

# CPE329-03

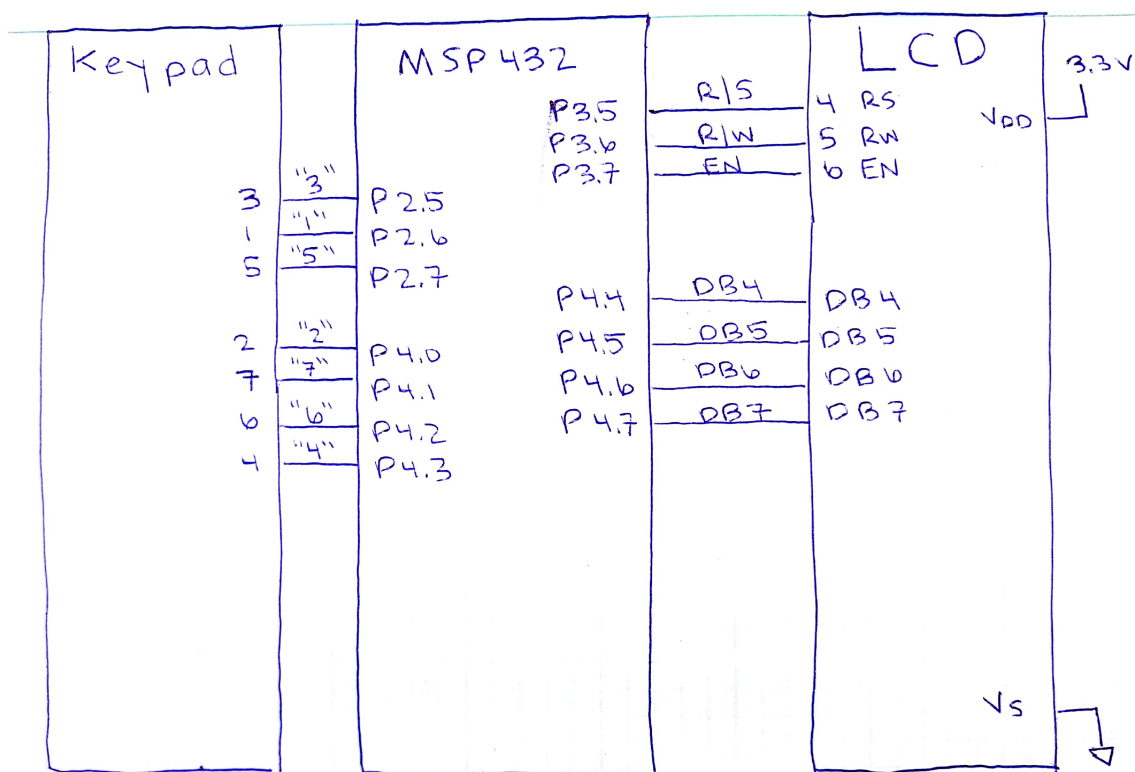
## A4 - Keypad

John Contovasilis  
Andes Le

April 17, 2018

1. <https://youtu.be/nYQmfk50dAk>

2. Schematic:



```

1 #include "msp.h"
2 #include "gpio.h"
3 #include "delay.h"
4 #include "lcd.h"
5 #include "keypad.h"
6
7 /**
8  * main.c
9  */
10
11 void updateDisplay();
12
13 void main(void)
14 {
15     set_DCO(FREQ_24_MHZ);           //set DCO to 24MHZ
16     lcdInit();                     //initialize display
17     initKeypad();                  //initialize keypad
18
19     while(1){
20         delay_ms(50, FREQ_24_MHZ);
21         updateDisplay();           //update display with most recent key press every 50
22         ms
23     }
24 }
25
26 void updateDisplay(){
27     sendCommand(0x00);             //clear display
28     sendCommand(0x10);
29     sendCharacter(key);             //send key character
30     sendCharacter(key<<4);
31     sendCommand(0x00);             //home display
32     sendCommand(0x20);
33 }

```

Listing 1: main.c source code

```

1  /*
   * keypad.h
3  *
   * Created on: Apr 16, 2018
5  * Author: Contovasilis
   */
7
9  #ifndef KEYPAD_H_
#define KEYPAD_H_

11 #include "msp.h"
#include "gpio.h"
13 #include <stdint.h>

15 #define R1_P          PORT4
#define R2_P          PORT4
17 #define R3_P          PORT4
#define R4_P          PORT4

19
21 #define R1            PIN0
#define R2            PIN1
23 #define R3            PIN2
#define R4            PIN3

25 #define C1_P          PORT2
#define C2_P          PORT2
27 #define C3_P          PORT2

29 #define C1            PIN5
#define C2            PIN6
31 #define C3            PIN7

33 #define RWPORT        R1_P | R2_P | R3_P | R4_P
#define CLPORT        C1_P | C2_P | C3_P
35 #define RWPINS        R1 | R2 | R3 | R4
#define CLPINS        C1 | C2 | C3

37
uint8_t initKeypad();
39 void scanKeypad(uint8_t iflg);

41 volatile uint8_t key;

43 #endif /* KEYPAD_H_ */

```

Listing 2: keypad.h source code

```

1  /*
   * keypad.c
3  *
   * Created on: Apr 16, 2018
5  * Author: Contovasilis
   */
7  #include "keypad.h"
   #include "interrupt.h"
9  #include <stdlib.h>

11 uint8_t initKeypad(){
    gpioStruct gpio;                //define struct
13    gpio.port = CLPORT;             //select column ports
    gpio.pins = CLPINS;             //select column pins
15    gpio.type = INPUT_PD;           //initialize GPIOs as inputs with pull-ups
    gpio_init(&gpio);               //configure struct

17    gpio.port = RWPORT;             //select row ports
19    gpio.pins = RWPINS;             //select row pins
    gpio.type = OUTPUT;             //initialize GPIOs as outputs
21    gpio_init(&gpio);               //configure struct

23    GPIO_set(RWPORT, RWPINS);       //set all row pins high

25    initExtInt(CLPORT, CLPINS, RISING); //initialize external interrupts on
                                       //column pins with rising edge trigger

27    enableExtInt(CLPORT, CLPINS);    //enable external interrupts
29    initNVIC(PORT2_IRQn);           //enable NVIC for column port

31    __enable_irq();                 //enable global interrupts

33 }

35 void scanKeypad(uint8_t iflg){
    uint8_t chrArray[12] = {'1', '4', '7', '*', '2', '5', '8', '0', '3', '6', '9', '#'};
37    uint8_t col = 0;                //array with keypad characters
                                       //initialize column iterator
39    while(col<4){                   //begin iteration loop
        GPIO_clear(RWPORT, (RWPINS) & ~(0x01<<col)); //set all except one row pin low
41        if(GPIO_read(CLPORT, CLPINS) & iflg){        //read column that produced the
                                                         //interrupt
            key = chrArray[col + ((iflg>>4) & ~(0x02))]; //set key to character pressed
            break;                                         //break if pressed key was found
45        }
        col++;                                           //repeat with next row set low
47        if
            GPIO_set(RWPORT, RWPINS);                   //pressed key was not found
                                                         //reset all rows high
49        }
        GPIO_set(RWPORT, RWPINS);                       //reset all rows high for next
51        key                                              //interrupt
    }

53 void PORT2_IRQHandler(void)        //ISR handler for port 2
55 {
    disableExtInt(CLPORT, CLPINS);    //disable column interrupts to avoid multiple
    interrupts
57    scanKeypad(readIFG(CLPORT, CLPINS)); //find which key in the row was pressed
    enableExtInt(CLPORT, CLPINS);     //re-enable column interrupts
59 }

```

Listing 3: keypad.c source code

```

2  /*
   * interrupt.h
   *
4  *   Created on: Apr 16, 2018
   *   Author: Contovasilis
6  */

8  #ifndef INTERRUPT_H_
#define INTERRUPT_H_

10 #include <stdint.h>

12 #define HAL_IES          (uint32_t)&P1->IES - (uint32_t)P1
14 #define HAL_IE          (uint32_t)&P1->IE - (uint32_t)P1
   #define HAL_IFG          (uint32_t)&P1->IFG - (uint32_t)P1
16 #define HAL_IV          (uint32_t)&P1->IV - (uint32_t)P1
   #define RISING          0
18 #define FALLING          1
   #define HWREG32(x)      (*((volatile uint32_t *) (x)))

20 void initExtInt(uint8_t selectedPort, uint32_t pins, uint8_t ies);
22 void enableExtInt(uint8_t selectedPort, uint32_t pins);
   void disableExtInt(uint8_t selectedPort, uint32_t pins);
24 void writeIFG(uint8_t selectedPort, uint32_t pins);
   uint8_t readIFG(uint8_t selectedPort, uint8_t mask);
26 void initNVIC(uint32_t irqNum);

28 #endif /* INTERRUPT_H_ */

```

Listing 4: interrupt.h source code

```

2  /*
3  * interrupt.c
4  *
5  * Created on: Apr 16, 2018
6  * Author: Contovasilis
7  */
8
9  #include "interrupt.h"
10 #include "msp.h"
11
12 static uint32_t GPIO_PORT_TO_BASE[] =
13 {
14     0x00,
15     (uint32_t)P1,
16     (uint32_t)P1+1,
17     (uint32_t)P3,
18     (uint32_t)P3+1,
19     (uint32_t)P5,
20     (uint32_t)P5+1,
21     (uint32_t)P7,
22     (uint32_t)P7+1,
23     (uint32_t)P9,
24     (uint32_t)P9+1
25 };
26
27 void initExtInt(uint8_t selectedPort, uint32_t pins, uint8_t ies){
28     uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];
29     if(ies){
30         HWREG32(baseAddress + HAL_IES) |= pins;
31     }
32     else{
33         HWREG32(baseAddress + HAL_IES) &= ~pins;
34     }
35 }
36
37 void enableExtInt(uint8_t selectedPort, uint32_t pins){
38     uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];
39     writeIFG(selectedPort, pins);
40     HWREG32(baseAddress + HAL_IE) |= pins;
41 }
42
43 void disableExtInt(uint8_t selectedPort, uint32_t pins){
44     uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];
45     HWREG32(baseAddress + HAL_IE) &= ~pins;
46 }
47
48 void writeIFG(uint8_t selectedPort, uint32_t pins){
49     uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];
50     HWREG32(baseAddress + HAL_IFG) &= ~pins;
51 }
52
53 uint8_t readIFG(uint8_t selectedPort, uint8_t mask){
54     uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];
55     return HWREG32(baseAddress + HAL_IFG) & mask;
56 }
57
58 void initNVIC(uint32_t irqNum){
59     NVIC->ISER[1] = 1<<((irqNum)&31);
60 }

```

Listing 5: interrupt.c source code

```

1  /*
   * gpio.h
3  *
   * Created on: Apr 9, 2018
   * Author: Contovasilis
   */
7
9  #ifndef GPIO_H_
#define GPIO_H_

11 #include <stdint.h>
#include "msp.h"
13 #include "lcd.h"

15 #define PIN0            0x0001
#define PIN1            0x0002
17 #define PIN2            0x0004
#define PIN3            0x0008
19 #define PIN4            0x0010
#define PIN5            0x0020
21 #define PIN6            0x0040
#define PIN7            0x0080

23
25 #define HAL_IN          (uint32_t)&P1->IN - (uint32_t)P1
#define HAL_OUT          (uint32_t)&P1->OUT - (uint32_t)P1
#define HAL_DIR          (uint32_t)&P1->DIR - (uint32_t)P1
27 #define HAL_REN          (uint32_t)&P1->REN - (uint32_t)P1
#define HAL_DS           (uint32_t)&P1->DS - (uint32_t)P1
29 #define HAL_SEL0        (uint32_t)&P1->SEL0 - (uint32_t)P1
#define HAL_SEL1        (uint32_t)&P1->SEL1 - (uint32_t)P1
31

33 #define HWREG16(x)      (*((volatile uint16_t *) (x)))

35
37 typedef enum GPIO_AF{
    AF1 = 0,
    AF2,
39    AF3
}GPIO_AF;

41
43 typedef enum GPIO_TYPE{
    OUTPUT=0,
    INPUT,
45    INPUT_PU,
    INPUT_PD,
47    AF
}GPIO_TYPE;

49
51 typedef enum GPIO_PORT{
    PORT1=1,
    PORT2,
53    PORT3,
    PORT4,
55    PORT5,
    PORT6
57 }GPIO_PORT;

59 typedef struct gpioStruct{
    GPIO_PORT port;
61    uint32_t pins;
    GPIO_TYPE type;
63    GPIO_AF af;
}gpioStruct;

65
67 void gpio_init(gpioStruct* gpio);
void GPIO_set(uint_fast8_t selectedPort, uint_fast16_t pins);
void GPIO_clear(uint_fast8_t selectedPort, uint_fast16_t pins);
69 void GPIO_toggle(uint_fast8_t selectedPort, uint_fast16_t pins);
uint8_t GPIO_read(uint_fast8_t selectedPort, uint_fast16_t pins);
71

#endif /* GPIO_H_ */

```

Listing 6: gpio.h source code

```

2  /*
3  * gpio.c
4  *
5  * Created on: Apr 9, 2018
6  * Author: Contovasilis
7  */
8
9  #include "gpio.h"
10
11 static uint32_t GPIO_PORT_TO_BASE[] =
12 {
13     0x00,
14     (uint32_t)P1,
15     (uint32_t)P1+1,
16     (uint32_t)P3,
17     (uint32_t)P3+1,
18     (uint32_t)P5,
19     (uint32_t)P5+1,
20     (uint32_t)P7,
21     (uint32_t)P7+1,
22     (uint32_t)P9,
23     (uint32_t)P9+1
24 };
25
26 void GPIO_set(uint_fast8_t selectedPort, uint_fast16_t pins){
27     uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];
28
29     HWREG16(baseAddress + HAL_OUT) |= pins;
30 }
31
32 void GPIO_clear(uint_fast8_t selectedPort, uint_fast16_t pins){
33     uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];
34
35     HWREG16(baseAddress + HAL_OUT) &= ~pins;
36 }
37
38 void GPIO_toggle(uint_fast8_t selectedPort, uint_fast16_t pins){
39     uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];
40
41     HWREG16(baseAddress + HAL_OUT) ^= pins;
42 }
43
44 uint8_t GPIO_read(uint_fast8_t selectedPort, uint_fast16_t pins){
45     uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];
46
47     return HWREG16(baseAddress + HAL_IN) & pins;
48 }
49
50 void gpio_init(gpioStruct* gpio){
51     uint32_t baseAddress = GPIO_PORT_TO_BASE[gpio -> port];
52     switch(gpio -> type){
53         case(OUTPUT):
54             HWREG16(baseAddress + HAL_SEL0) &= ~(gpio -> pins);
55             HWREG16(baseAddress + HAL_SEL1) &= ~(gpio -> pins);
56             HWREG16(baseAddress + HAL_DIR) |= gpio -> pins;
57             HWREG16(baseAddress + HAL_OUT) &= ~(gpio -> pins);
58             break;
59         case(INPUT):
60             HWREG16(baseAddress + HAL_SEL0) &= ~(gpio -> pins);
61             HWREG16(baseAddress + HAL_SEL1) &= ~(gpio -> pins);
62             HWREG16(baseAddress + HAL_DIR) &= ~(gpio -> pins);
63             HWREG16(baseAddress + HAL_REN) &= ~(gpio -> pins);
64             break;
65         case(INPUT_PU):
66             HWREG16(baseAddress + HAL_SEL0) &= ~(gpio -> pins);
67             HWREG16(baseAddress + HAL_SEL1) &= ~(gpio -> pins);
68             HWREG16(baseAddress + HAL_DIR) &= ~(gpio -> pins);
69             HWREG16(baseAddress + HAL_REN) |= gpio -> pins;
70             HWREG16(baseAddress + HAL_OUT) |= gpio-> pins;
71             break;
72         case(INPUT_PD):
73             HWREG16(baseAddress + HAL_SEL0) &= ~(gpio -> pins);
74             HWREG16(baseAddress + HAL_SEL1) &= ~(gpio -> pins);
75             HWREG16(baseAddress + HAL_DIR) &= ~(gpio -> pins);
76             HWREG16(baseAddress + HAL_REN) |= gpio -> pins;
77             HWREG16(baseAddress + HAL_OUT) &= ~(gpio -> pins);

```



```

78         break;
79     case (AF) :
80         switch(gpio -> af){
81             case (AF1):
82                 HWREG16(baseAddress + HAL_SEL0) &= gpio -> pins;
83                 HWREG16(baseAddress + HAL_SEL1) &= ~(gpio -> pins);
84                 break;
85             case (AF2):
86                 HWREG16(baseAddress + HAL_SEL0) &= ~(gpio -> pins);
87                 HWREG16(baseAddress + HAL_SEL1) &= gpio -> pins;
88                 break;
89             case (AF3):
90                 HWREG16(baseAddress + HAL_SEL0) &= gpio -> pins;
91                 HWREG16(baseAddress + HAL_SEL1) &= gpio -> pins;
92                 break;
93         }
94     }

```

Listing 7: gpio.c source code

```

1  /*
   * lcd.h
3  *
   * Created on: Apr 9, 2018
5  * Author: Contovasilis
   */
7
8  #ifndef LCD_H_
9  #define LCD_H_
11
12 #include <stdint.h>
13 #include <stdio.h>
14
15 #define CLEAR      0x01
16 #define HOME       0x02
17 #define ENTRY      0x04
18 #define DISPLAY    0x08
19 #define CURSOR     0x10
20 #define FUNCTION   0x20
21 #define CGRAM      0x40
22 #define DDRAM      0x80
23
24 #define DPORT      PORT4
25
26 #define DB0         PIN0
27 #define DB1         PIN1
28 #define DB2         PIN2
29 #define DB3         PIN3
30 #define DB4         PIN4
31 #define DB5         PIN5
32 #define DB6         PIN6
33 #define DB7         PIN7
34
35 #define CPORT       PORT3
36
37 #define ENA         PIN5
38 #define RW          PIN6
39 #define RS          PIN7
40
41 void sendCommand(uint8_t cmd);
42 void sendCharacter(uint8_t chr);
43 void sendString(char *str);
44 void sendInteger(int IntegerToDisplay);
45 void lcdInit();
46
47 #endif /* LCD_H_ */

```

Listing 8: lcd.h source code

```

1  /*
2   * lcd.c
3   *
4   * Created on: Apr 9, 2018
5   * Author: Contovasilis
6   */
7
8  #include "lcd.h"
9  #include "delay.h"
10 #include "gpio.h"
11
12
13 void lcdInit(){
14     delay_ms(50, FREQ_24_MHZ);    //startup delay
15
16     gpioStruct gpio;
17
18     gpio.port = CPORT;
19     gpio.pins = RS | RW | ENA;
20     gpio.type = OUTPUT;
21     gpio_init(&gpio);
22
23
24     gpio.port = DPORT;
25     gpio.pins = DB4 | DB5 | DB6 | DB7;
26     gpio.type = OUTPUT;
27     gpio_init(&gpio);
28
29     sendCommand(0x20);            //set LDC in 4 bit mode
30
31     sendCommand(0x20);            //5*8 Font, 1 line display
32     sendCommand(0x00);
33
34     sendCommand(0x00);            //Display ON, no cursor, no blink
35     sendCommand(0xC0);
36
37     sendCommand(0x00);            //Clear LCD
38     sendCommand(0x10);
39
40     sendCommand(0x00);            //Increment cursor, no shift
41     sendCommand(0x60);
42 }
43
44 void sendCommand(uint8_t cmd){
45     GPIO_clear(CPORT, RS);        //clear RS
46     GPIO_clear(CPORT, RW);        //clear RW
47     GPIO_set(CPORT, ENA);         //set ENA
48
49     GPIO_set(DPORT, cmd);         //send command byte or nibble
50     delay_ms(5, FREQ_24_MHZ);     //wait
51     GPIO_clear(CPORT, ENA);       //clear ENA
52     GPIO_clear(DPORT, 0xF0);     //clear data port
53 }
54
55
56 void sendCharacter(uint8_t chr){
57     GPIO_set(CPORT, RS);          //set RS
58     GPIO_clear(CPORT, RW);        //clear RW
59     GPIO_set(CPORT, ENA);         //set ENA
60
61     GPIO_set(DPORT, chr);         //send character byte or nibble
62     delay_ms(5, FREQ_24_MHZ);     //wait
63     GPIO_clear(CPORT, ENA);       //clear ENA
64     GPIO_clear(CPORT, RS);        //clear RS
65     GPIO_clear(DPORT, 0xF0);     //clear data port
66 }
67
68 void sendString(char *str)
69 {
70     while(*str > 0)
71     {
72         sendCharacter(*str);      //send first character
73         sendCharacter(*str<<4);
74         *str++;
75     }
76 }

```

```

77 void sendInteger(int IntegerToDisplay)
79 {
    char StringToDisplay[16];
81     sprintf(StringToDisplay, "%d", IntegerToDisplay);
    sendString(StringToDisplay);
83 }

```

Listing 9: lcd.c source code

```

1  /*
   * delay.h
3  *
   *   Created on: Apr 4, 2018
5  *   Author: Contovasilis
   */
7
9 #ifndef DELAY_H_
#define DELAY_H_
11
13 #include <stdint.h>
15
17 #define FREQ_1_5_MHZ      (uint16_t)15
#define FREQ_3_MHZ        (uint16_t)30
15 #define FREQ_6_MHZ      (uint16_t)60
#define FREQ_12_MHZ       (uint16_t)120
17 #define FREQ_24_MHZ      (uint16_t)240
#define FREQ_48_MHZ       (uint16_t)480
19
21 inline delay_ms(uint32_t time, uint16_t freq);
inline delay_us(uint32_t time, uint16_t freq);
23 void set_DCO(uint16_t freq);
25 #endif /* DELAY_H_ */

```

Listing 10: delay.h source code

```

1  /*
2   * delay.c
3   *
4   * Created on: Apr 4, 2018
5   * Author: Contovasilis
6   */
7  #include "delay.h"
8  #include "msp.h"
9
10 inline delay_ms(uint32_t time, uint16_t freq){
11     uint32_t ticks = freq * time * 10;
12
13     while(ticks){
14         ticks--;
15     }
16 }
17
18 inline delay_us(uint32_t time, uint16_t freq){
19     uint32_t ticks = ((freq * time) / 100) - 2;
20
21     while(ticks){
22         ticks--;
23     }
24 }
25
26 void set_DCO(uint16_t freq){
27
28     CS->KEY = CS_KEY_VAL;
29
30     switch(freq)
31     {
32         case FREQ_1_5_MHZ:
33             CS->CTL0 = CS_CTL0_DCORSEL_0;
34             break;
35         case FREQ_3_MHZ:
36             CS->CTL0 = CS_CTL0_DCORSEL_1;
37             break;
38         case FREQ_6_MHZ:
39             CS->CTL0 = CS_CTL0_DCORSEL_2;
40             break;
41         case FREQ_12_MHZ:
42             CS->CTL0 = CS_CTL0_DCORSEL_3;
43             break;
44         case FREQ_24_MHZ:
45             CS->CTL0 = CS_CTL0_DCORSEL_4;
46             break;
47         case FREQ_48_MHZ:
48             while ((PCM->CTL1 & PCM_CTL1_PMR_BUSY));
49             PCM->CTL0 = PCM_CTL0_KEY_VAL | PCM_CTL0_AMR_1;
50             while ((PCM->CTL1 & PCM_CTL1_PMR_BUSY));
51             FLCTL->BANK0_RDCTL = (FLCTL->BANK0_RDCTL & ~(FLCTL_BANK0_RDCTL_WAIT_MASK
52 )) | FLCTL_BANK0_RDCTL_WAIT_1;
53             FLCTL->BANK1_RDCTL = (FLCTL->BANK1_RDCTL & ~(FLCTL_BANK1_RDCTL_WAIT_MASK
54 )) | FLCTL_BANK1_RDCTL_WAIT_1;
55             CS->CTL0 = CS_CTL0_DCORSEL_5;
56             break;
57     }
58
59     CS->CTL1 = CS_CTL1_SEL__DCOCLK;
60     CS->KEY = 0;
61 }

```

Listing 11: delay.c source code