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: I²C Serial Interface

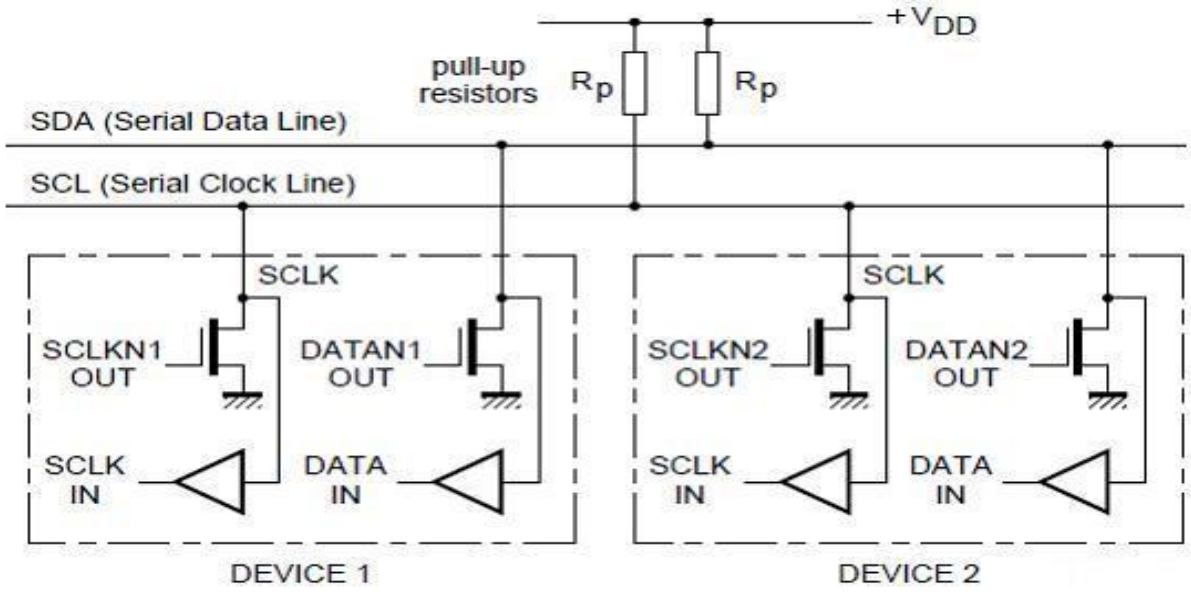
Objectives

- How to use
  - the NuMicro™ NUC100 series driver to do the fast application software development
  - I²C Serial Interface
  - Serial EEPROM (64K: 24LC64)
  - Remote 8-bit I/O expander (PCF8574)

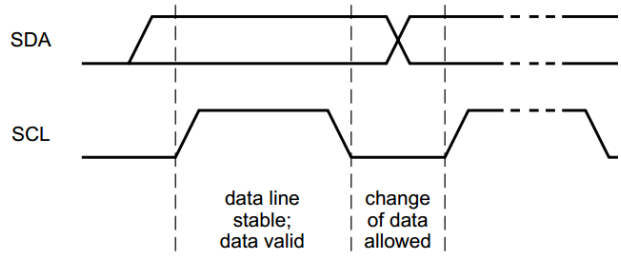
Background Theory

I2C Serial Interface

I2C is bi-directional serial bus with two wires that provides a simple and efficient method of data exchange between devices. The I2C standard is a true multi-master bus including collision detection and arbitration that prevents data corruption if two or more masters attempt to control the bus simultaneously. Serial, 8-bit oriented bi-directional data transfers can be made up 1.0 Mbps. A two-wired and bi-directional bus, comprised by SDA and SCL, may include multi masters and multi slaves with Open-Drain driving type, connected to a positive supply voltage via pull-up resistors.

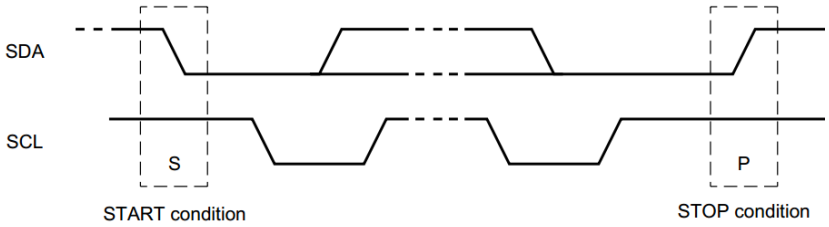


Bit transfer

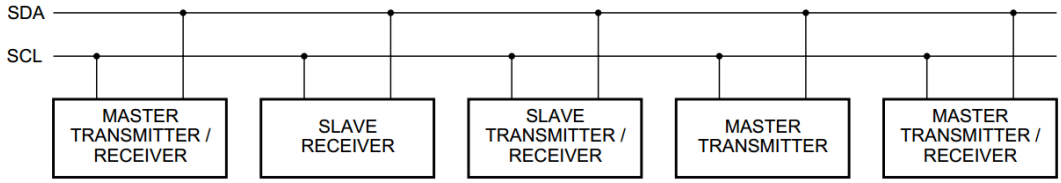


One data bit is transferred during each clock pulse.

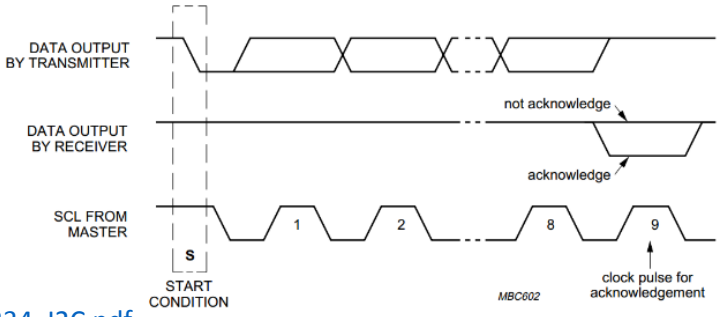
Start and stop conditions



System configuration



Acknowledge



See lecture note: L334_I2C.pdf

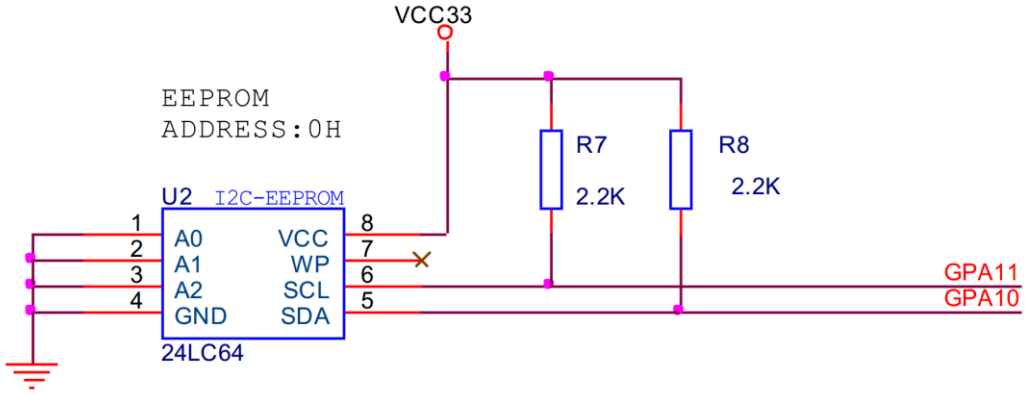
Equipment required

- Nu_LB-002 (Nuvoton learning board)

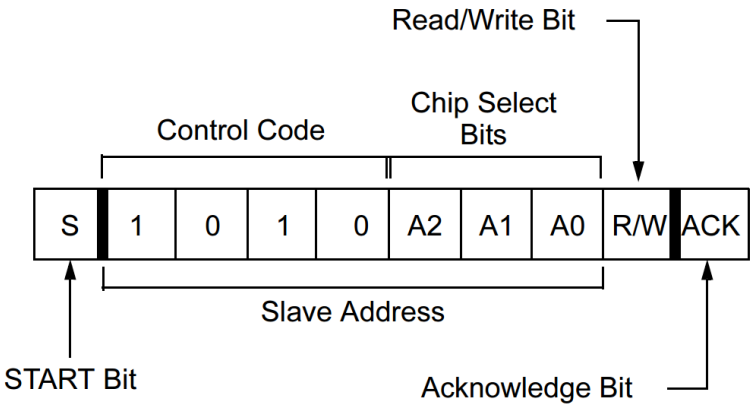
Reference:

1. Nu_LB-002 Rev 2.1 User's Manual
2. NuMicro™ NUC130_140 Technical Reference Manual EN V2.02
3. NuMicro™ NUC100 Series Driver Reference Guide V1.05.002
4. 24LC64 datasheet
5. PCF8574 datasheet

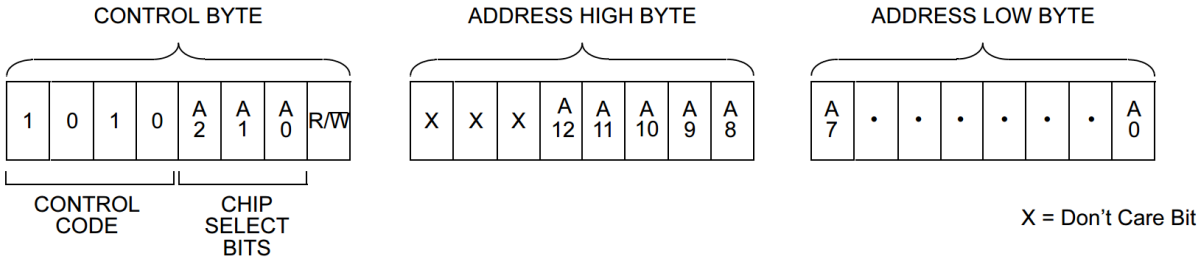
I2C Serial Interface (I2C1): GPA10-GPA11 connect to I2C Flash (24LC64)



24LC64: a 64 Kbit Electrically Erasable PROM

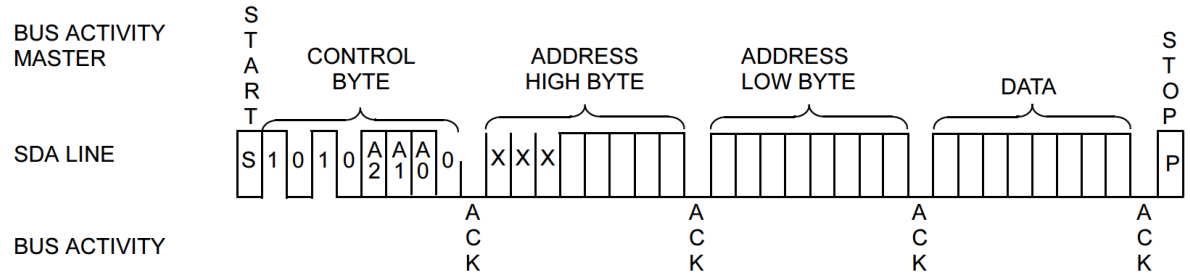


Control byte format



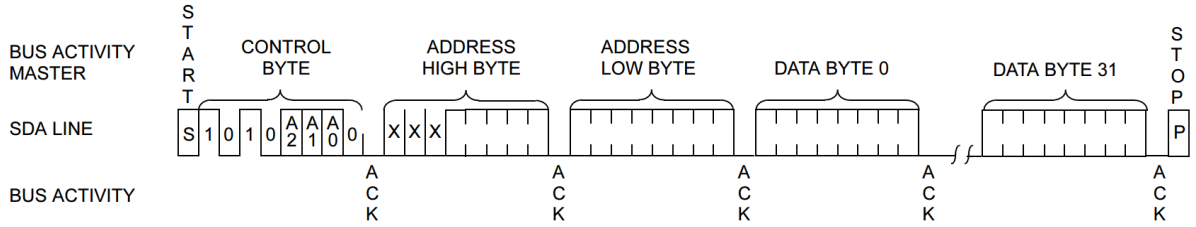
X = Don't Care Bit

ADDRESS SEQUENCE BIT ASSIGNMENTS



X = don't care bit

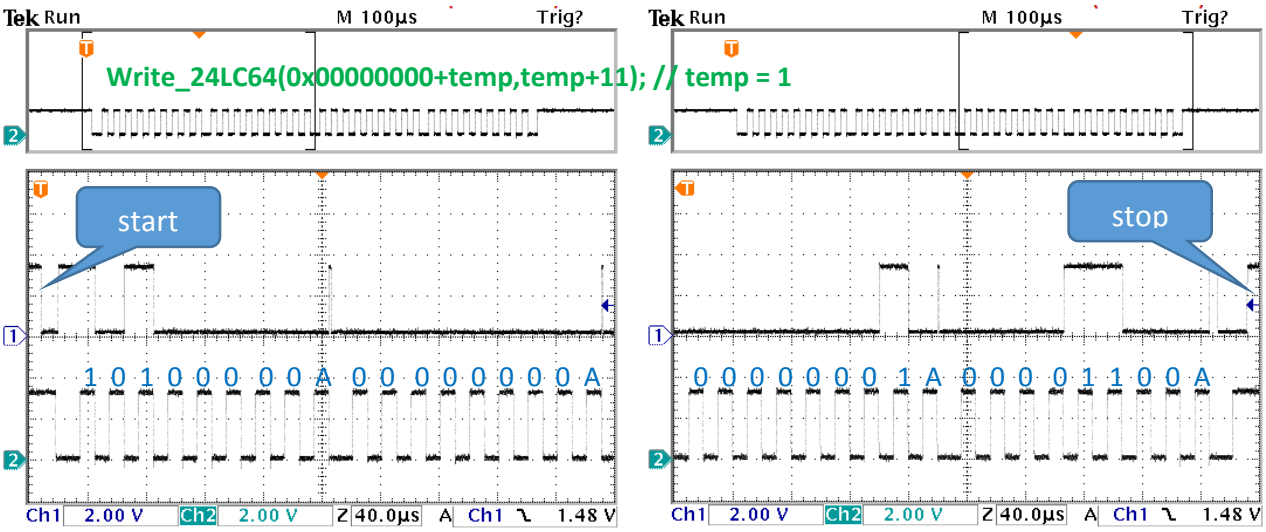
BYTE WRITE

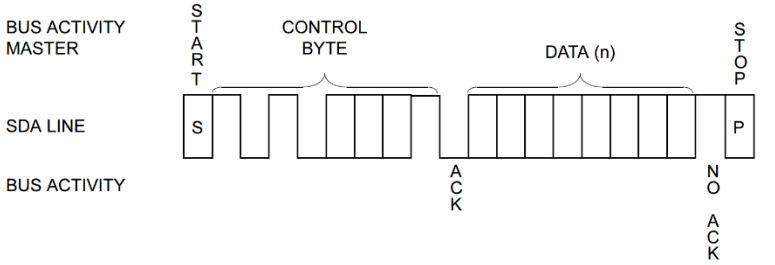


X = don't care bit

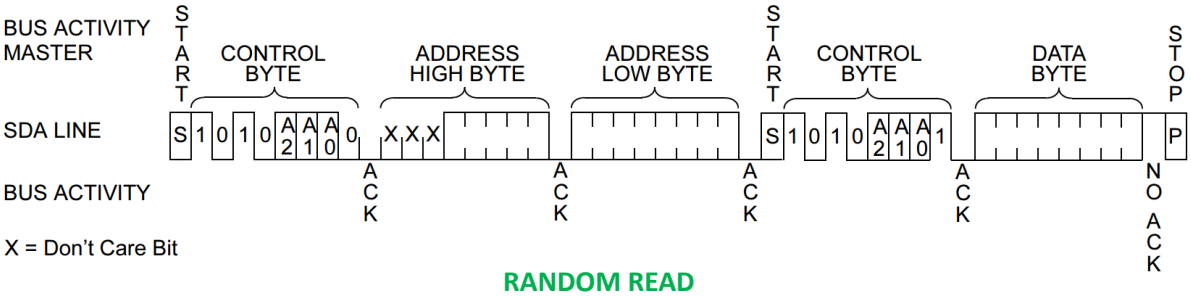
PAGE WRITE (up to 32 bytes or roll over)

```
13  void Write_24LC64(uint32_t address,uint8_t data )
14  {
15    uint32_t i;
16    SystemCoreClock = DrvSYS_GetHCLKFreq();
17    // Open I2C1 and set clock = 50Kbps
18    DrvI2C_Open(I2C_PORT1, 50000);
19
20    // send i2c start
21    DrvI2C_Ctrl(I2C_PORT1, 1, 0, 0, 0); // set start
22    while (I2C1->I2CON.SI == 0);        // poll si flag
23
24    // send writer command
25    I2C1->I2CDAT = 0XA0;                // control code with address (0)
26    DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 0); // clr si flag
27    while (I2C1->I2CON.SI == 0);        // poll si flag
28
29    // send address high
30    I2C1->I2CDAT = (address>>8)&0XFF;
31    DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 1); // clr si and set ack
32    while (I2C1->I2CON.SI == 0);        // poll si flag
33
34    // send address low
35    I2C1->I2CDAT = address&0XFF;
36    DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 1); // clr si and set ack
37    while (I2C1->I2CON.SI == 0);        // poll si flag
38
39    // send data
40    I2C1->I2CDAT = data;                // write data to
41    DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 1); // clr si and set ack
42    while (I2C1->I2CON.SI == 0);        // poll si flag
43
44    // send i2c stop
45    DrvI2C_Ctrl(I2C_PORT1, 0, 1, 1, 0); // send stop
46    while( I2C1->I2CON.STO);            /* if a STOP condition is detected
47                        this flag will be cleared by hardware automatically. */
48    // while( I2C1->CON.SI == 0 );
49
50    for(i=0;i<60;i++);
51
52    DrvI2C_Close(I2C_PORT1);
53
54    for(i=0;i<6000;i++);
55    for(i=0;i<6000;i++);
56  }
```



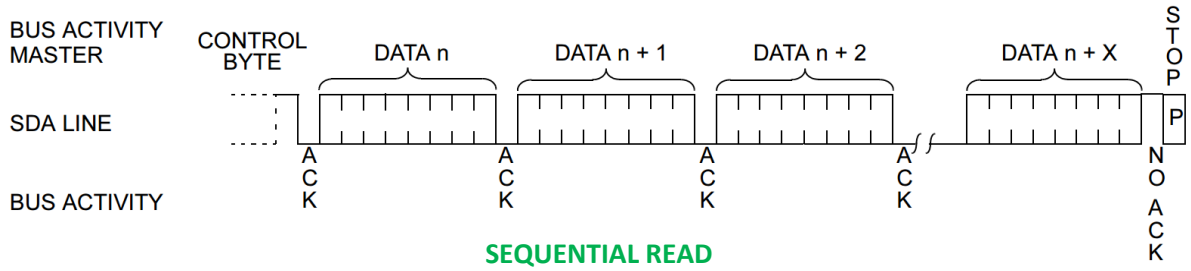


CURRENT ADDRESS READ



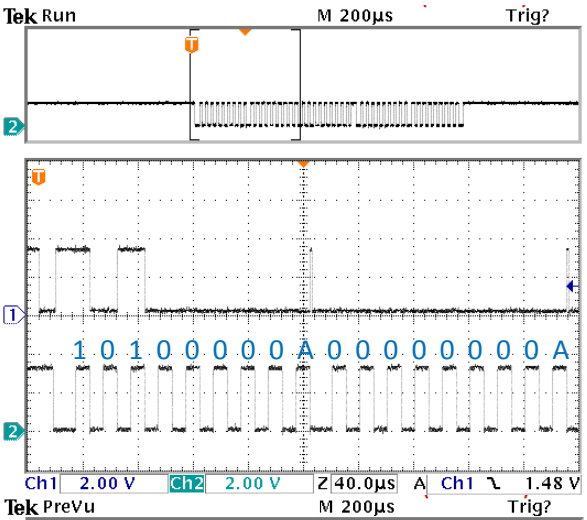
X = Don't Care Bit

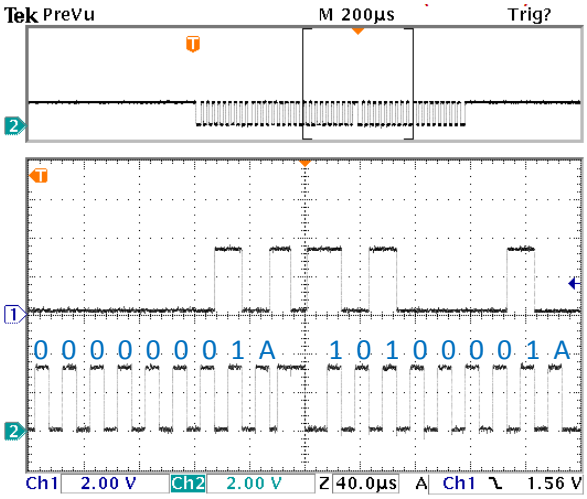
RANDOM READ

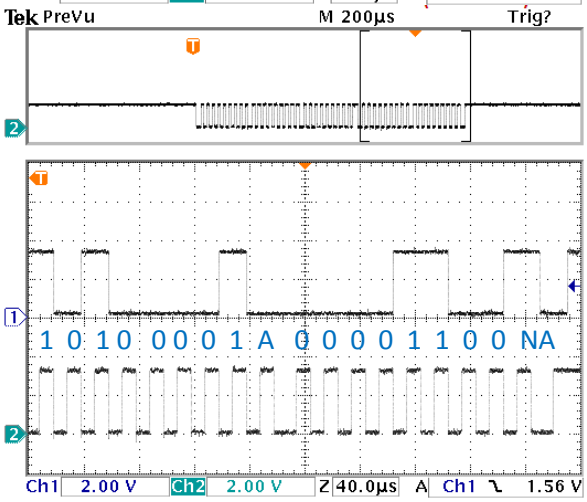


SEQUENTIAL READ

```
58  uint8_t Read_24LC64(uint32_t address)
59  {
60    uint8_t TEMP;
61    // Open I2C1 and set clock = 50Kbps
62    SystemCoreClock = DrvSYS_GetHCLKFreq();
63    DrvI2C_Open(I2C_PORT1, 50000);
64    // send i2c start
65    DrvI2C_Ctrl(I2C_PORT1, 1, 0, 1, 0); // set start
66    while (I2C1->I2CON.SI == 0);        // poll si flag
67
68    // send writer command
69    I2C1->I2CDAT = 0XA0;
70    DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 0); // clr si
71    while (I2C1->I2CON.SI == 0);        // poll si flag
72
73    // send address high
74    I2C1->I2CDAT = (address>>8)&0XFF;
75    DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 1); // clr si and set ack
76    while (I2C1->I2CON.SI == 0);        // poll si flag
77
78    // send address low
79    I2C1->I2CDAT = address&0XFF;
80    DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 0); // clr si and set ack
81    while (I2C1->I2CON.SI == 0);        // poll si flag
82
83    // send start flag
84    DrvI2C_Ctrl(I2C_PORT1, 1, 0, 1, 0); // clr si and send start
85    while (I2C1->I2CON.SI == 0);        // poll si flag
86
87    // send read command
88    I2C1->I2CDAT = 0XA1;
89    DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 1); // clr si
90    while (I2C1->I2CON.SI == 0);        // poll si flag
91
92    // resive data
93    I2C1->I2CDAT = 0XFF;
94    DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 0); // clr si
95    while (I2C1->I2CON.SI == 0);        // poll si flag
96    TEMP= I2C1->I2CDAT;
97
98    // send i2c stop
99    DrvI2C_Ctrl(I2C_PORT1, 0, 1, 1, 0); // clr si and set stop
100   while( I2C1->I2CON.STO);            /* if a STOP condition is detected
101                     this flag will be cleared by hardware automatically. */
102   DrvI2C_Close(I2C_PORT1);
103
104   return TEMP;
105 }
```







Read_24LC64(0x00000000+temp); // temp = 1

Procedure 1: 24LC64: GPA10-GPA11 connect to I2C Flash

1. Replace the content of the '**Smpl_Start_Kit.c**' with the '**24LC64'** lab file.
2. Compile the project, and run the program. (Add ScanKey.c, EEPROM_24LC64.c from "C:\Nuvoton\BSP Library\NUC100SeriesBSP_CMSIS_v1.05.003\NuvotonPlatform_Keil\Src\NUC1xx-LB_002\", and DrvI2C.c from "C:\Nuvoton\BSP Library\NUC100SeriesBSP_CMSIS_v1.05.003\NuvotonPlatform_Keil\Src\Driver\" to the project,)
3. Study the program and answer the following questions.

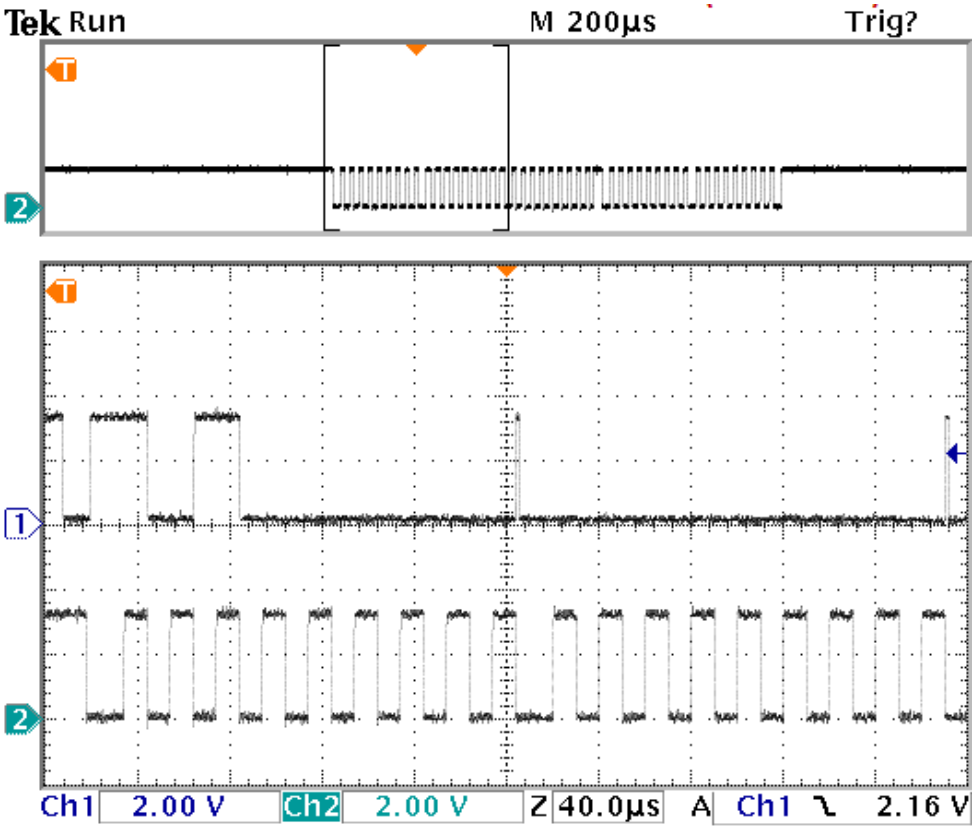
```
int main(void) {
  uint32_t i2cdata = 0, i;
  unsigned char temp;
  char addr[16] = "Address:";
  char Write[16] = "Write:";
  char read[16] = "Read:";

  UNLOCKREG();  /* Unlock the protected registers */
  DrvSYS_SetOscCtrl(E_SYS_XTL12M, 1); /* Enable the 12MHz oscillator */
  SysTimerDelay(5000); /* Waiting for 12M Xtal stalble */
  /* HCLK clock source. 0: external 12MHz; 4:internal 22MHz RC oscillator */
  DrvSYS_SelectHCLKSource(0);
  LOCKREG(); /*lock the protected registers */
  DrvSYS_SetClockDivider(E_SYS_HCLK_DIV, 0);
  /* HCLK clock frequency = HCLK clock source / (HCLK_N + 1) */

  Initial_pannel();  //call initial pannel function
  clr_all_pannal();
  print_lcd(0, "I2C with 24LC65");
  print_lcd(1, "test read and  ");
  print_lcd(2, "write function ");
  print_lcd(3, "press key1-key9");

  //initial key board
  for(i=0;i<6;i++)
    DrvGPIO_Open(E_GPA, i, E_IO_QUASI);

  DrvGPIO_InitFunction(E_FUNC_I2C1);

  while (1) {
    temp = Scankey();
    if (temp==1) {
      print_lcd(0,"Key1 had pressed ");
      print_lcd(1,"                 ");
      print_lcd(2,"                 ");
      print_lcd(3,"                 ");
      Write_24LC64(0x00000000+temp,temp+11);
      i2cdata= Read_24LC64(0x00000000+temp);
      sprintf(addr+8,"%x",temp);
      sprintf(Write+6,"%x",temp+11);
      sprintf(read+5,"%x",i2cdata);
      print_lcd(1,addr);
      print_lcd(2,Write);
      print_lcd(3,read);
      }
    if (temp==2) {
      print_lcd(0,"Key2 had pressed ");
      print_lcd(1,"                 ");
```

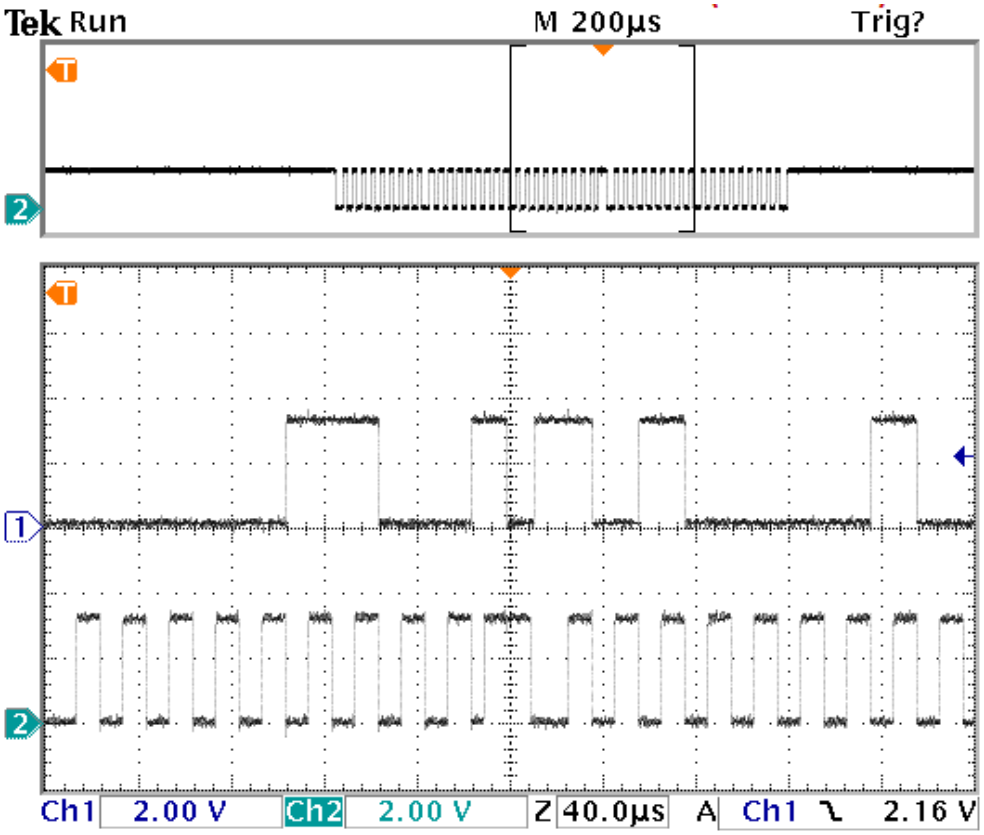
Questions (24LC64)

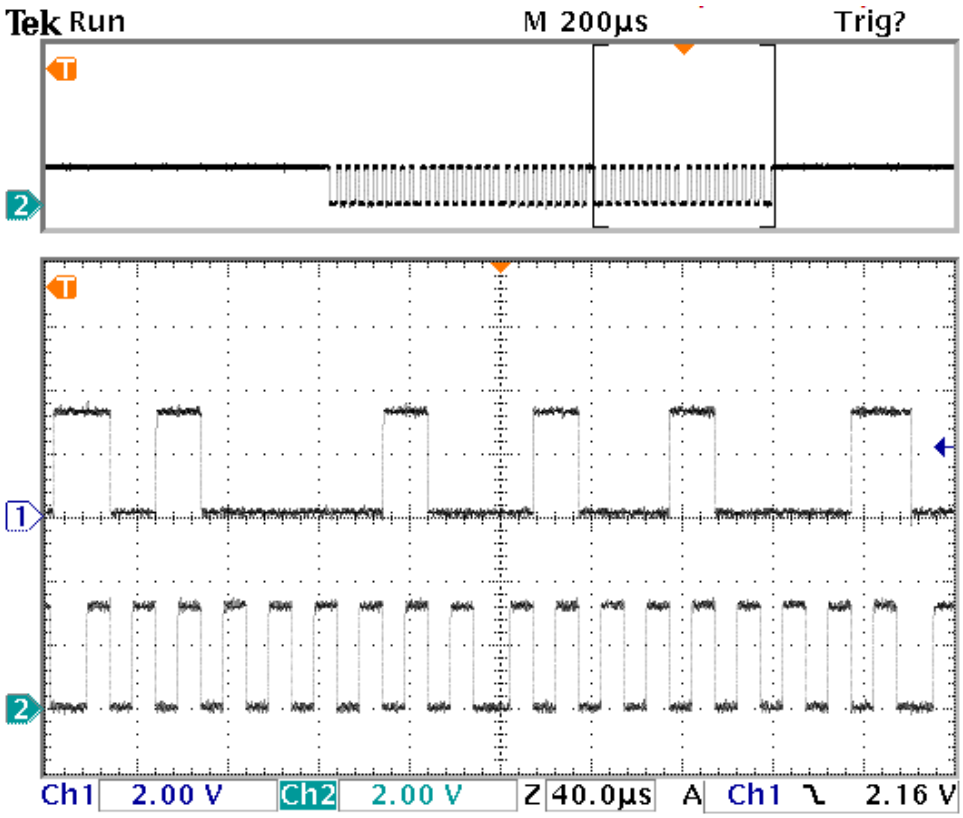
1. How long does it take to write one byte of data?

2. How long does it take to read one byte of data?

3. Label the signal to indicate start, stop, control byte, address, and data.







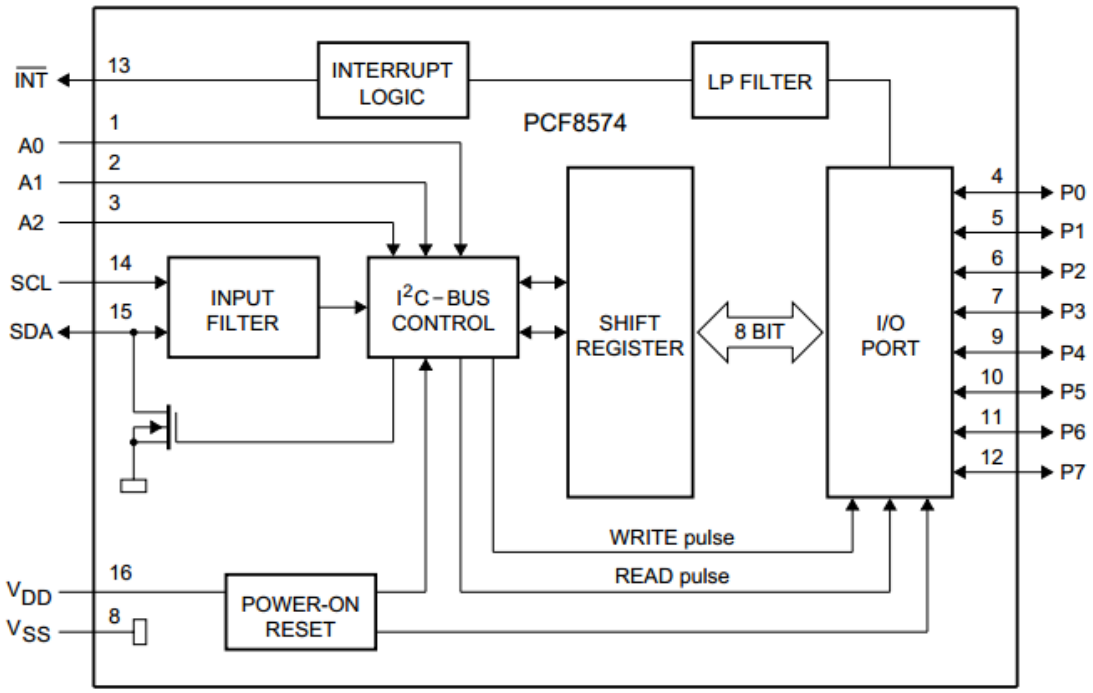
4. What is the maximum clock frequency can we use?

5. How long does it take to write one page of data (with the maximum frequency)?
   (Show how to get the result)

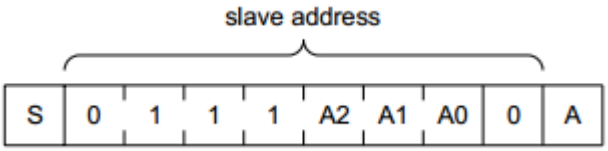
PCF8574A: Remote 8-bit I/O expander.

The device consists of an 8-bit quasi-bidirectional port and an I2C-bus interface. The PCF8574 has a low current consumption and includes latched outputs with high current drive capability for directly driving LEDs. It also possesses an interrupt line (INT) which can be connected to the interrupt logic of the microcontroller. By sending an interrupt signal on this line, the remote I/O can inform the microcontroller if there is incoming data on its ports without having to communicate via the I2C-bus. This means that the PCF8574 can remain a simple slave device.

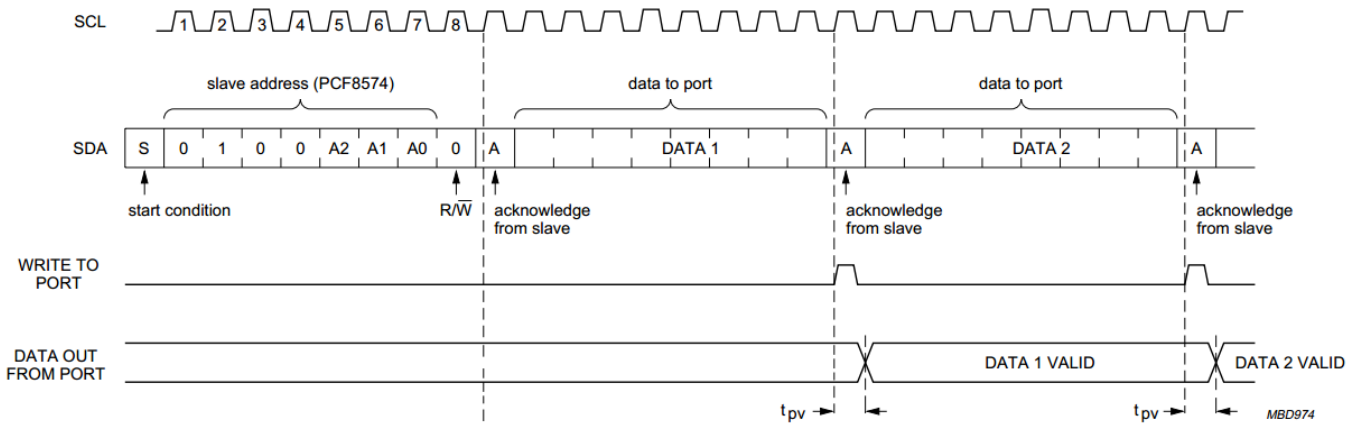
BLOCK DIAGRAM

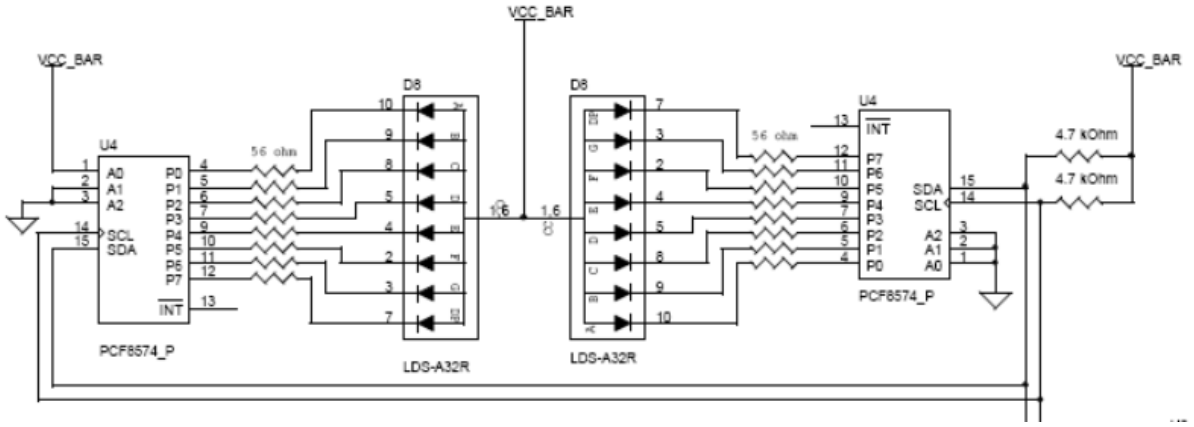


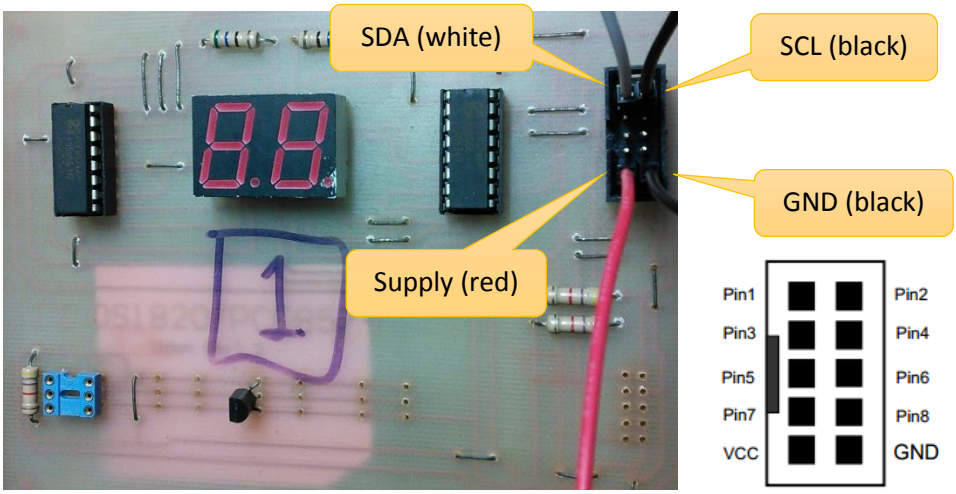
Addressing



WRITE mode (output).







Procedure 2: PCF8574A: Remote 8-bit I/O expander.

1. Replace the content of the '**Smpl_Start_Kit.c**' with the '**PCF8574A'** lab file.
2. Connect the PCF8574A board with the Nu_LB-002 learning board. (Connect 4 wires: SDA (white-wire: Pin1) to GPA10, SCL (short-black-wire: Pin2) to GPA11, Supply (red-wire: VCC) and GND (long-black-wire: GND).
3. Compile the project, and run the program.
4. Study the program and do the assignment in the class.

```
29  #define DELAY300ms  300000 // The maximal delay time is 335000 us.
30
31  uint8_t HEX2Disp(uint8_t hexNum) {
32    static const uint8_t lookUp[16] = {
33      0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8,
34      0x80, 0x90, 0x88, 0x83, 0xC6, 0xA1, 0x86, 0x8E
35      };
36    uint8_t hexDisp = lookUp[hexNum];
37    return hexDisp;
38    }
```

```
void Write_to_any8574(uint8_t i2c_addr, uint8_t data) {
  uint32_t i;
  SystemCoreClock = DrvSYS_GetHCLKFreq();
  //Open I2C1 and set clock = 50Kbps
  DrvI2C_Open(I2C_PORT1, 50000);

  //send i2c start
  DrvI2C_Ctrl(I2C_PORT1, 1, 0, 0, 0); // set start
  while (I2C1->I2CON.SI == 0);        // poll si flag

  //send writer command
  I2C1->I2CDAT = i2c_addr;            // send writer command to 8574
  DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 0); // clr si flag
  while (I2C1->I2CON.SI == 0);        // poll si flag

  //send data
  I2C1->I2CDAT = data;                // write data to
  DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 1); // clr si and set ack
  while (I2C1->I2CON.SI == 0);        // poll si flag

  //send i2c stop
  DrvI2C_Ctrl(I2C_PORT1, 0, 1, 1, 0); // send stop
  while (I2C1->I2CON.STO);            /* if a STOP condition is detected
                    this flag will be cleared by hardware automatically. */
  //while (I2C1->I2CON.SI == 0);      // poll si flag

  for(i=0;i<60;i++);
  DrvI2C_Close(I2C_PORT1);
  for(i=0;i<6000;i++);
  for(i=0;i<6000;i++);
  }

int main(void) {
  unsigned char temp, i;

  /* Unlock the protected registers */
  UNLOCKREG();
  /* Enable the 12MHz oscillator oscillation */
  DrvSYS_SetOscCtrl(E_SYS_XTL12M, 1);
  /* Waiting for 12M Xtal to stable */
  SysTimerDelay(5000);
  /* HCLK clock source. 0: external 12MHz; 4:internal 22MHz RC oscillator */
  DrvSYS_SelectHCLKSource(0);
  /*lock the protected registers */
  LOCKREG();
  /* HCLK clock frequency = HCLK clock source / (HCLK_N + 1) */
  DrvSYS_SetClockDivider(E_SYS_HCLK_DIV, 0);

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

  print_lcd(0, "I2C with       ");
  print_lcd(1, "       24LC65  ");
  print_lcd(2, "       PCF8574 ");
  print_lcd(3, "press key1-key4");

  // initial keyboard
  for(i=0;i<6;i++)
    DrvGPIO_Open(E_GPA, i, E_IO_QUASI);

  DrvGPIO_InitFunction(E_FUNC_I2C1);

  while(1) {
    temp=Scankey();
    if (temp == 1 ) {
      print_lcd(0, "Key1 had pressed ");
      Write_to_any8574(0x70, 1);
      }
    if (temp == 2) {
      print_lcd(0, "Key2 had pressed ");
      Write_to_any8574(0x70, 0xFF);
      }
    if (temp == 3) {
      print_lcd(0, "Key3 had pressed ");

      for(i=0;i<16;i++) {
        Write_to_any8574(0x72, HEX2Disp(i));
        DrvSYS_Delay(DELAY300ms);     // delay
        }
      }
    if (temp == 4) {
      print_lcd(0, "Key4 had pressed ");
      Write_to_any8574(0x72, 0xFF);
      }
    }
  }
```

Assignment(s)

Summarize what you suppose to learn in this class.