King Mongkut's University of Technology Thonburi

Department of Electronics and Telecommunication Engineering Faculty of Engineering
EIE/ENE 335 Digital Circuit and Microprocessor Lab for the 3rd year student

Experiment: introduction to Nu_LB-002 (Cortex™-M0)

Objectives

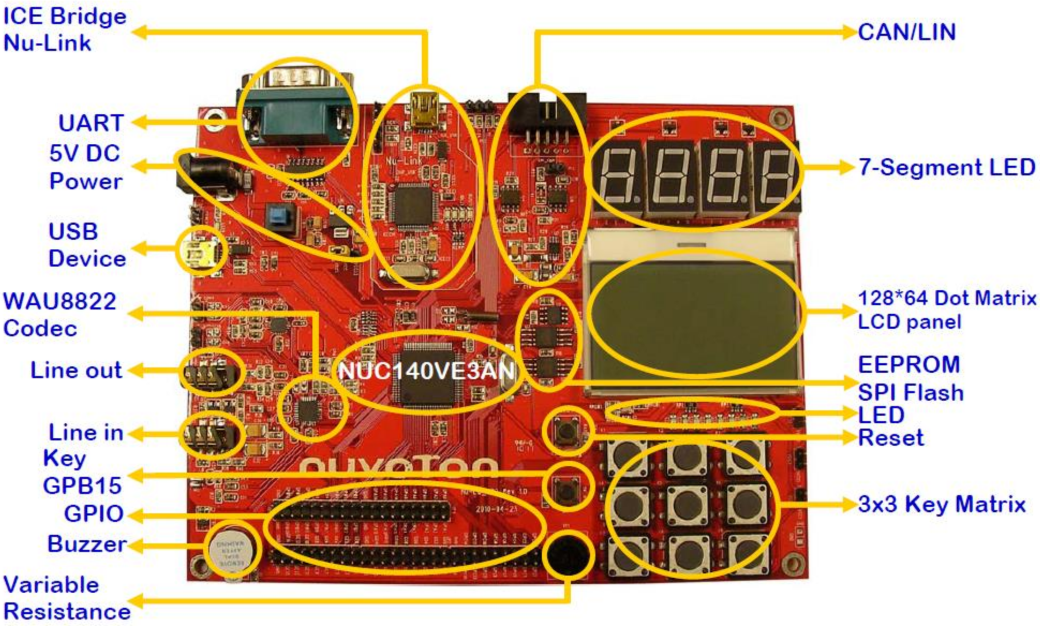
- To be familiar with Nu_LB-002 (Nuvoton learning board)
- Introduction to Keil
- How to use the NuMicro™ NUC100 series driver to do the fast application software development
- How to use GPIO/LCD

Background Theory

Nu_LB-002

The NUC100 series is 32-bit microcontrollers with embedded ARM® Cortex™-M0 core for industrial control at a cost equivalent to a traditional 8-bit microcontroller.

The NUC1XX series with Cortex™-M0 core runs up to 50MHz, up to 32K/64K/128K-byte embedded flash, and 4K/8K/16K-byte embedded SRAM, it also integrates Timers, Watchdog Timer, RTC, PDMA, UART, SPI/SSP, I2C, PWM Timer, GPIO, LIN, CAN, USB 2.0 FS Device, 12-bit ADC, Analog Comparator, Low Voltage Detector and Brown-out detector.



Equipment required

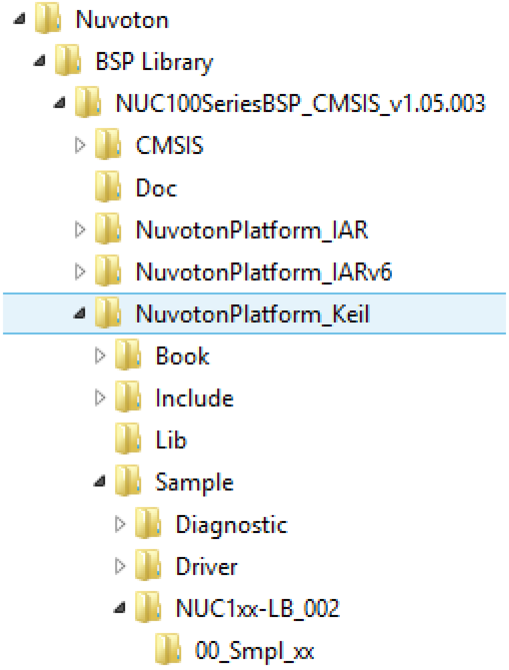
- Nu_LB-002 (Nuvoton learning board)

Reference:

1. Nu_LB-002 Rev 2.1 User's Manual
2. NuMicro^{TM} NUC130_140 Technical Reference Manual EN V2.02
3. NuMicro^{TM} NUC100 Series Driver Reference Guide V1.05.002

Procedure 1: the first example project

1. After installation of the software according to the procedure on the web-page (http://webstaff.kmutt.ac.th/~dejwoot.kha/CalenderYear/Year2556/ene335/index.html : HowTO install Nu tools), there are subdirectories under drive C:



```
Nuvoton
  BSP Library
    NUC100SeriesBSP_CMSIS_v1.05.003
      CMSIS
      Doc
      NuvotonPlatform_IAR
      NuvotonPlatform_IARv6
      NuvotonPlatform_Keil
        Book
        Include
        Lib
        Sample
          Diagnostic
          Driver
          NUC1xx-LB_002
            00_Smpl_xx
```

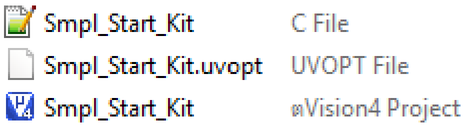
2. Download the zip file (Smpl_Start_Kit.zip) and save in the directory ("C:\Nuvoton\BSP Library\NUC100SeriesBSP_CMSIS_v1.05.003\NuvotonPlatform_Keil\Sample\NUC1xx-LB_002\").

3. Unzip the file into the same directory. And rename the new directory to your preference, so you can work on your own. (Smpl_Start_Kit_ENE335_secXX)

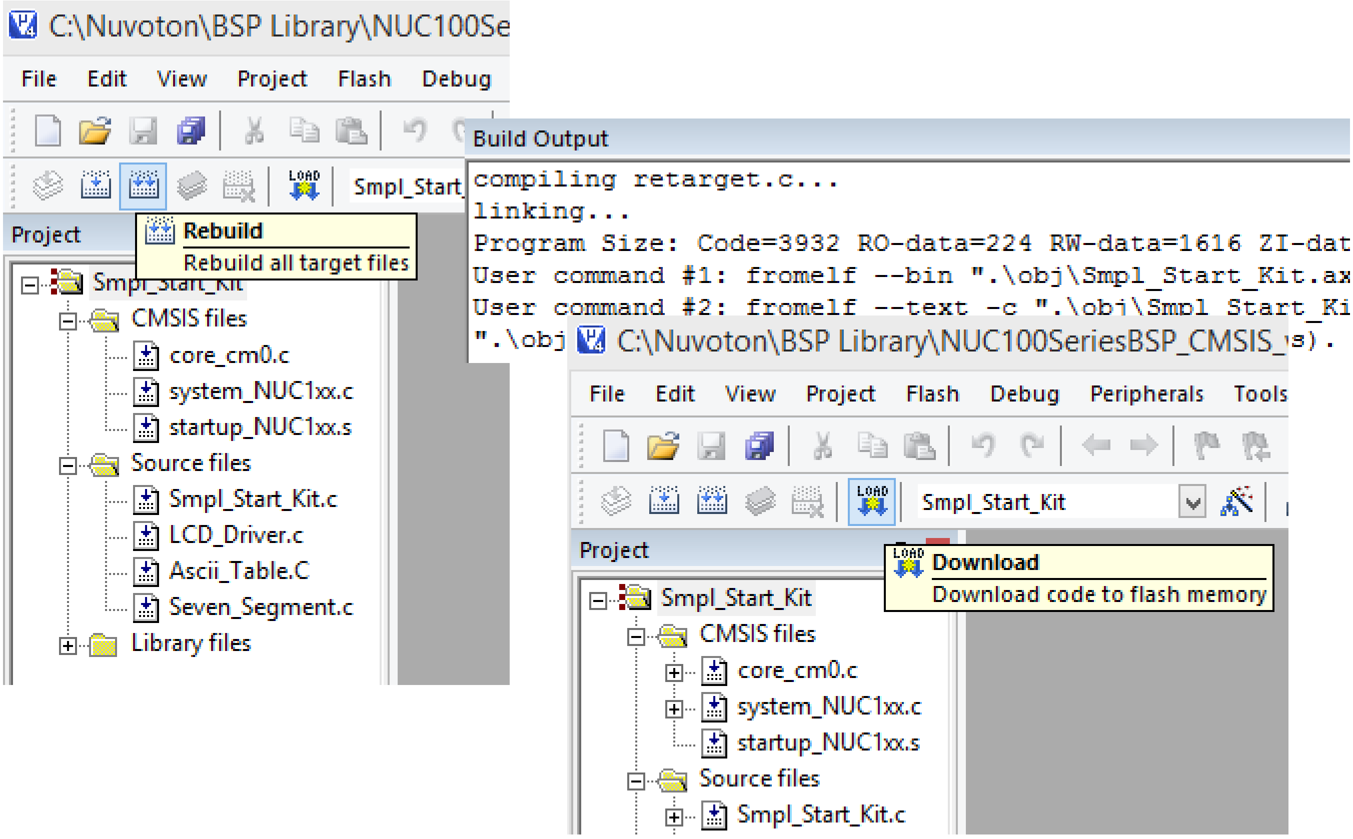
4. Open the project file by going to the new subdirectory that contain 3 files. Double click Smpl_Start_Kit.uvproj (Vision4 Project).

5. The Keil window will show up.

6. Compile the project to create the executable file by click at '**Rebuild**' icon.

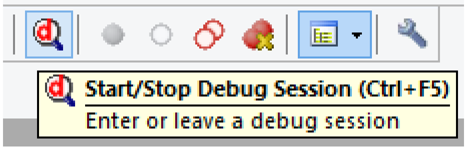


Smpl_Start_Kit	C File
Smpl_Start_Kit.uvopt	UVOPT File
Smpl_Start_Kit	µVision4 Project

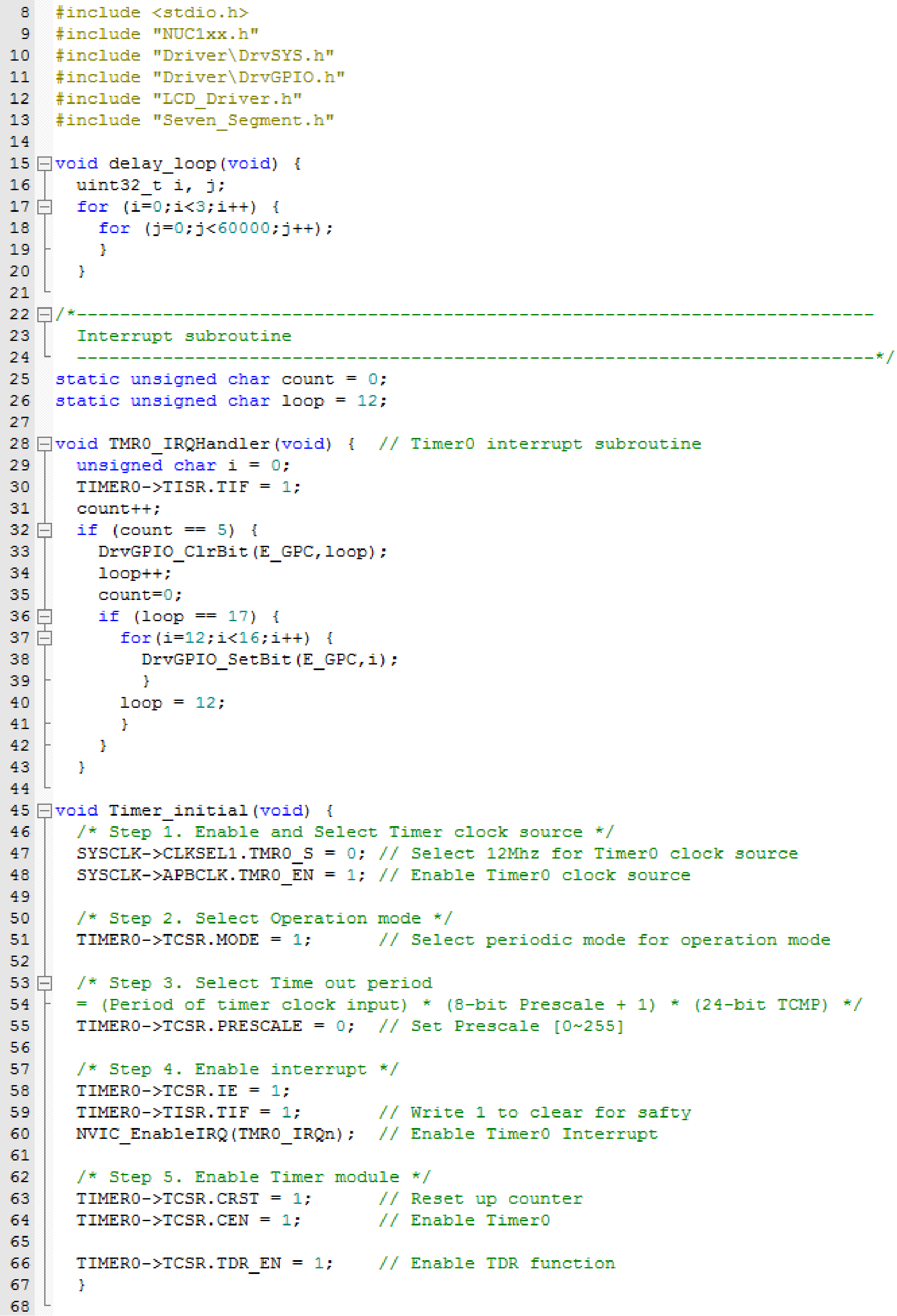


```
compiling retarget.c...
linking...
Program Size: Code=3932 RO-data=224 RW-data=1616 ZI-dat
User command #1: fromelf --bin ".\obj\Smpl_Start_Kit.ax
User command #2: fromelf --text -c ".\obj\Smpl_Start_Ki
".\obj                                              s).
```

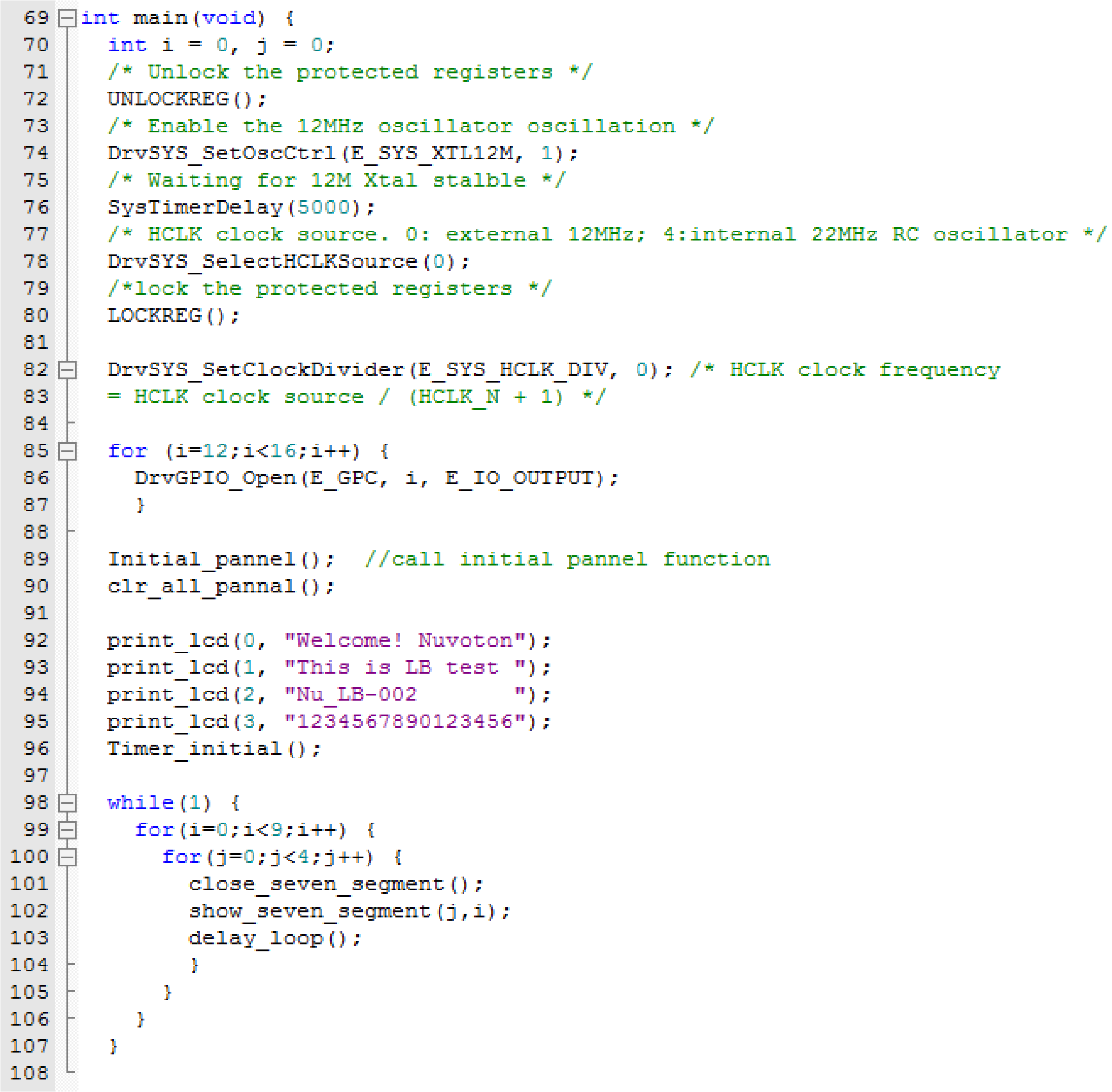
7. Make sure that there is no error after compilation is done. Connect the ICE Bridge Nu-Link to the PC using USB cable. Download the program to the Nu_LB-002 by clicking at the ‘**LOAD**’ icon.
8. To run the program, reset the Nu_LB-002. (Use the reset button on the board)
9. Or download and debug the program by clicking at the ‘Start/Stop **Debug**’ icon.


Start/Stop Debug Session (Ctrl+F5)
Enter or leave a debug session

10. Look at the ‘Smpl_Start_Kit.c’ and answer the following questions.



```c
#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvSYS.h"
#include "Driver\DrvGPIO.h"
#include "LCD_Driver.h"
#include "Seven_Segment.h"

void delay_loop(void) {
  uint32_t i, j;
  for (i=0;i<3;i++) {
    for (j=0;j<60000;j++);
    }
  }

/*-----------------------------------------------------------------------------
  Interrupt subroutine
  ----------------------------------------------------------------------------*/
static unsigned char count = 0;
static unsigned char loop = 12;

void TMR0_IRQHandler(void) {  // Timer0 interrupt subroutine
  unsigned char i = 0;
  TIMER0->TISR.TIF = 1;
  count++;
  if (count == 5) {
    DrvGPIO_ClrBit(E_GPC,loop);
    loop++;
    count=0;
    if (loop == 17) {
      for(i=12;i<16;i++) {
        DrvGPIO_SetBit(E_GPC,i);
        }
      loop = 12;
      }
    }
  }

void Timer_initial(void) {
  /* Step 1. Enable and Select Timer clock source */
  SYSCLK->CLKSEL1.TMR0_S = 0; // Select 12Mhz for Timer0 clock source
  SYSCLK->APBCLK.TMR0_EN = 1; // Enable Timer0 clock source

  /* Step 2. Select Operation mode */
  TIMER0->TCSR.MODE = 1;      // Select periodic mode for operation mode

  /* Step 3. Select Time out period
  = (Period of timer clock input) * (8-bit Prescale + 1) * (24-bit TCMP) */
  TIMER0->TCSR.PRESCALE = 0;  // Set Prescale [0~255]

  /* Step 4. Enable interrupt */
  TIMER0->TCSR.IE = 1;
  TIMER0->TISR.TIF = 1;       // Write 1 to clear for safty
  NVIC_EnableIRQ(TMR0_IRQn);  // Enable Timer0 Interrupt

  /* Step 5. Enable Timer module */
  TIMER0->TCSR.CRST = 1;      // Reset up counter
  TIMER0->TCSR.CEN = 1;       // Enable Timer0

  TIMER0->TCSR.TDR_EN = 1;    // Enable TDR function
  }

```



```
 69 int main(void) {
 70   int i = 0, j = 0;
 71   /* Unlock the protected registers */
 72   UNLOCKREG();
 73   /* Enable the 12MHz oscillator oscillation */
 74   DrvSYS_SetOscCtrl(E_SYS_XTL12M, 1);
 75   /* Waiting for 12M Xtal stalble */
 76   SysTimerDelay(5000);
 77   /* HCLK clock source. 0: external 12MHz; 4:internal 22MHz RC oscillator */
 78   DrvSYS_SelectHCLKSource(0);
 79   /*lock the protected registers */
 80   LOCKREG();
 81 
 82   DrvSYS_SetClockDivider(E_SYS_HCLK_DIV, 0); /* HCLK clock frequency
 83   = HCLK clock source / (HCLK_N + 1) */
 84 
 85   for (i=12;i<16;i++) {
 86     DrvGPIO_Open(E_GPC, i, E_IO_OUTPUT);
 87     }
 88 
 89   Initial_pannel();  //call initial pannel function
 90   clr_all_pannal();
 91 
 92   print_lcd(0, "Welcome! Nuvoton");
 93   print_lcd(1, "This is LB test ");
 94   print_lcd(2, "Nu_LB-002       ");
 95   print_lcd(3, "1234567890123456");
 96   Timer_initial();
 97 
 98   while(1) {
 99     for(i=0;i<9;i++) {
100       for(j=0;j<4;j++) {
101         close_seven_segment();
102         show_seven_segment(j,i);
103         delay_loop();
104         }
105       }
106     }
107   }
108 
```

Questions (the first example project)

1. What does the function on line 33 (DrvGPIO_ClrBit(E_GPC,loop);) do? When loop = 15. (Try set Breakpoint on line 33, then 'Run' and 'Step over the current line' in Debug mode)

2. What does the function on line 38 (DrvGPIO_SetBit(E_GPC,i);) do? When i = 14.

3. What does the function on line 102 (show_seven_segment(j,i);) do? When j = 1 and i = 3.

4. What is the time out period of Timer0? (Show how to get the result)

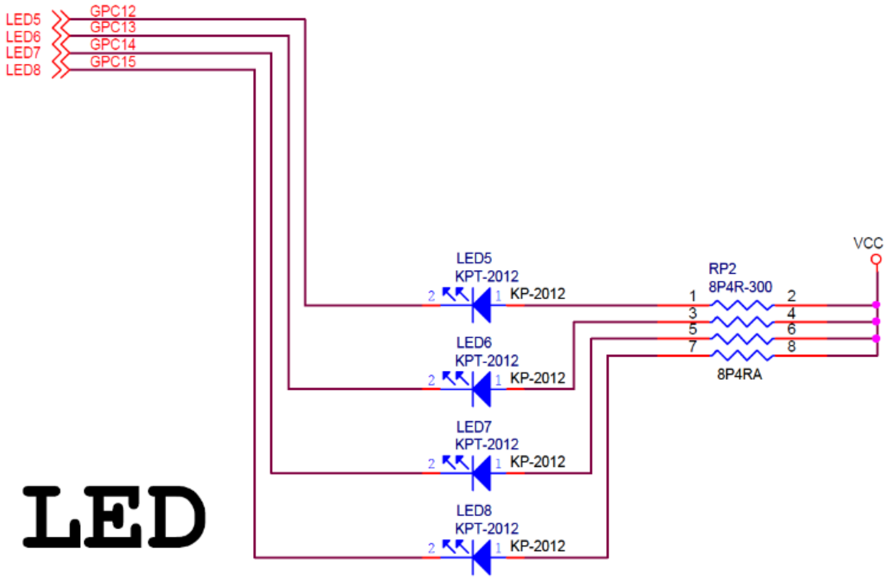
5. How long does the LED (GPC12) turn 'ON'? (Show how to get the result)

General Purpose I/O (GPIO)

NuMicro™ NUC130/NUC140 has up to 80 General Purpose I/O pins can be shared with other function pins; it depends on the chip configuration. These 80 pins are arranged in 5 ports named with GPIOA, GPIOB, GPIOC, GPIOD and GPIOE. Each port equips maximum 16 pins. Each one of the 80 pins is independent and has the corresponding register bits to control the pin mode function and data.

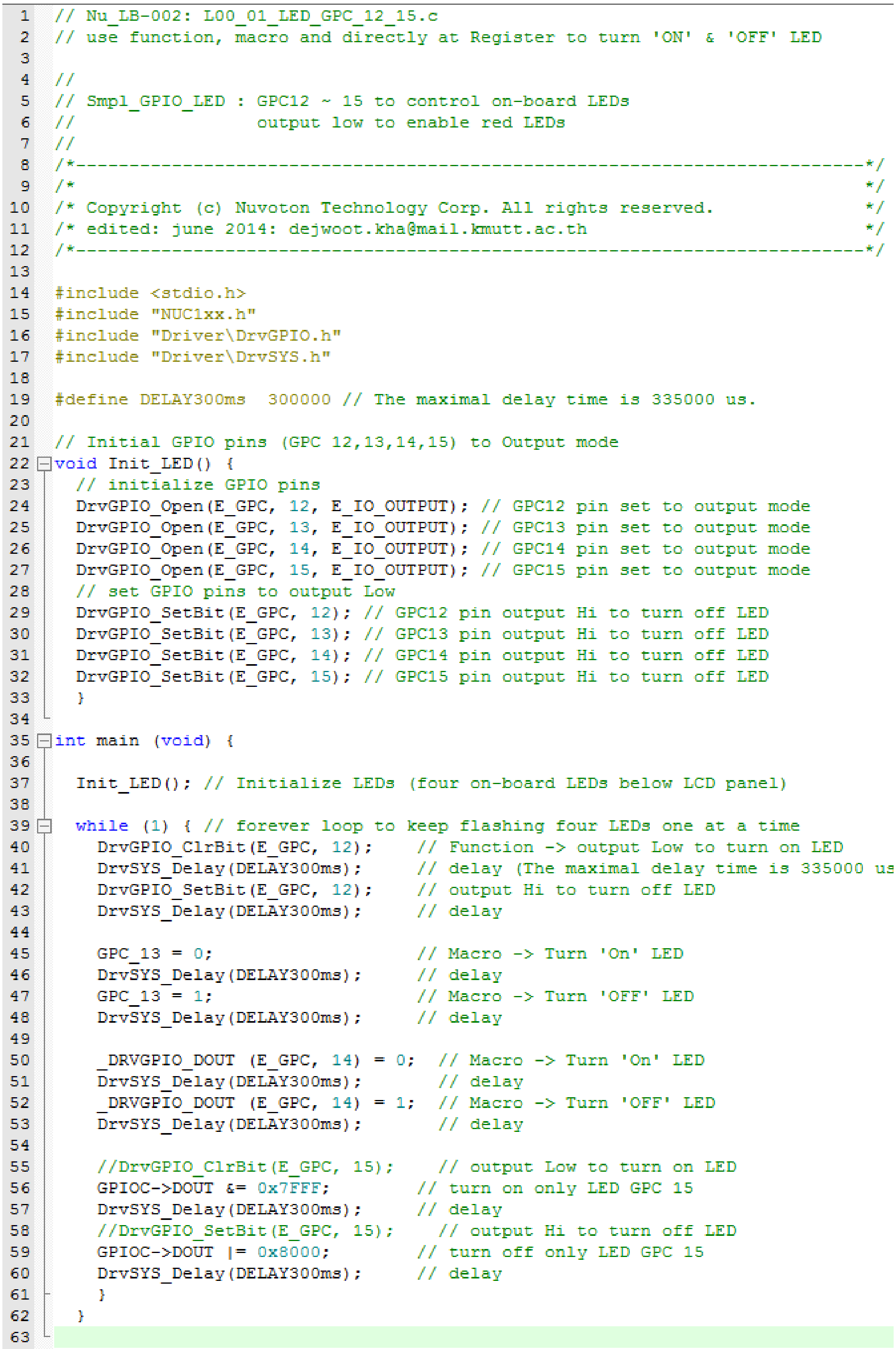
The I/O type of each of I/O pins can be configured by software individually as input, output, and open-drain or quasi-bidirectional mode. After reset, the I/O type of all pins stay in quasi-bidirectional mode and port data register GPIOx_DOUT[15:0] resets to 0x0000_FFFF. Each I/O pin equips a very weakly individual pull-up resistor which is about 110 KΩ~300 KΩ for VDD is from 5.0 V to 2.5 V.

LED: GPC8-GPC15 to control LED function.



Procedure 2: LED: GPC8-GPC15 to control LED function.

1. Replace the content of the ‘**Smpl_Start_Kit.c**’ with the ‘**LED**’ lab file.
2. Compile the project, and run the program.
3. Study the program and answer the following questions.



```c
// Nu_LB-002: L00_01_LED_GPC_12_15.c
// use function, macro and directly at Register to turn 'ON' & 'OFF' LED

//
// Smpl_GPIO_LED : GPC12 ~ 15 to control on-board LEDs
//                 output low to enable red LEDs
//
/*--------------------------------------------------------------------------*/
/*                                                                          */
/* Copyright (c) Nuvoton Technology Corp. All rights reserved.              */
/* edited: june 2014: dejwoot.kha@mail.kmutt.ac.th                          */
/*--------------------------------------------------------------------------*/

#include <stdio.h>
#include "NUC1xx.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvSYS.h"

#define DELAY300ms  300000 // The maximal delay time is 335000 us.

// Initial GPIO pins (GPC 12,13,14,15) to Output mode
void Init_LED() {
  // initialize GPIO pins
  DrvGPIO_Open(E_GPC, 12, E_IO_OUTPUT); // GPC12 pin set to output mode
  DrvGPIO_Open(E_GPC, 13, E_IO_OUTPUT); // GPC13 pin set to output mode
  DrvGPIO_Open(E_GPC, 14, E_IO_OUTPUT); // GPC14 pin set to output mode
  DrvGPIO_Open(E_GPC, 15, E_IO_OUTPUT); // GPC15 pin set to output mode
  // set GPIO pins to output Low
  DrvGPIO_SetBit(E_GPC, 12); // GPC12 pin output Hi to turn off LED
  DrvGPIO_SetBit(E_GPC, 13); // GPC13 pin output Hi to turn off LED
  DrvGPIO_SetBit(E_GPC, 14); // GPC14 pin output Hi to turn off LED
  DrvGPIO_SetBit(E_GPC, 15); // GPC15 pin output Hi to turn off LED
  }

int main (void) {

  Init_LED(); // Initialize LEDs (four on-board LEDs below LCD panel)

  while (1) { // forever loop to keep flashing four LEDs one at a time
    DrvGPIO_ClrBit(E_GPC, 12);    // Function -> output Low to turn on LED
    DrvSYS_Delay(DELAY300ms);     // delay (The maximal delay time is 335000 us
    DrvGPIO_SetBit(E_GPC, 12);    // output Hi to turn off LED
    DrvSYS_Delay(DELAY300ms);     // delay

    GPC_13 = 0;                   // Macro -> Turn 'On' LED
    DrvSYS_Delay(DELAY300ms);     // delay
    GPC_13 = 1;                   // Macro -> Turn 'OFF' LED
    DrvSYS_Delay(DELAY300ms);     // delay

    _DRVGPIO_DOUT (E_GPC, 14) = 0;  // Macro -> Turn 'On' LED
    DrvSYS_Delay(DELAY300ms);       // delay
    _DRVGPIO_DOUT (E_GPC, 14) = 1;  // Macro -> Turn 'OFF' LED
    DrvSYS_Delay(DELAY300ms);       // delay

    //DrvGPIO_ClrBit(E_GPC, 15);    // output Low to turn on LED
    GPIOC->DOUT &= 0x7FFF;        // turn on only LED GPC 15
    DrvSYS_Delay(DELAY300ms);     // delay
    //DrvGPIO_SetBit(E_GPC, 15);    // output Hi to turn off LED
    GPIOC->DOUT |= 0x8000;        // turn off only LED GPC 15
    DrvSYS_Delay(DELAY300ms);     // delay
    }
  }

```

Questions (LED)

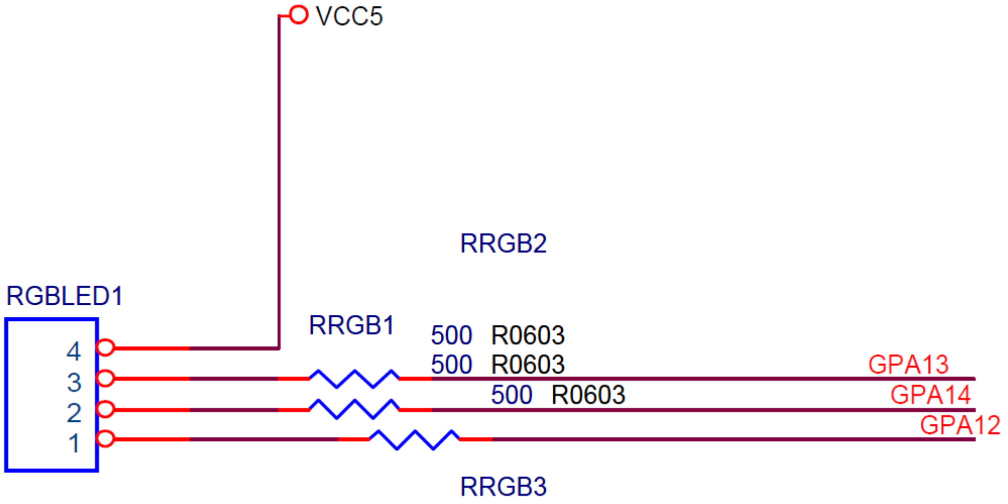
1. What is the difference between Function on line 40 and Macro on line 45?

2. From previous question, which one is execute faster and why?

3. What is the difference between Macro on line 45 and Macro on line 50?

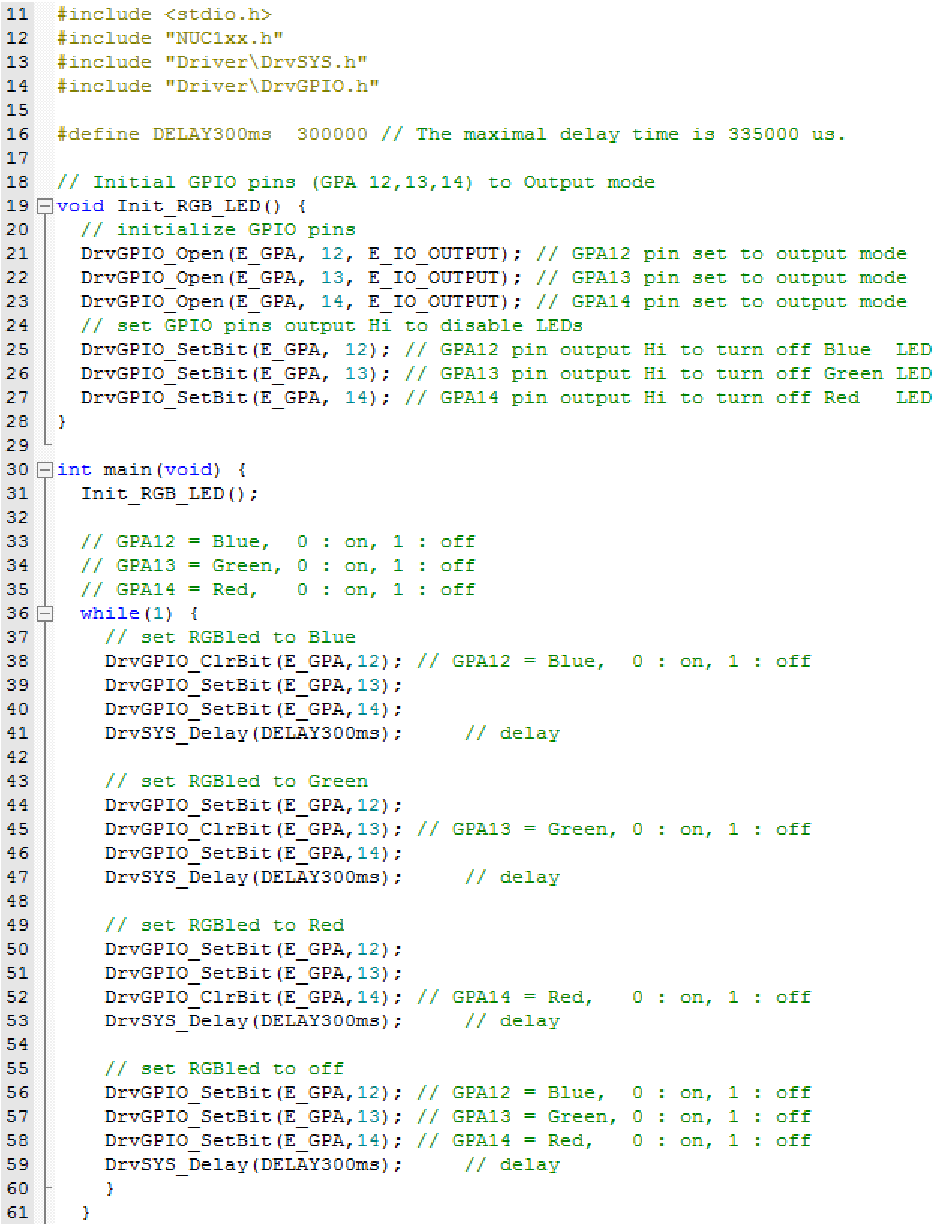
4. From previous question, which one is execute faster and why?

RGBLED: GPA12-GPA13 to control RGB LED function



Procedure 3: RGBLED: GPA12-GPA13 to control RGB LED function

1. Replace the content of the ‘**Smpl_Start_Kit.c**’ with the ‘**RGB-LED**’ lab file.
2. Compile the project, and run the program.
3. Study the program and answer the following questions.



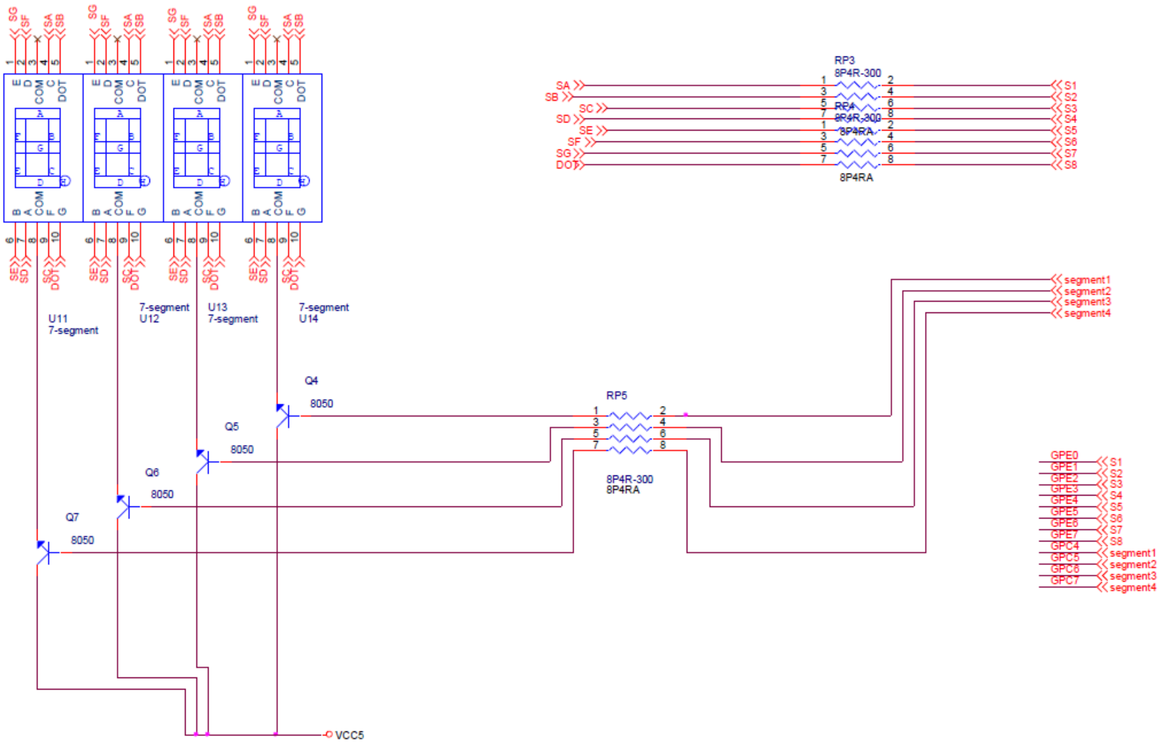
```
11 #include <stdio.h>
12 #include "NUC1xx.h"
13 #include "Driver\DrvSYS.h"
14 #include "Driver\DrvGPIO.h"
15
16 #define DELAY300ms  300000 // The maximal delay time is 335000 us.
17
18 // Initial GPIO pins (GPA 12,13,14) to Output mode
19 void Init_RGB_LED() {
20   // initialize GPIO pins
21   DrvGPIO_Open(E_GPA, 12, E_IO_OUTPUT); // GPA12 pin set to output mode
22   DrvGPIO_Open(E_GPA, 13, E_IO_OUTPUT); // GPA13 pin set to output mode
23   DrvGPIO_Open(E_GPA, 14, E_IO_OUTPUT); // GPA14 pin set to output mode
24   // set GPIO pins output Hi to disable LEDs
25   DrvGPIO_SetBit(E_GPA, 12); // GPA12 pin output Hi to turn off Blue  LED
26   DrvGPIO_SetBit(E_GPA, 13); // GPA13 pin output Hi to turn off Green LED
27   DrvGPIO_SetBit(E_GPA, 14); // GPA14 pin output Hi to turn off Red   LED
28 }
29
30 int main(void) {
31   Init_RGB_LED();
32
33   // GPA12 = Blue,  0 : on, 1 : off
34   // GPA13 = Green, 0 : on, 1 : off
35   // GPA14 = Red,   0 : on, 1 : off
36   while(1) {
37     // set RGBled to Blue
38     DrvGPIO_ClrBit(E_GPA,12); // GPA12 = Blue,  0 : on, 1 : off
39     DrvGPIO_SetBit(E_GPA,13);
40     DrvGPIO_SetBit(E_GPA,14);
41     DrvSYS_Delay(DELAY300ms);     // delay
42
43     // set RGBled to Green
44     DrvGPIO_SetBit(E_GPA,12);
45     DrvGPIO_ClrBit(E_GPA,13); // GPA13 = Green, 0 : on, 1 : off
46     DrvGPIO_SetBit(E_GPA,14);
47     DrvSYS_Delay(DELAY300ms);     // delay
48
49     // set RGBled to Red
50     DrvGPIO_SetBit(E_GPA,12);
51     DrvGPIO_SetBit(E_GPA,13);
52     DrvGPIO_ClrBit(E_GPA,14); // GPA14 = Red,   0 : on, 1 : off
53     DrvSYS_Delay(DELAY300ms);     // delay
54
55     // set RGBled to off
56     DrvGPIO_SetBit(E_GPA,12); // GPA12 = Blue,  0 : on, 1 : off
57     DrvGPIO_SetBit(E_GPA,13); // GPA13 = Green, 0 : on, 1 : off
58     DrvGPIO_SetBit(E_GPA,14); // GPA14 = Red,   0 : on, 1 : off
59     DrvSYS_Delay(DELAY300ms);     // delay
60     }
61   }
```

Questions (RGBLED)

1. What is an alternate function for the GPA12

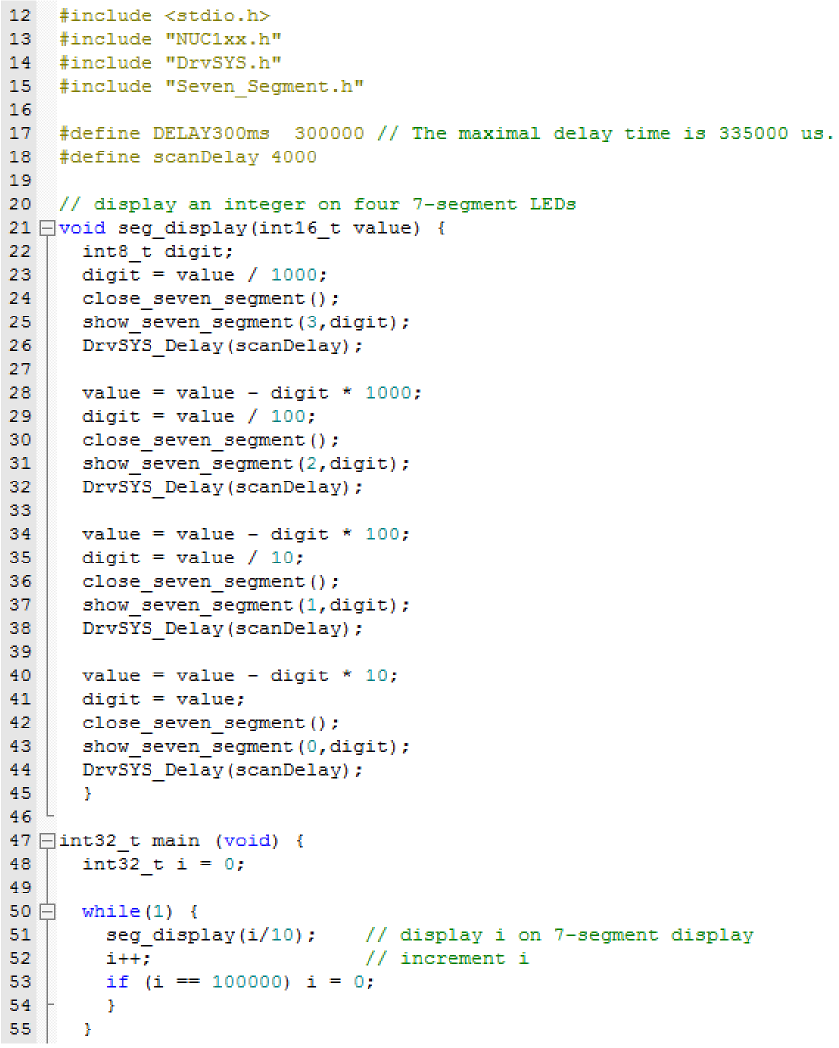
2. What Macros functions can we replace the function on line 38? (2 answers)

7-SEGMENT: GPE0-GPE7, GPC4-GPC7 control



Procedure 4: 7-SEGMENT: GPE0-GPE7, GPC4-GPC7 control

1. Replace the content of the ‘**Smpl_Start_Kit.c**’ with the ‘7SEG’ lab file.
2. Compile the project, and run the program.
3. Study the program and answer the following question.

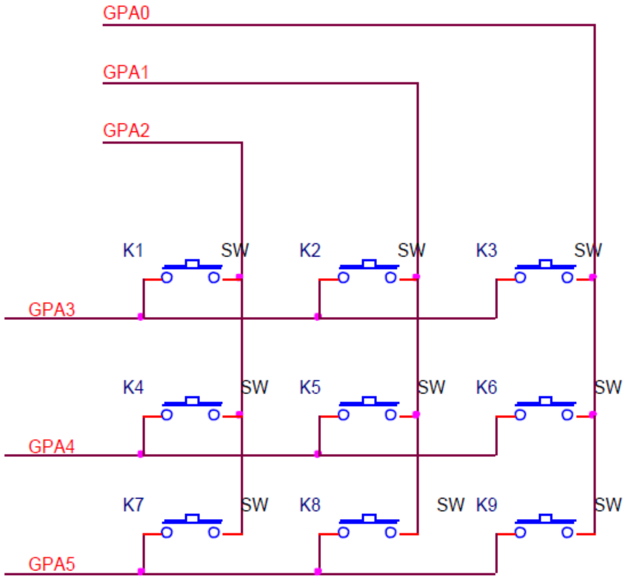


```c
#include <stdio.h>
#include "NUC1xx.h"
#include "DrvSYS.h"
#include "Seven_Segment.h"

#define DELAY300ms  300000 // The maximal delay time is 335000 us.
#define scanDelay 4000

// display an integer on four 7-segment LEDs
void seg_display(int16_t value) {
  int8_t digit;
  digit = value / 1000;
  close_seven_segment();
  show_seven_segment(3,digit);
  DrvSYS_Delay(scanDelay);

  value = value - digit * 1000;
  digit = value / 100;
  close_seven_segment();
  show_seven_segment(2,digit);
  DrvSYS_Delay(scanDelay);

  value = value - digit * 100;
  digit = value / 10;
  close_seven_segment();
  show_seven_segment(1,digit);
  DrvSYS_Delay(scanDelay);

  value = value - digit * 10;
  digit = value;
  close_seven_segment();
  show_seven_segment(0,digit);
  DrvSYS_Delay(scanDelay);
  }

int32_t main (void) {
  int32_t i = 0;

  while(1) {
    seg_display(i/10);     // display i on 7-segment display
    i++;                   // increment i
    if (i == 100000) i = 0;
    }
  }
```

Questions (7-SEGMENT)

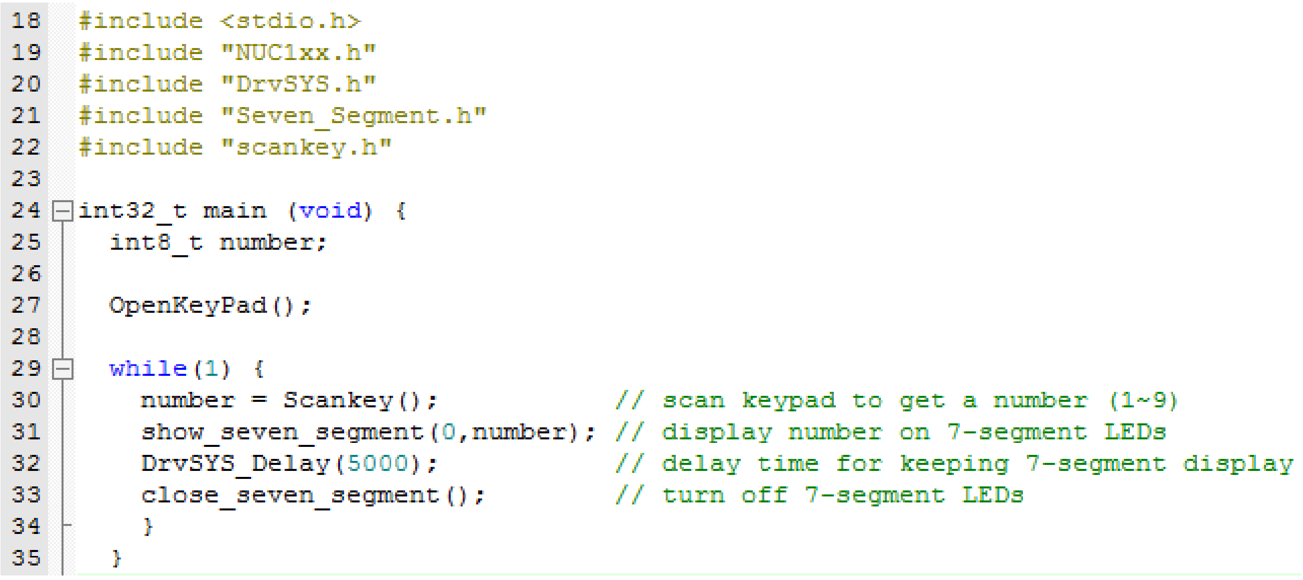
1. What is the function to display the left most digit with ‘7’.

KEYBOARD: GPA0-GPA5 to control KEYBOARD function

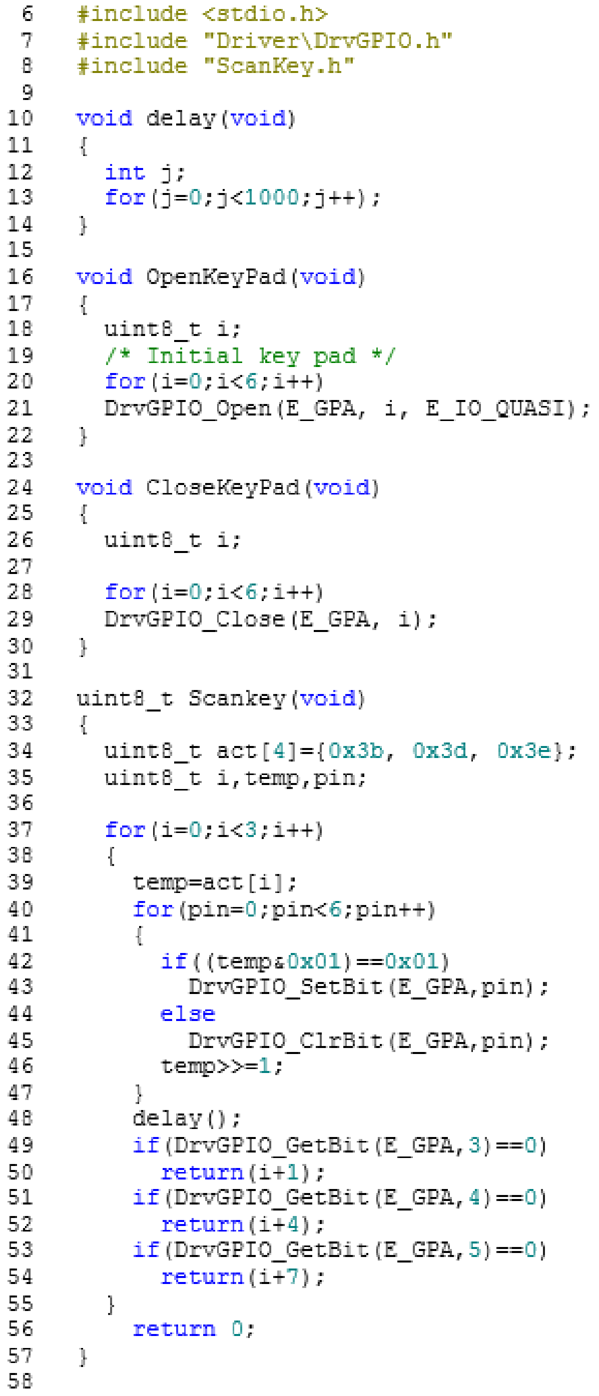


Procedure 5: KEYBOARD: GPA0-GPA5 to control KEYBOARD function

1. Replace the content of the ‘**Smpl_Start_Kit.c**’ with the ‘KeyPad-7SEG’ lab file.
2. Compile the project, and run the program. (Add ScanKey.c to the project, from "C:\Nuvoton\BSP Library\NUC100SeriesBSP_CMSIS_v1.05.003\NuvotonPlatform_Keil\Src\NUC1xx-LB_002\ScanKey.c")
3. Study the program and answer the following question.



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

int32_t main (void) {
  int8_t number;

  OpenKeyPad();

  while(1) {
    number = Scankey();           // scan keypad to get a number (1~9)
    show_seven_segment(0,number); // display number on 7-segment LEDs
    DrvSYS_Delay(5000);           // delay time for keeping 7-segment display
    close_seven_segment();        // turn off 7-segment LEDs
    }
  }
```

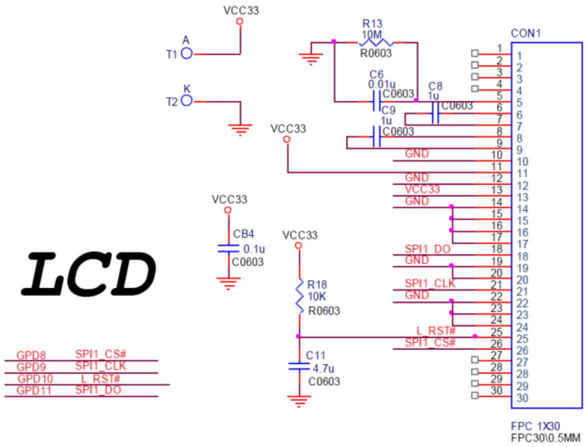


```
 6   #include <stdio.h>
 7   #include "Driver\DrvGPIO.h"
 8   #include "ScanKey.h"
 9
10   void delay(void)
11   {
12     int j;
13     for(j=0;j<1000;j++);
14   }
15
16   void OpenKeyPad(void)
17   {
18     uint8_t i;
19     /* Initial key pad */
20     for(i=0;i<6;i++)
21     DrvGPIO_Open(E_GPA, i, E_IO_QUASI);
22   }
23
24   void CloseKeyPad(void)
25   {
26     uint8_t i;
27
28     for(i=0;i<6;i++)
29     DrvGPIO_Close(E_GPA, i);
30   }
31
32   uint8_t Scankey(void)
33   {
34     uint8_t act[4]={0x3b, 0x3d, 0x3e};
35     uint8_t i,temp,pin;
36
37     for(i=0;i<3;i++)
38     {
39       temp=act[i];
40       for(pin=0;pin<6;pin++)
41       {
42         if((temp&0x01)==0x01)
43           DrvGPIO_SetBit(E_GPA,pin);
44         else
45           DrvGPIO_ClrBit(E_GPA,pin);
46         temp>>=1;
47       }
48       delay();
49       if(DrvGPIO_GetBit(E_GPA,3)==0)
50         return(i+1);
51       if(DrvGPIO_GetBit(E_GPA,4)==0)
52         return(i+4);
53       if(DrvGPIO_GetBit(E_GPA,5)==0)
54         return(i+7);
55     }
56       return 0;
57   }
58
```

Questions (KEYBOARD)

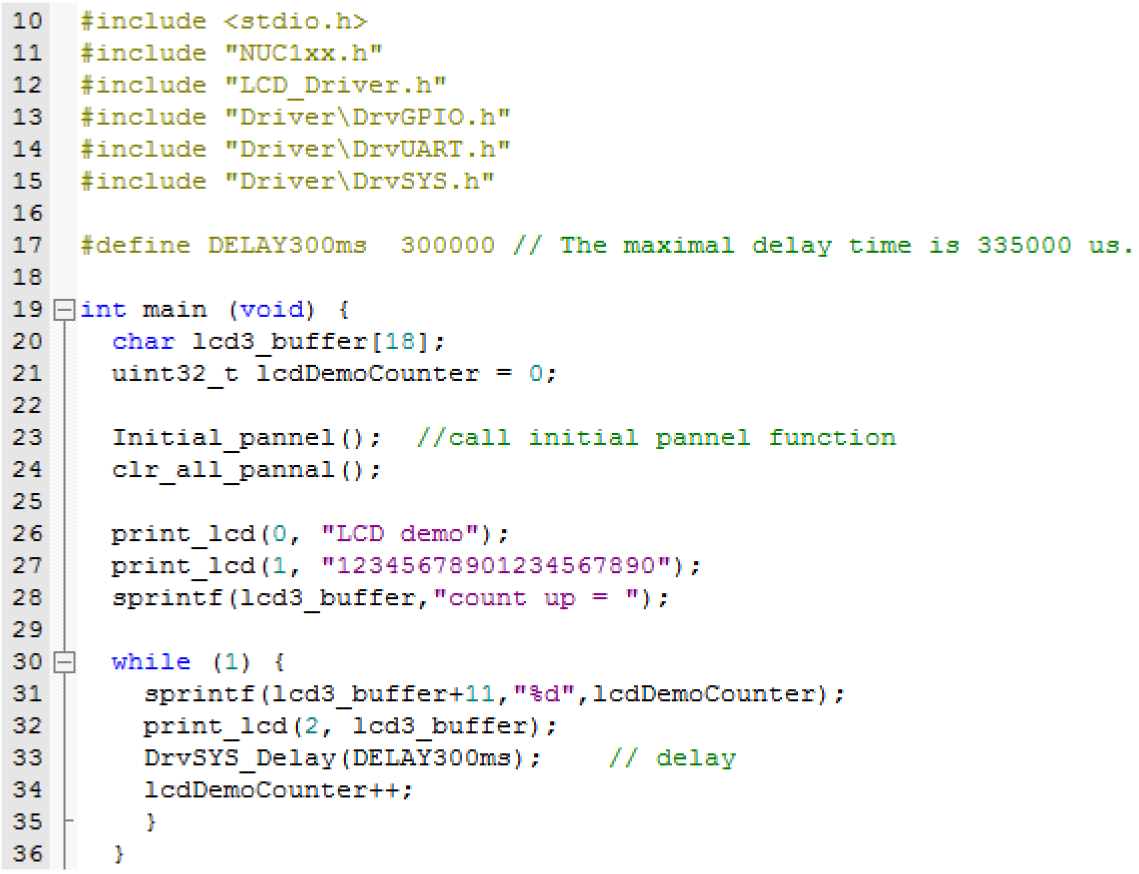
1. What is going to happen if the code on line 34 (in ScanKey.c) are changed from 0x3b, 0x3d, 0x3e to 0x3F, 0x3F, and 0x3d, respectively?

LCD: GPD8-GPD11 connect to LCD function.



Procedure 6: LCD: GPD8-GPD11 connect to LCD function.

1. Replace the content of the ‘**Smpl_Start_Kit.c**’ with the ‘**LCD**’ lab file.
2. Compile the project, and run the program.
3. Study the program and do the assignment in the class.



```c
#include <stdio.h>
#include "NUC1xx.h"
#include "LCD_Driver.h"
#include "Driver\DrvGPIO.h"
#include "Driver\DrvUART.h"
#include "Driver\DrvSYS.h"

#define DELAY300ms  300000 // The maximal delay time is 335000 us.

int main (void) {
  char lcd3_buffer[18];
  uint32_t lcdDemoCounter = 0;

  Initial_pannel();  //call initial pannel function
  clr_all_pannal();

  print_lcd(0, "LCD demo");
  print_lcd(1, "12345678901234567890");
  sprintf(lcd3_buffer,"count up = ");

  while (1) {
    sprintf(lcd3_buffer+11,"%d",lcdDemoCounter);
    print_lcd(2, lcd3_buffer);
    DrvSYS_Delay(DELAY300ms);    // delay
    lcdDemoCounter++;
    }
  }
```

Assignment(s)

Summarize what you suppose to learn in this class.