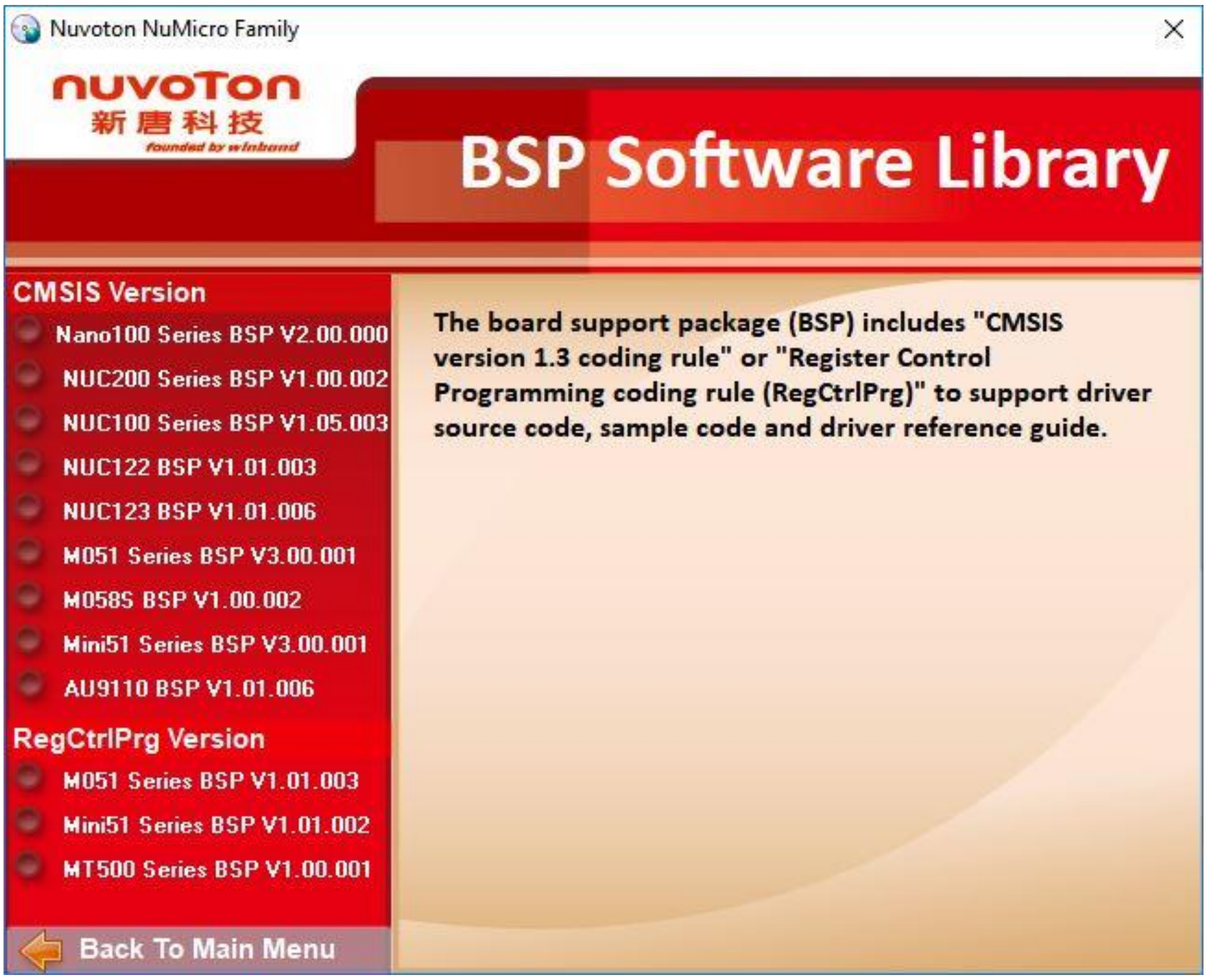
Keil RVMDK EV Version: 4.50
Nu-Link Keil Driver Version: 1.24.6211

Setup MDK-ARM V4.12

Welcome to Keil µVision
Release 6/2010

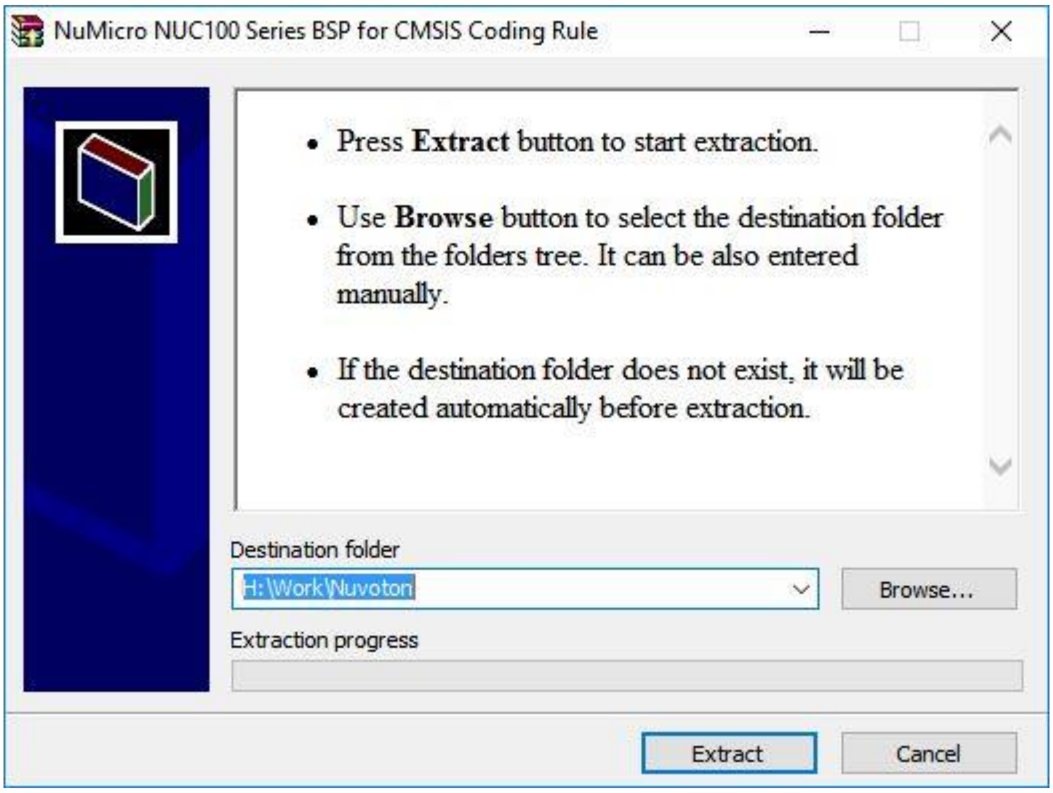
This SETUP program installs:
MDK-ARM V4.12
This SETUP program may be used to update a previous product installation.
However, you should make a backup copy before proceeding.
It is recommended that you exit all Windows programs before continuing with SETUP.
Follow the instructions to complete the product installation.

3. Click **BSP Software Library**. Select **NUC100 Series BSP V1.05.003**.


Nuvoton NuMicro Family
nuvoTon
新唐科技
founded by winbond
BSP Software Library
CMSIS Version
Nano100 Series BSP V2.00.000
NUC200 Series BSP V1.00.002
NUC100 Series BSP V1.05.003
NUC122 BSP V1.01.003
NUC123 BSP V1.01.006
M051 Series BSP V3.00.001
M058S BSP V1.00.002
Mini51 Series BSP V3.00.001
AU9110 BSP V1.01.006
RegCtrlPrg Version
M051 Series BSP V1.01.003
Mini51 Series BSP V1.01.002
MT500 Series BSP V1.00.001
Back To Main Menu

The board support package (BSP) includes "CMSIS version 1.3 coding rule" or "Register Control Programming coding rule (RegCtrlPrg)" to support driver source code, sample code and driver reference guide.

4. Choose the location to extract the libraries. **Remember where you've extracted the file**. We will need to refer to this folder afterward.

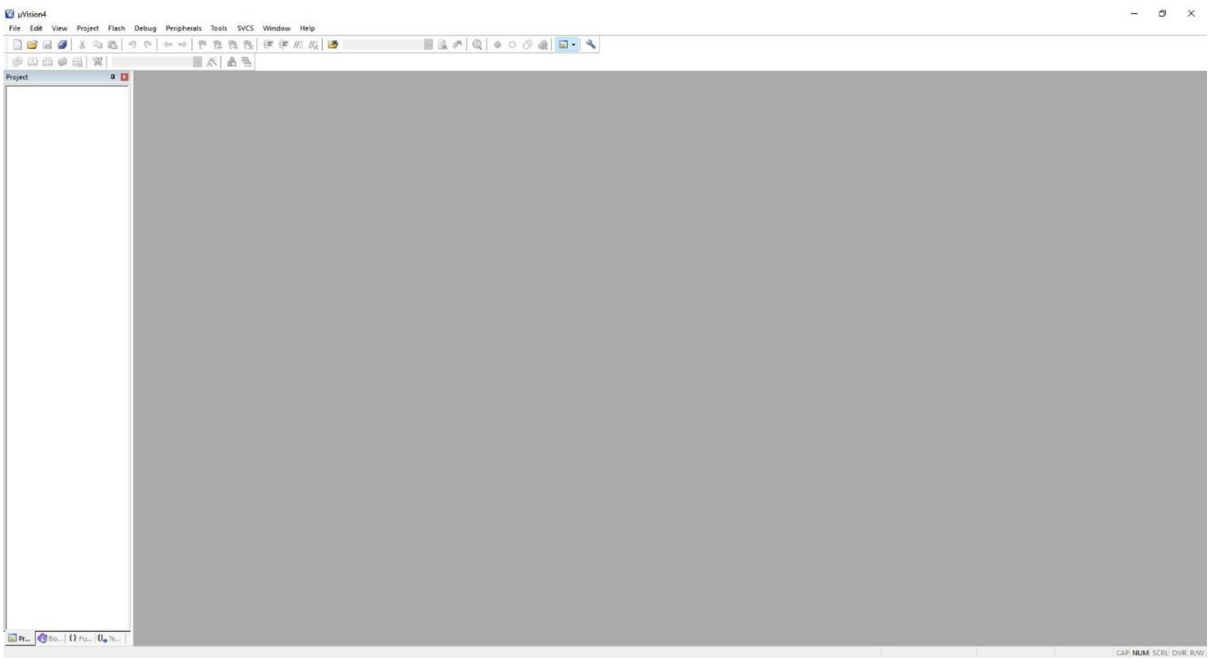

NuMicro NUC100 Series BSP for CMSIS Coding Rule

- Press **Extract** button to start extraction.
- Use **Browse** button to select the destination folder from the folders tree. It can be also entered manually.
- If the destination folder does not exist, it will be created automatically before extraction.

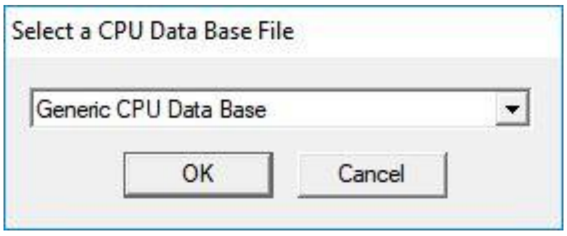
Destination folder: H:\Work\Nuvoton

CREATING EMPTY PROJECT

1. Launch Keil software.

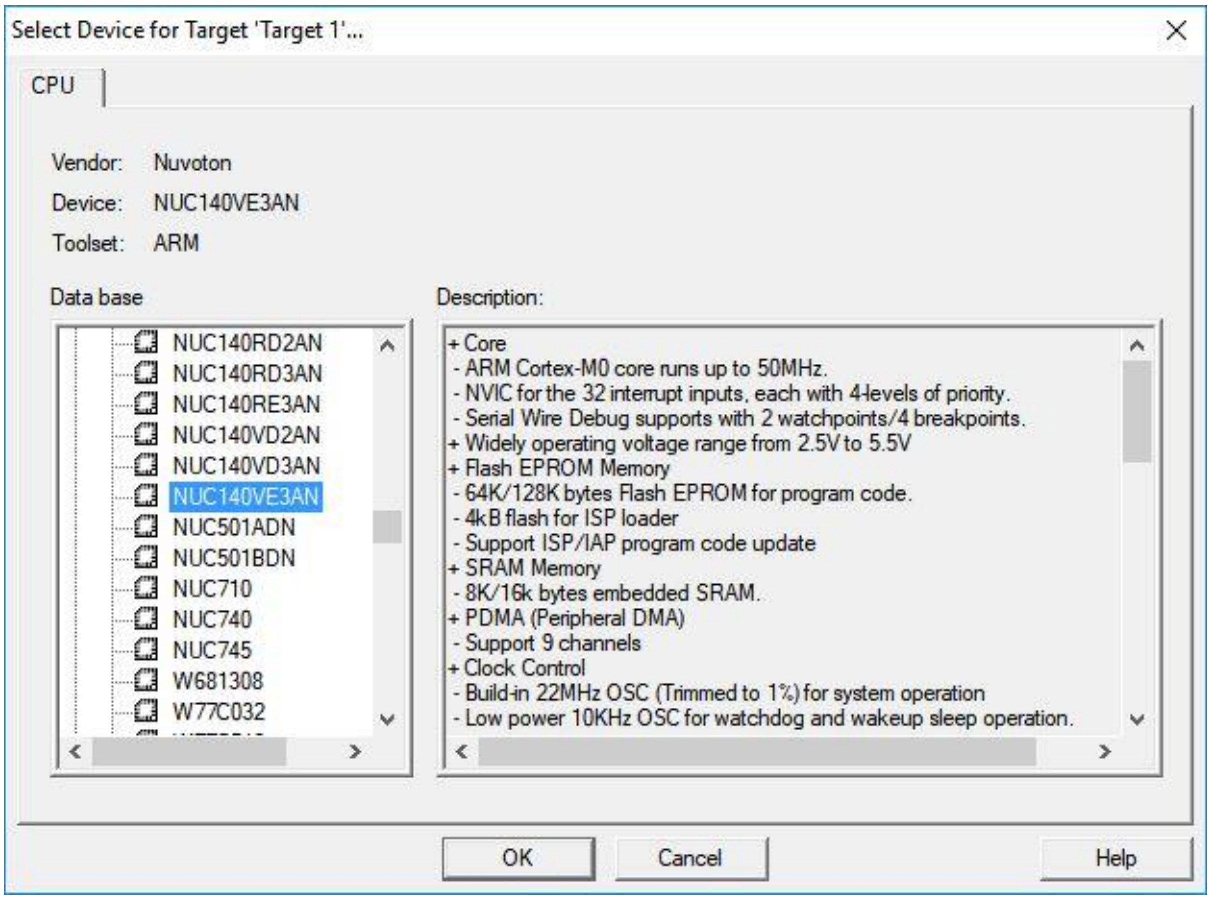

µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project

2. Select *Project > New µVision Project*
3. Save the project in a new folder. This is to avoid clutter as more files will be generated later.
4. Select **Generic CPU Data Base** when the following window appear.


Select a CPU Data Base File

Generic CPU Data Base

5. Find **Nuvoton** vendor and expand the tree. Find the device **NUC140VE3AN** and press OK.



Select Device for Target 'Target 1'...

CPU

Vendor: Nuvoton
Device: NUC140VE3AN
Toolset: ARM

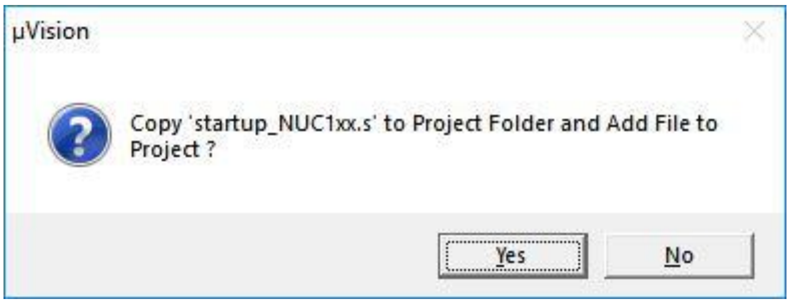
Data base

NUC140RD2AN
NUC140RD3AN
NUC140RE3AN
NUC140VD2AN
NUC140VD3AN
NUC140VE3AN
NUC501ADN
NUC501BDN
NUC710
NUC740
NUC745
W681308
W77C032

Description:

```
+ Core
 - ARM Cortex-M0 core runs up to 50MHz.
 - NVIC for the 32 interrupt inputs, each with 4-levels of priority.
 - Serial Wire Debug supports with 2 watchpoints/4 breakpoints.
+ Widely operating voltage range from 2.5V to 5.5V
+ Flash EPROM Memory
 - 64K/128K bytes Flash EPROM for program code.
 - 4kB flash for ISP loader
 - Support ISP/IAP program code update
+ SRAM Memory
 - 8K/16k bytes embedded SRAM.
+ PDMA (Peripheral DMA)
 - Support 9 channels
+ Clock Control
 - Build-in 22MHz OSC (Trimmed to 1%) for system operation
 - Low power 10KHz OSC for watchdog and wakeup sleep operation.
```

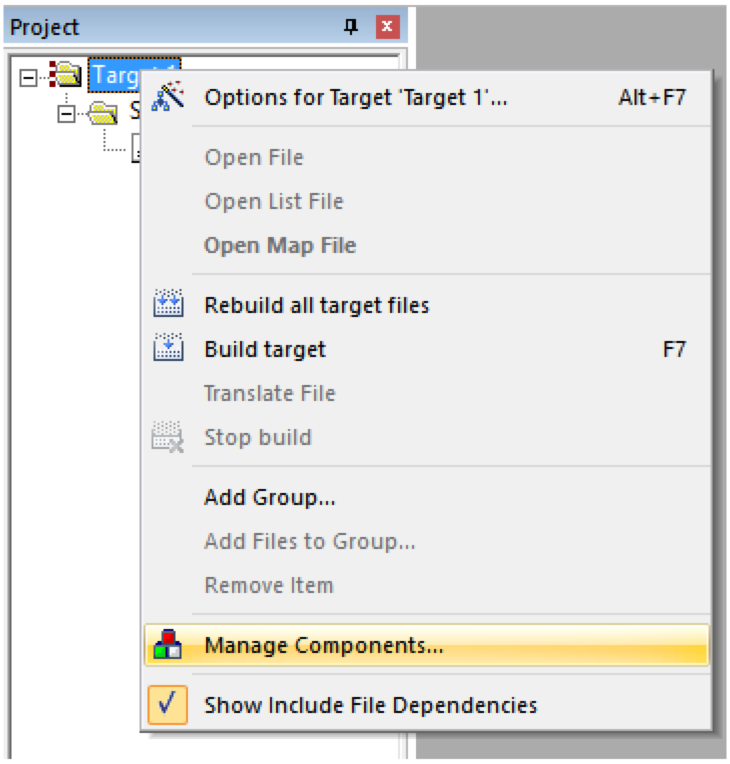
6. When the following window appear, select **Yes**.



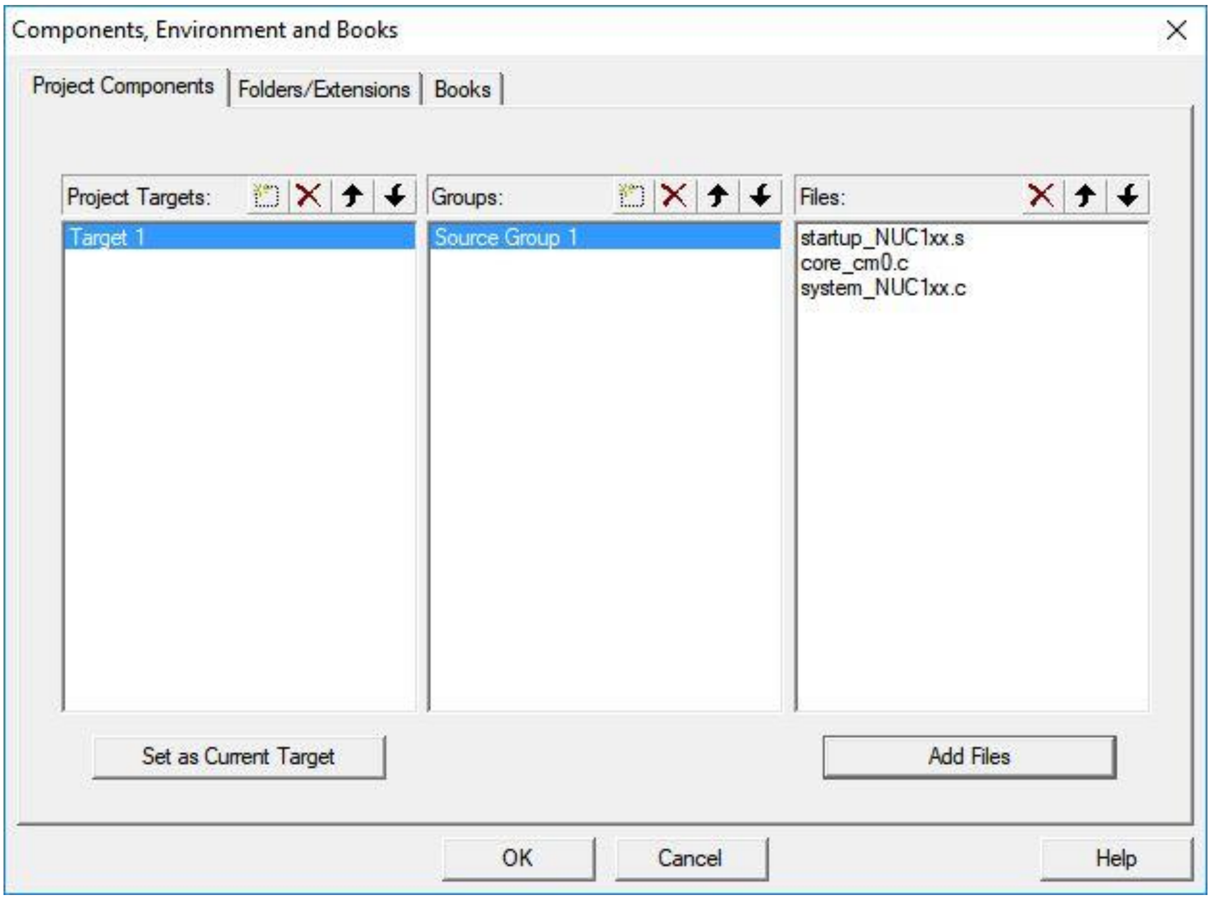
µVision

Copy 'startup_NUC1xx.s' to Project Folder and Add File to Project ?

7. In the Project tab, right-click on the target name and select *Manage Components…*



8. Select *Add Files* and browse to the BSP Software Library folder that you have extracted earlier. You should add 2 files in this step. The relative locations of the files are:
   a. <EXTRACTED LIBRARY DIRECTORY> \CMSIS\CM0\CoreSupport\core_cm0.c
   b. <EXTRACTED LIBRARY DIRECTORY> \CMSIS\CM0\DeviceSupport\Nuvoton\NUC1xx\system_NUC1xx.c
9. Click **OK** once you have finished.

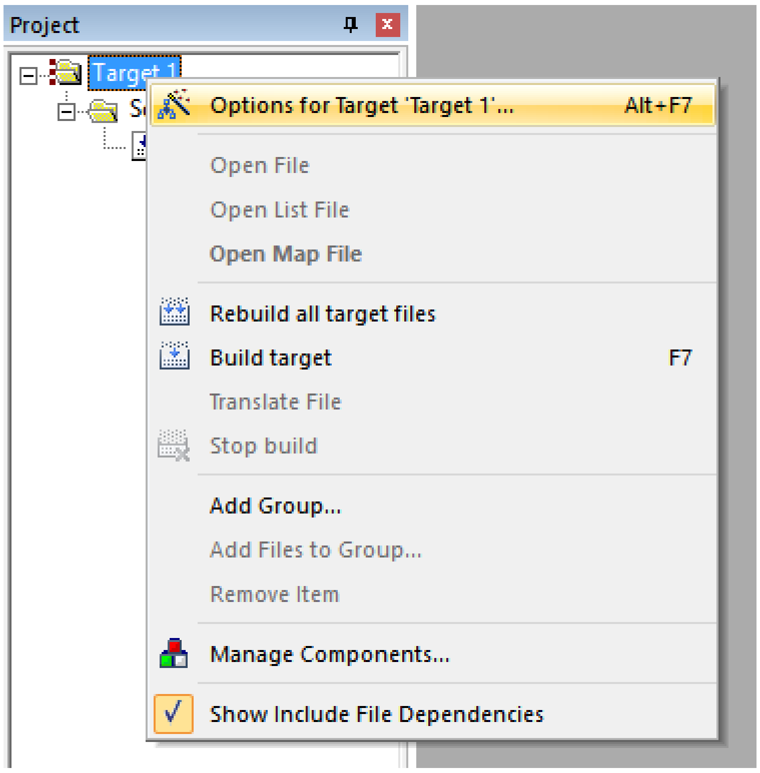


Target 1

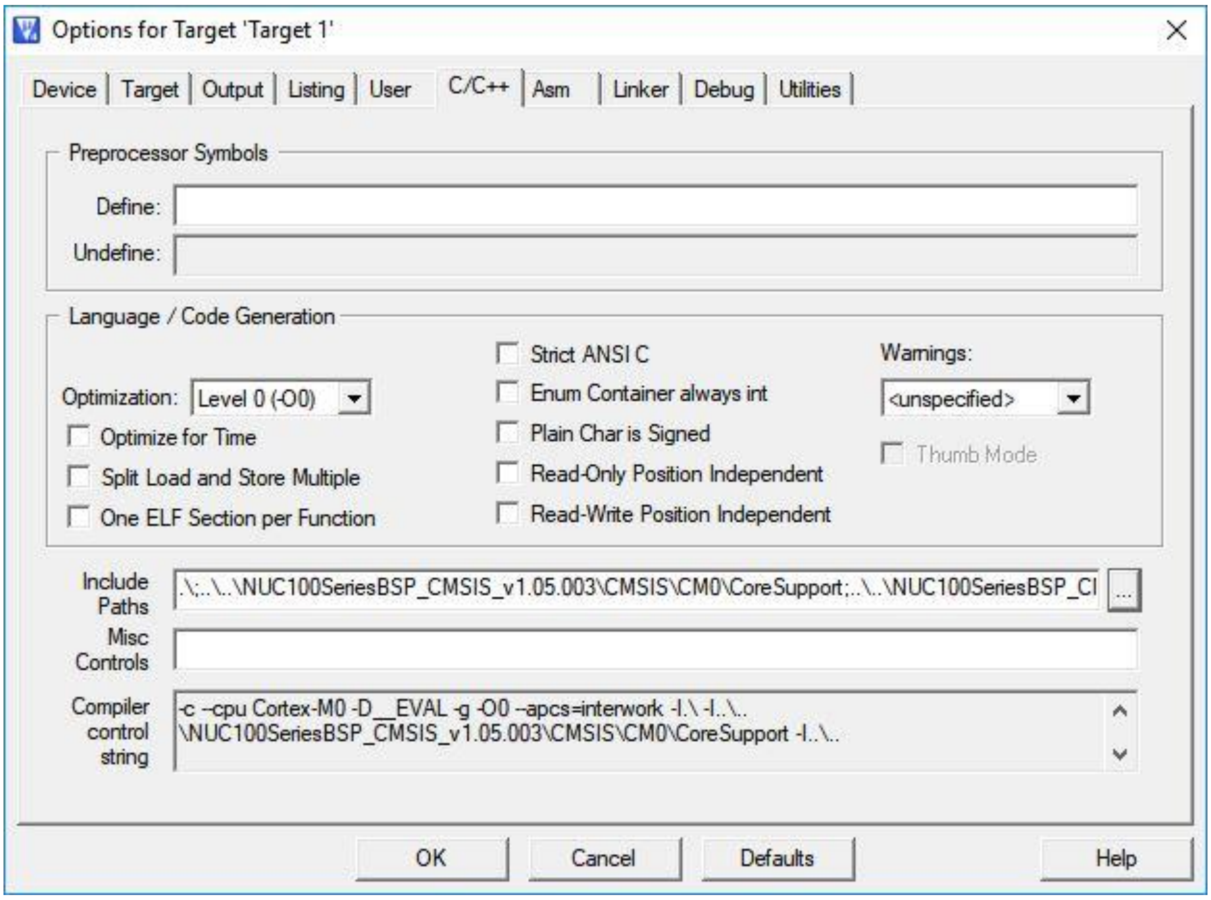
Source Group 1

startup_NUC1xx.s
core_cm0.c
system_NUC1xx.c

10. In the Project tab, right-click on the target name and select *Options for Target*.



11. In C/C++ tab, press "**…**" button next to the *Include Paths* field.



Options for Target 'Target 1'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

Preprocessor Symbols
Define:
Undefine:

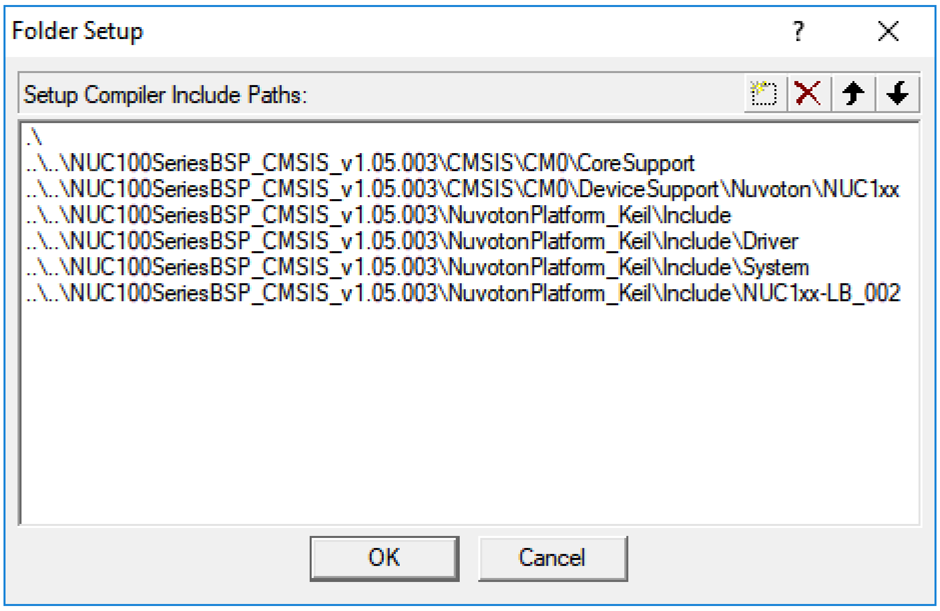
Language / Code Generation
Optimization: Level 0 (-O0)
☐ Optimize for Time
☐ Split Load and Store Multiple
☐ One ELF Section per Function
☐ Strict ANSI C
☐ Enum Container always int
☐ Plain Char is Signed
☐ Read-Only Position Independent
☐ Read-Write Position Independent
Warnings: <unspecified>
☐ Thumb Mode

Include Paths: .\;..\..\NUC100SeriesBSP_CMSIS_v1.05.003\CMSIS\CM0\CoreSupport;..\..\NUC100SeriesBSP_CI
Misc Controls:
Compiler control string:

```
-c --cpu Cortex-M0 -D__EVAL -g -O0 --apcs=interwork -I.\ -I..\..
\NUC100SeriesBSP_CMSIS_v1.05.003\CMSIS\CM0\CoreSupport -I..\..
```

OK | Cancel | Defaults | Help

12. Include the following paths in order and press **OK**:



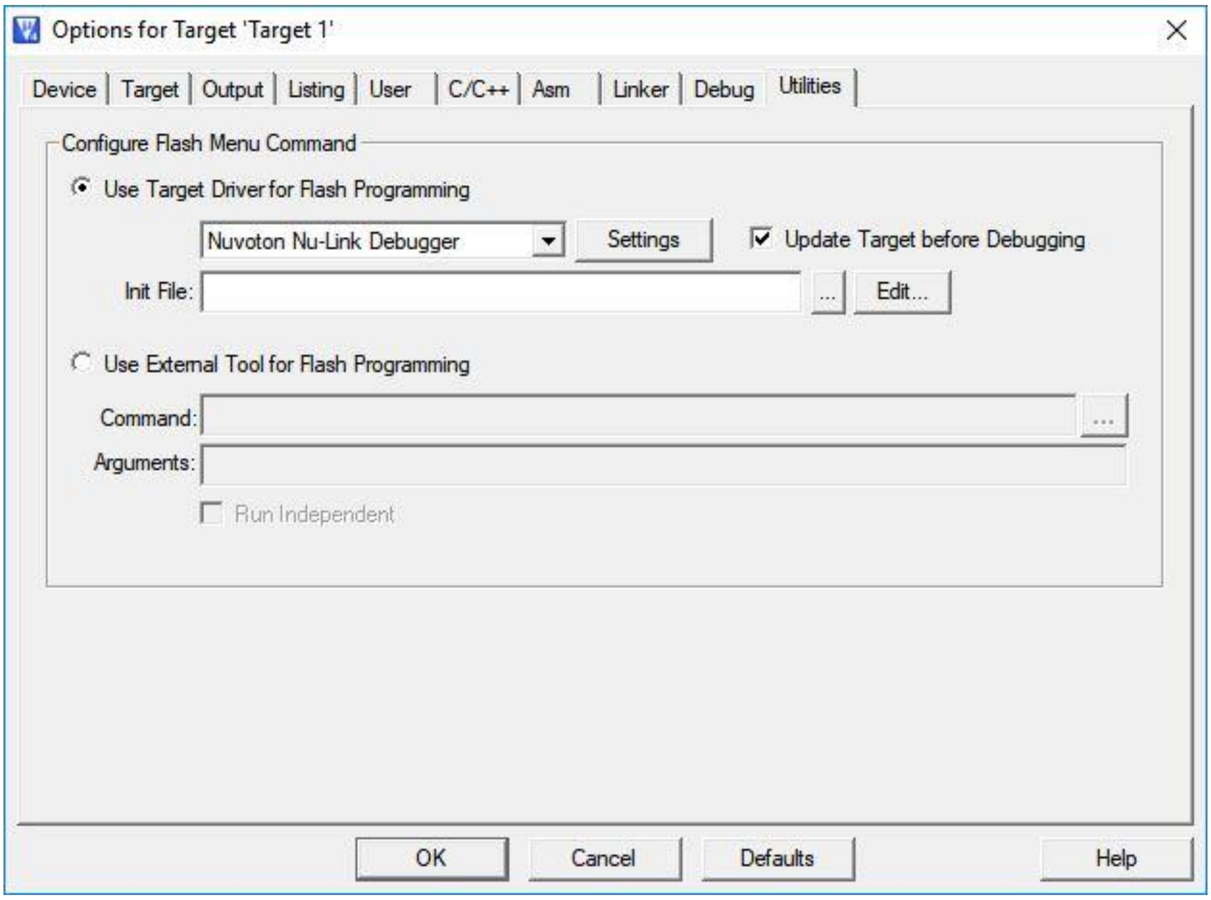
Folder Setup

Setup Compiler Include Paths:

```
.\
..\..\NUC100SeriesBSP_CMSIS_v1.05.003\CMSIS\CM0\CoreSupport
..\..\NUC100SeriesBSP_CMSIS_v1.05.003\CMSIS\CM0\DeviceSupport\Nuvoton\NUC1xx
..\..\NUC100SeriesBSP_CMSIS_v1.05.003\NuvotonPlatform_Keil\Include
..\..\NUC100SeriesBSP_CMSIS_v1.05.003\NuvotonPlatform_Keil\Include\Driver
..\..\NUC100SeriesBSP_CMSIS_v1.05.003\NuvotonPlatform_Keil\Include\System
..\..\NUC100SeriesBSP_CMSIS_v1.05.003\NuvotonPlatform_Keil\Include\NUC1xx-LB_002
```

OK | Cancel

13. Next, click *Utilities* tab. Change the *Use Target Driver for Flash Programming* option to **Nuvoton Nu-Link Debugger**. Click **OK** when you are done.



(●) Use Target Driver for Flash Programming

Nuvoton Nu-Link Debugger

[✓] Update Target before Debugging

Init File:

( ) Use External Tool for Flash Programming

Command:

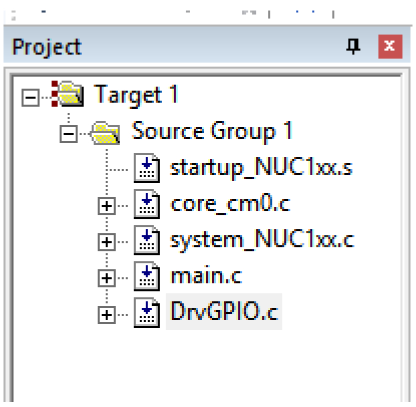
Arguments:

[ ] Run Independent

14. Finally, we are going to create a new file where we are going to write the main programming. Go to *File > New.* Save the file as '**main.c**' in the project folder. We need to include the file in our project. Right click on the *Source Group 1* title. Select ***Add Files to Group 'Source Group 1'…*** Select the newly created main.c file and press **ADD**.
15. For the first example program, we will create LED blinking program. For that, we need to add GPIO library. Like the previous step, browse to the following folder and add the file:

```
NUC100SeriesBSP_CMSIS_v1.05.003\NuvotonPlatform_Keil\Src\Driver\DrvGPIO.c
```

16. Your project should look like the following:

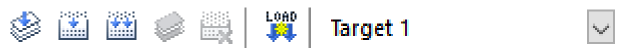


```
Target 1
  Source Group 1
    startup_NUC1xx.s
    core_cm0.c
    system_NUC1xx.c
    main.c
    DrvGPIO.c
```

17. Copy the following code into main.c file:

```
#include <stdio.h>
#include "NUC1xx.h"
#include "DrvGPIO.h"

void delay()
{
      unsigned long i;

      for(i=0; i<1000000; i++) {
            __NOP();
      }
}

int main (void)
{
      DrvGPIO_Open(E_GPA, 13, E_IO_OUTPUT);

      while(1)
      {
            DrvGPIO_ClrBit(E_GPA,13);
            delay();

            DrvGPIO_SetBit(E_GPA,13);
            delay();
      }
}
```

16. Press **Rebuild** and **Download** the program to your board. Press **RESET** button on the board to start the program.



CODING EXAMPLES

DIGITAL OUTPUT

In the previous section, we have tried a simple LED blink program. We will dissect the code to further understand what it really meant.

HEADER FILES INCLUDE SECTION

```
#include <stdio.h>
#include "NUC1xx.h"
#include "DrvGPIO.h"
```

The lines above are quite self-explanatory. We simply include the libraries that we are going to use, which in this case is the GPIO pins. stdio and NUC1xx libraries are compulsory.

DELAY LOOP

```
for(i=0; i<1000000; i++) {
      __NOP();
}
```

There is basically no reliable delay function for ARM processor as in Arduino IDE. The library did provide a delay function, but it only goes up to 335000 microseconds. The closest that we can do is by using __NOP() function. This function is for executing "no operation" in a cycle which takes about 1 microsecond. To achieve a delay, we need to call this function multiple time. Calling the function for 1000000 times causes delay of about 1 second.

PIN DEFINITION

```
DrvGPIO_Open(E_GPA, 13, E_IO_OUTPUT);
```

The function call above sets GPA13 pin as OUTPUT. To set GPC14 pin as output, we can call:

```
DrvGPIO_Open(E_GPC, 14, E_IO_OUTPUT);
```

DIGITAL SIGNAL WRITE

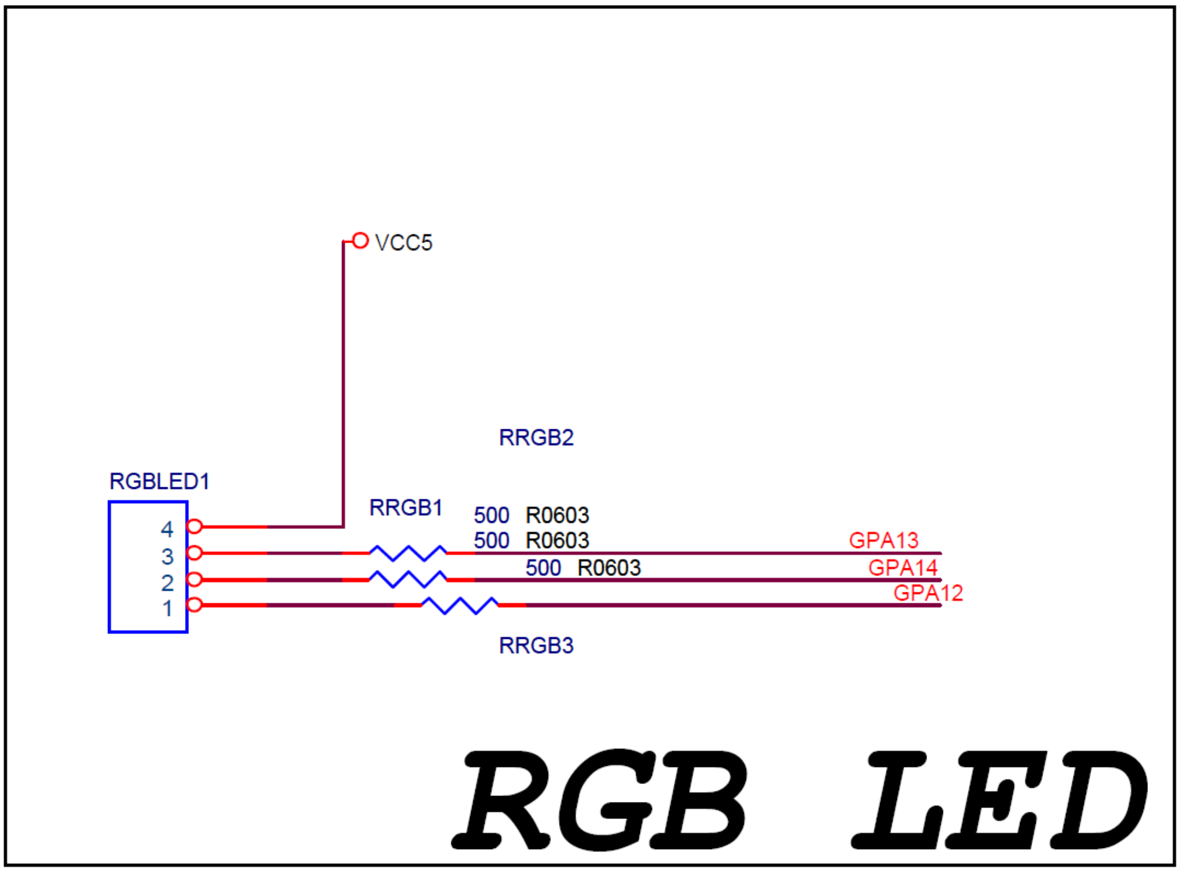
To set LOW signal to GPA13 pin, we need to call the following function:

```
DrvGPIO_ClrBit(E_GPA, 13);
```

To set HIGH signal to GPA13 pin, we need to call the following function:

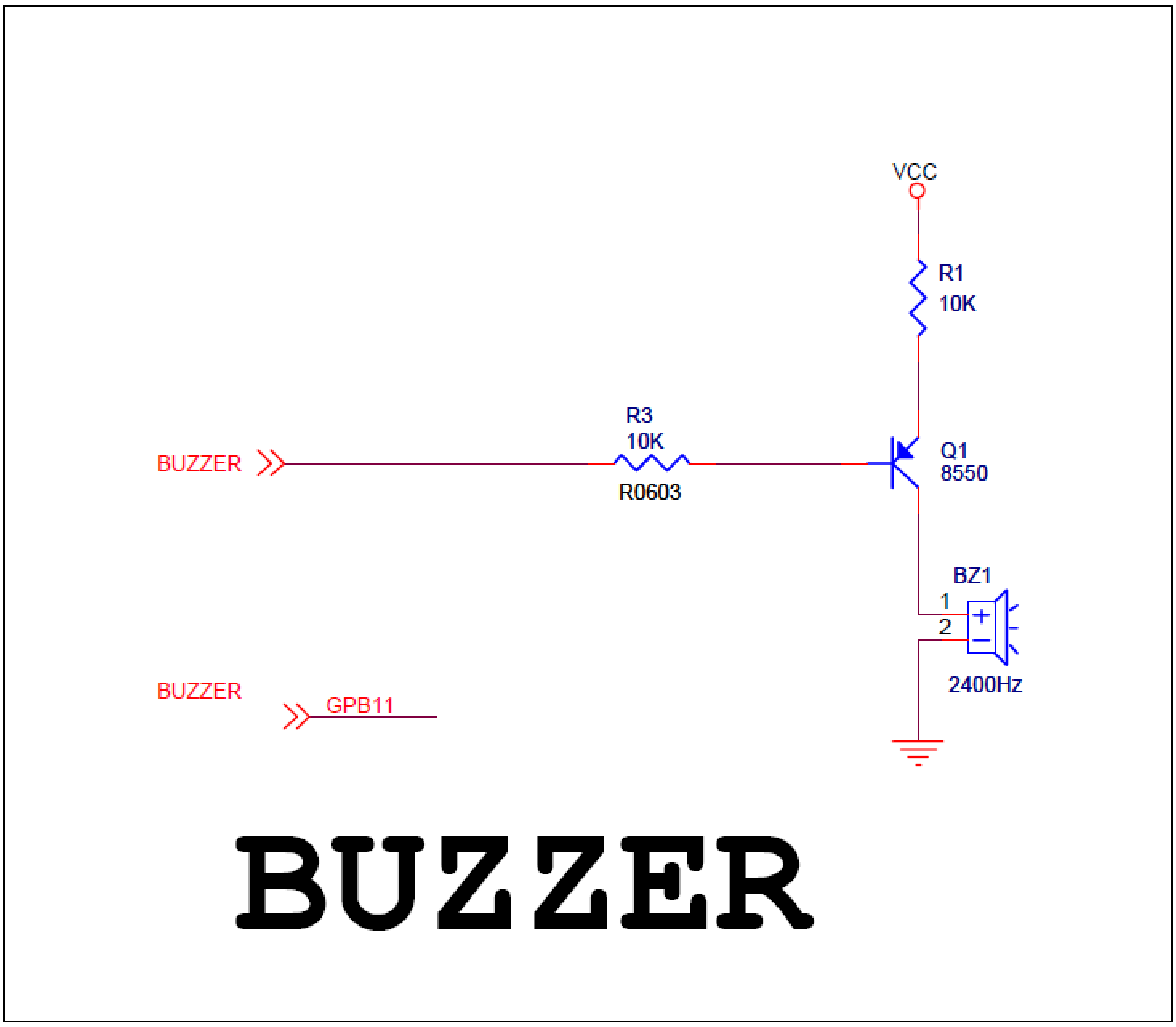
```
DrvGPIO_SetBit(E_GPA, 13);
```

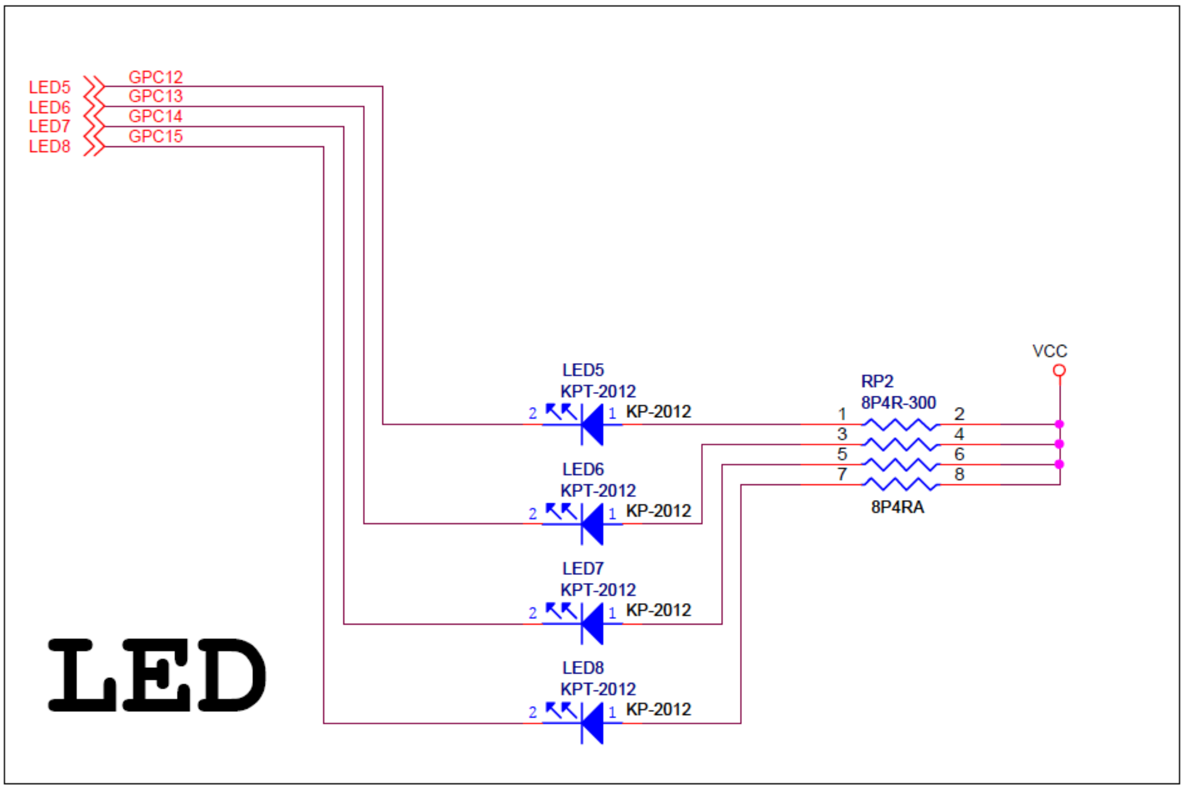
If you have run the example code, you should see green LED blinking at the bottom left of the LCD. This is because GPA13 is connected to the RGB LED. You can see the schematic below:



As you can see from the schematic, the RGB LED common pin is connected to VCC5, which means that this LED is a common anode type. To turn the LED on, we must give LOW signal to GPA13. To turn it off, we must give HIGH signal to GPA13. Try turning GPA14 and GPA12 LOW to turn on red and blue LED. Play around with it and mix some colours.

You may also try experimenting with buzzer, 4 other LEDs, and 7-segment displays attached on the board as well using this example code. Similarly, the buzzer and LEDs are active low.


VCC
R1
10K
R3
10K
R0603
BUZZER
Q1
8550
BZ1
1
2
2400Hz
BUZZER
GPB11
BUZZER


LED5
LED6
LED7
LED8
GPC12
GPC13
GPC14
GPC15
LED5
KPT-2012
KP-2012
LED6
KPT-2012
KP-2012
LED7
KPT-2012
KP-2012
LED8
KPT-2012
KP-2012
RP2
8P4R-300
8P4RA
VCC
LED

DIGITAL INPUT

```
#include <stdio.h>
#include "NUC1xx.h"
#include "DrvGPIO.h"

int main (void)
{
      int pinValue;
      DrvGPIO_Open (E_GPB, 15, E_IO_INPUT);
      DrvGPIO_Open (E_GPA, 12, E_IO_OUTPUT);

      while(1)
      {
            pinValue = DrvGPIO_GetBit(E_GPB, 15);

            if(pinValue == 1) {
                  DrvGPIO_SetBit(E_GPA, 12);
            }
            else {
                  DrvGPIO_ClrBit(E_GPA, 12);
            }
      }
}
```

To execute digital input operation, we are still going to use DrvGPIO.h library.

PIN DEFINITION

To initialize a pin as INPUT, we can use the following command:

```
DrvGPIO_Open (E_GPB, 15, E_IO_INPUT);
```

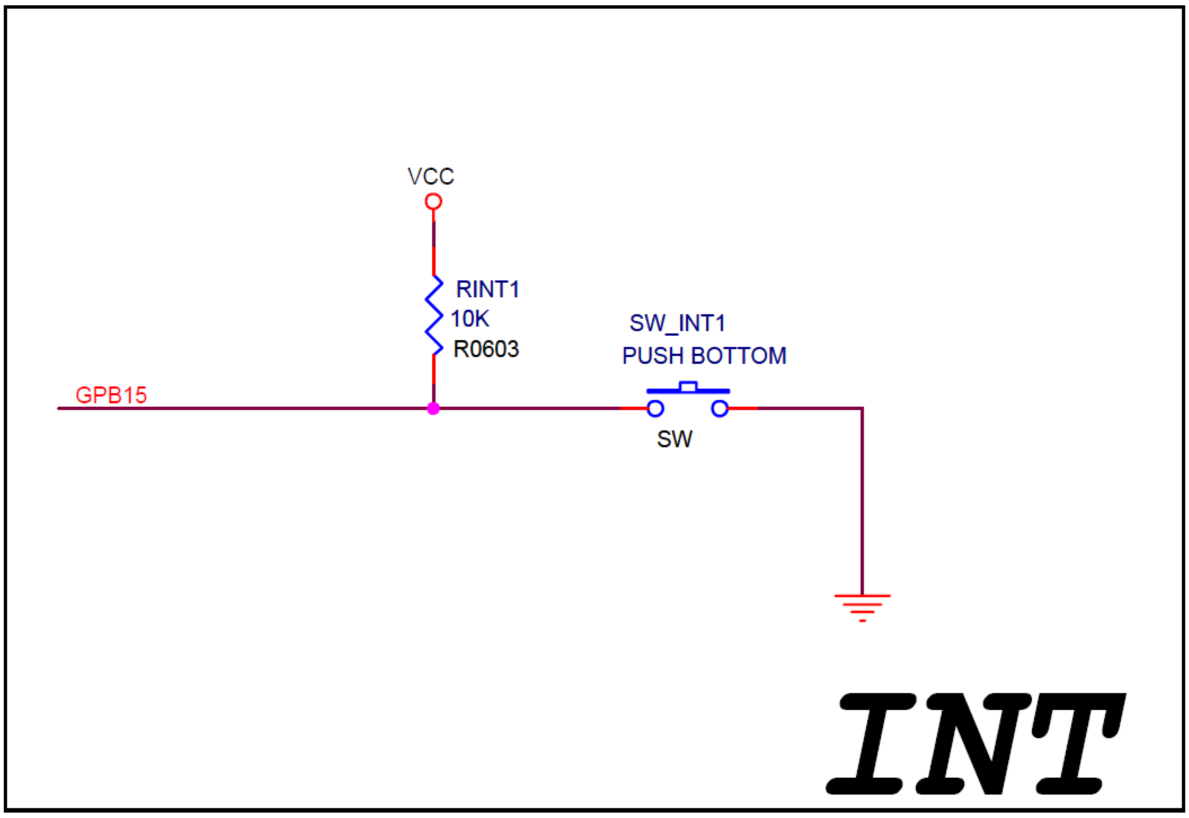
The command will initialize GPB15 pin as input. GPB15 is the SW_INT button just below the RESET button on the board.

DIGITAL SIGNAL READ

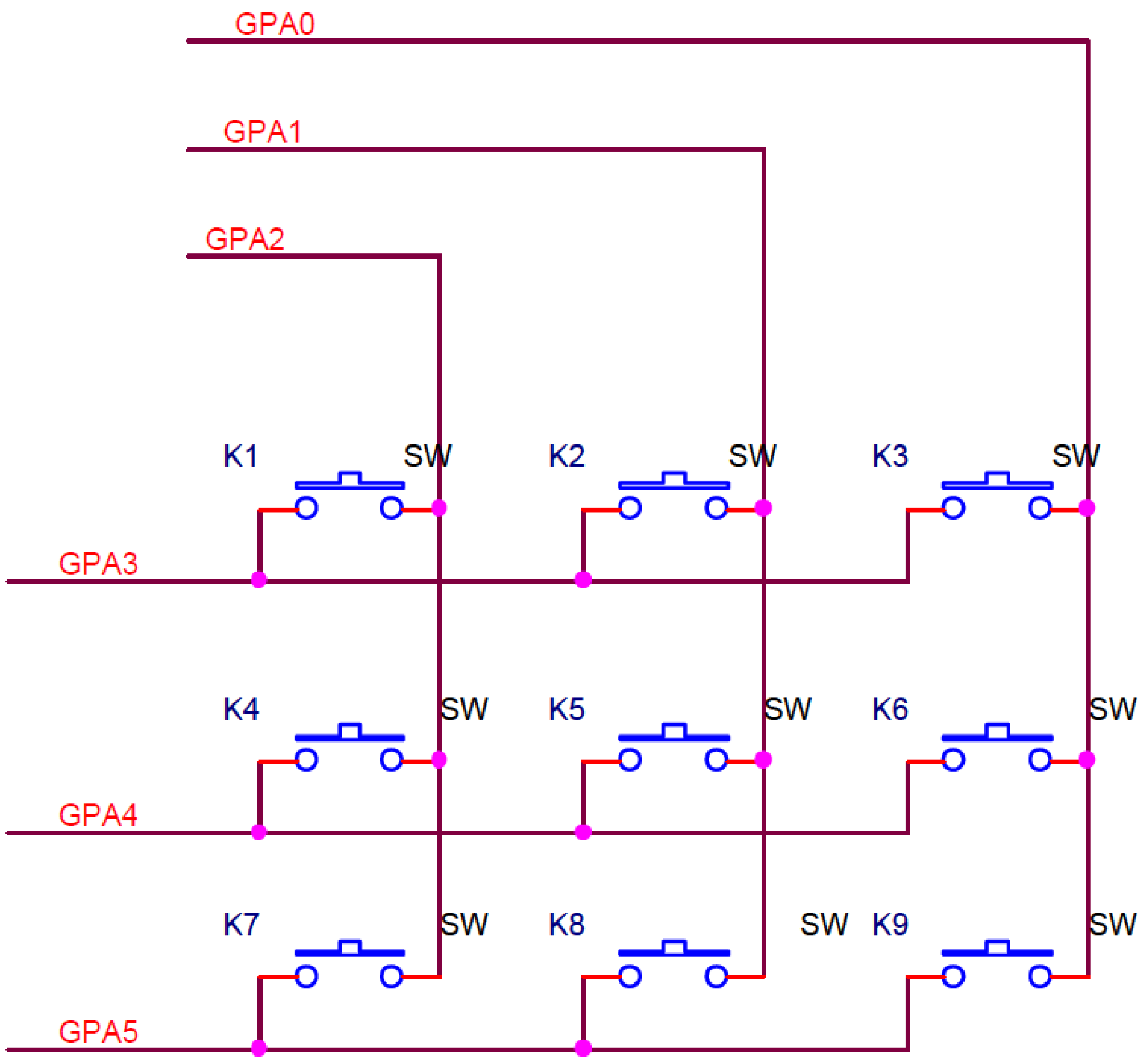
To read the value of the input pin, call the following function:

```
DrvGPIO_GetBit(E_GPB, 15);
```

It will return an integer value of 1 or 0. In the example, we store the value of the pin in an integer called *pinValue*. Then, we can use this value in *if* statement for turning the LED on or off. The SW_INT button is wired in pull-up configuration. This makes it so that the input read function returns 0 when pressed and 1 when released.



By combining your knowledge on both digital read and write functions, make the 7-segment displays shows the value of the corresponding button pressed. For example, pressing the top left button will display 1 on the 7-segment display. 2 on the next button and so on.


GPA0
GPA1
GPA2
K1
SW
K2
SW
K3
SW
GPA3
K4
SW
K5
SW
K6
SW
GPA4
K7
SW
K8
SW
K9
SW
GPA5

LIQUID CRYSTAL DISPLAY (LCD)

We need to use LCD driver in this example, so we need to add it to the project. By using the same step shown in section "*CREATING EMPTY PROJECT*" step 15, add *LCD_Driver.c* and *Ascii_Table.c* to your project folder. The folder can be found in:

```
NUC100SeriesBSP_CMSIS_v1.05.003\NuvotonPlatform_Keil\ Src\NUC1xx-
LB_002\LCD_Driver.c
```

```
NUC100SeriesBSP_CMSIS_v1.05.003\NuvotonPlatform_Keil\ Src\NUC1xx-
LB_002\Ascii_Table.C
```

```
#include <stdio.h>
#include "NUC1xx.h"
#include "DrvGPIO.h"
#include "LCD_Driver.h"

int main ()
{
      Initial_pannel();
      clr_all_pannal();

      DrvGPIO_Open(E_GPD, 14, E_IO_OUTPUT);
      DrvGPIO_ClrBit(E_GPD, 14);

      print_lcd(0, "     Line 1     ");
      print_lcd(1, "     Line 2     ");
      print_lcd(2, "     Line 3     ");
      print_lcd(3, "     Line 4     ");

      return 0;
}
```

ADDITIONAL LIBRARY

In this example, we need to include a new library that handles LCD related functions:

```
#include "LCD_Driver.h"
```

LCD INITIALIZATION

The following line initializes the LCD panel. It is the correct spelling.

```
Initial_pannel();
```

CLEAR LCD

The following line erases the entire LCD panel. It is also the correct spelling.

```
clr_all_pannal();
```

Backlight

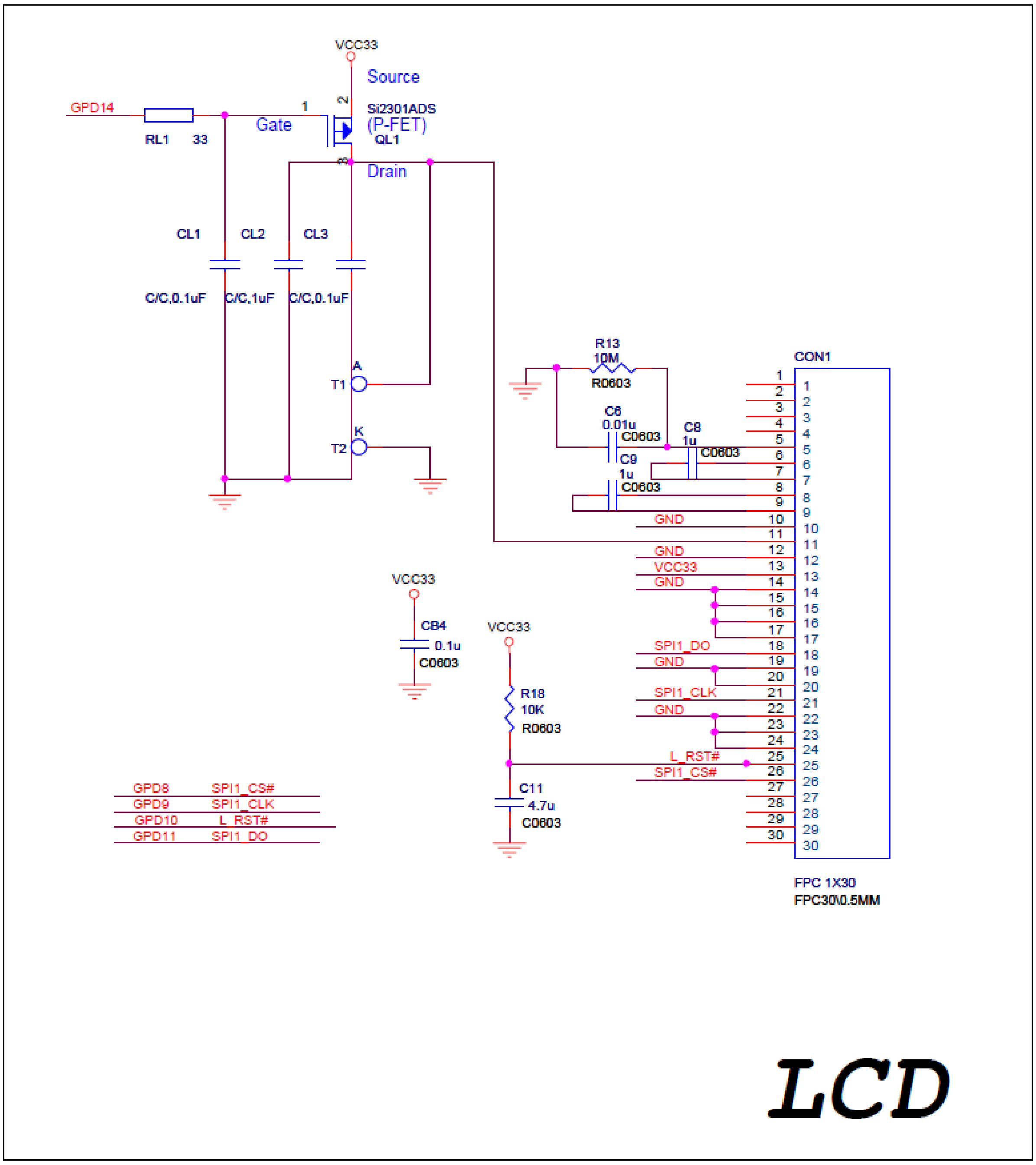
To make it easier to read the text on the LCD, we can turn on the backlight on. That is the following lines are for:

```
DrvGPIO_Open(E_GPD, 14, E_IO_OUTPUT);
DrvGPIO_ClrBit(E_GPD, 14);
```

Print Text on LCD

When using the LCD Driver library, we can print text across 4 rows and 16 columns. For example, if you want to print “WORDS” in the 3rd line, you may write:

```
print_lcd(2, "WORDS”);
```



7-SEGMENT DISPLAY

We need to use Seven Segment driver in this example, so we need to add it to the project. By using the same step shown in section "*CREATING EMPTY PROJECT*" step 15, add *Seven_Segment.c* to your project folder. The folder can be found in:

```
NUC100SeriesBSP_CMSIS_v1.05.003\NuvotonPlatform_Keil\ Src\NUC1xx-LB_002\
Seven_Segment.c
```

```
#include <stdio.h>
#include "NUC1xx.h"
#include "Seven_Segment.h"

void printNum(int num)
{
      int i;

      for(i=0; i<4; i++)
      {
            close_seven_segment();
            show_seven_segment(i, num % 10);
            num /= 10;
      }
}

int main ()
{
      while(1)
      {
            printNum(1234);
      }
}
```

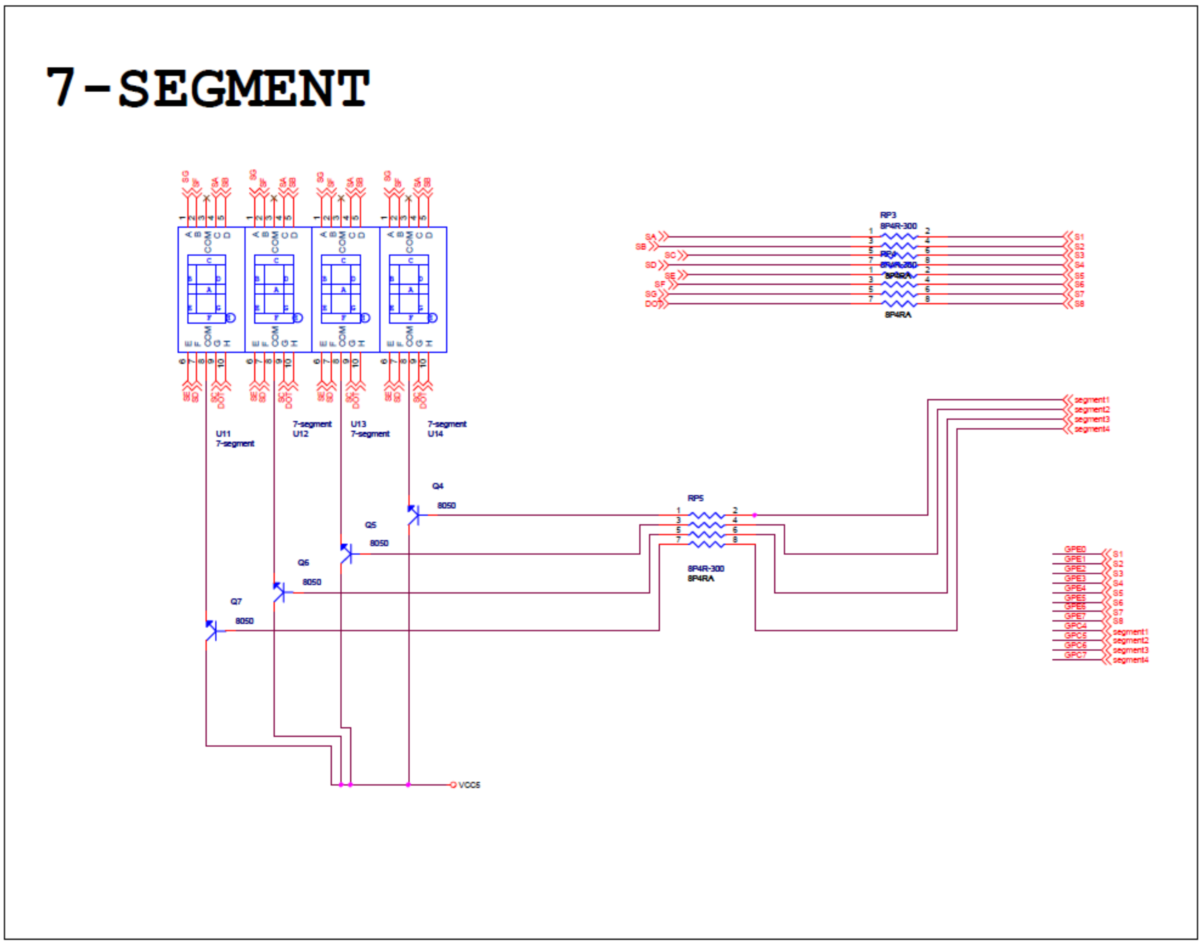
ADDITIONAL LIBRARY

In this example, we need to include a new library that handles 7-segment related functions:

```
#include "Seven_Segment.h"
```

PRINTNUM FUNCTION

The function basically loops through 4 last digits of the number that you passed to it. We then print one digit at a time starting from the back.



As you can see in the diagram above, all 4 units of the 7-segment displays are sharing the same data line. This would cause all segments displaying the same number. To print different number on each display, we must turn off the other displays and turn on only 1 display at a time. For example, when we call the function *printNum(1234)*, we are going to turn off all display, show the rightmost display, send digit 4 data and move to the next digit, which is 3. The process repeats for digit 2 and 1. This process causes the display to be dimmed since each display is only shown 1/4 time per cycle.

Fortunately, there is a way to brighten the display a bit. The idea is to pause for a moment when displaying each digit. This will allow the display to reach higher brightness level before being turned off again. However, if you put too long of a delay, the transition between each display will be visible. Try changing the number of __NOP() loop calls to see this effect.

```
#include <stdio.h>
#include "NUC1xx.h"
#include "Seven_Segment.h"

void printNum(int num)
{
      int i, j;

      for(i=0; i<4; i++)
      {
            close_seven_segment();
            show_seven_segment(i, num % 10);
            num /= 10;

            for(j=0; j<10000; j++)
                  __NOP();
      }
}

int main ()
{
      while(1)
      {
            printNum(1234);
      }
}
```