HCMC University of Technology and Education

Faculty of Electrical & Electronic Engineering

MICROPROCESSOR in PRACTICE

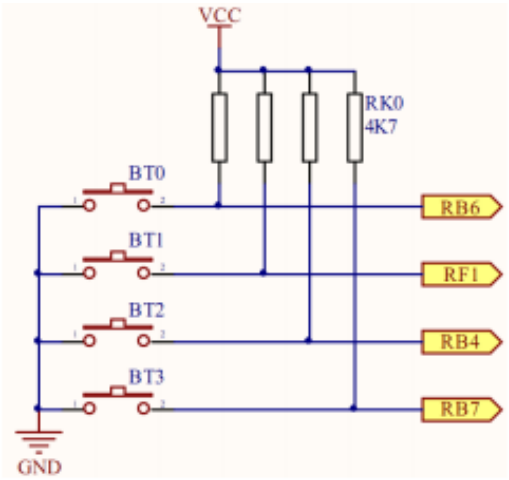
LAB 2

Control 32 LEDs Using Buttons

Instructor: NGO BA VIET, PhD

HCMC 8/2024

1. **Circuit diagram for interfacing a microcontroller with 4 buttons**



Buttons and keypads are used for interfacing between humans and electronic circuits to control them. For example, a computer keyboard, a phone keypad, or the keypad on a fuel dispenser for entering the amount of money or liters to be sold, and a washing machine with a keypad to adjust the washing mode or select the water level, etc.

2. **Define the names of the buttons in the library**

```
#define  bt0     pin_b6
#define  bt1     pin_f1
#define  bt2     pin_b4
#define  bt3     pin_b7

#define  on      bt0
#define  off     bt1
#define  inv     bt2

#define  up      bt0
#define  mod     bt1
#define  dw      bt2
#define  clr     bt3

#define  stop    bt3
#define  on1     bt0
#define  off1    bt1
#define  on2     bt2
#define  off2    bt3
```

3. **Exercise**

Exercise 321: Program to control 8 LEDs using 2 buttons, ON and OFF. When powered on, the LEDs are off. When the ON button is pressed, all 8 LEDs turn on, and when the OFF button is pressed, all 8 LEDs turn off. Save the file with the name 'bai_321_on_off_8led'.

```
#include <tv_kit_vdk_18f6722_all.c>
void main()
{
   set_up_port();
   while(true)
   {
      while(input(ON));
      xuat_32led_don_4byte(0,0,0,0xff);
      while(input(Off));
      xuat_32led_don_4byte(0,0,0,0);
   }
}
```

Refer to the method of writing the program using the IF statement:

```
#include <tv_kit_vdk_18f6722_all.c>
void main()
{
   set_up_port();
   xuat_32led_don_4byte(0,0,0,0);
   while(true)
   {
      if(!input(on))
      xuat_32led_don_4byte(0,0,0,0xff);

      else if(!input(off))
      xuat_32led_don_4byte(0,0,0,0);
    }
}
```

The difference between two programs using **while** and **if**: Both of these programs achieve the same control results, but if the control system needs to handle additional control requirements, the difference becomes apparent. Using the **while** statement will not be able to meet other requirements, whereas using the **if** statement can accommodate additional requirements. Depending on the needs, we can choose the appropriate method.

Exercise 322: Program to control 32 LEDs using 3 buttons: ON, OFF, and INV. When powered on, the 8 LEDs are off. When the ON button is pressed, 4 LEDs turn on. When the INV button is pressed, the 4 LEDs that are on will turn off, and the 4 LEDs that are off will turn on. When the OFF button is pressed, all 8 LEDs turn off. Save the file with the name 'bai_322_on_off_inv_8led'.

```
#include <tv_kit_vdk_18f6722_all.c>
unsigned int8 y;
void main()
{
   set_up_port();
   y=0;
   xuat_32led_don_4byte(0,0,0,0);
   while(true)
   {
      while(input(on));
      y=0x0f;
      xuat_32led_don_4byte(0,0,0,y);
      do
      {
         if (!input(inv))
         {
            y=~y;
            xuat_32led_don_4byte(0,0,0,y);
         }
      }while(input(off));
      xuat_32led_don_4byte(0,0,0,0);
    }
}
```

Exercise 323: Program to control 8 LEDs using 3 buttons: ON, OFF, and INV. When powered on, all 8 LEDs are off. Pressing the ON button will turn on 4 LEDs. Pressing the OFF button will turn off all LEDs. Pressing the INV button will invert the state of the LEDs: the 4 LEDs that are on will turn off, and the 4 LEDs that are off will turn on. The program includes debounce handling and waits for the button to be released. Save the file with the name 'bai_323_on_off_inv_cd_8led'.

```
#include <tv_kit_vdk_18f6722_all.c>
unsigned int8 y;

void phim_inv()
{
   if (!input(inv))
   {
       delay_ms(20);
       {
         if (!input(inv))
         {
            y=~y;
            xuat_32led_don_4byte(0,0,0,y);
            while(!input(inv));
         }
      }
```

```
    }
}

void main()
{
    set_up_port();
    y=0x00;
    while(true)
    {
        while(input(on));
        y=0x0f;
        xuat_32led_don_4byte(0,0,0,y);
        do
        {
            phim_inv();
        }
        while(input(off));
        xuat_32led_don_4byte(0,0,0,0);
    }
}
```

Exercise 324: Use a microcontroller to interface with 16 LEDs and 3 buttons named UP, DW, and CLR. When powered on, all 16 LEDs are off. Pressing UP will make the LEDs gradually light up from right to left—each press will turn on one LED. Pressing DW will make the LEDs gradually turn off in the reverse direction. Pressing CLR will clear all the LEDs. Save the file with the name 'bai_324_up_dw_clr_16led'.

Exercise 325: Similar to Exercise 324, but the CLR button also has the function to reverse the direction of the LED lighting and turning off. For example, when UP is pressed, the LEDs light up gradually from right to left. After pressing CLR, pressing UP will make the LEDs light up one by one from left to right. Similarly, for the DW button. Save the file with the name 'bai_325_up_dw_clr_16led_dao_chieu'.

Exercise 326: Write a program to perform 3 tasks:

Task 0: All 32 LEDs are off.

Task 1: 32 LEDs light up and turn off gradually from right to left.

Task 2: 32 LEDs light up and turn off gradually from left to right.

Use the UP button (BT0) to switch between tasks in the order 0, 1, 2. Use the DW button (BT1) to switch between tasks in the order 2, 1, 0. By default, start with Task 0. Save the file with the name 'bai_326_32led_3y_cau_up_dw'.

```
#include <tv_kit_vdk_18f6722_all.c>
usi8 tt_ct;
void b308_32led_std_pst(usi16 dl)
{
   usi8 i; usi32 x;
   x = 0;
   for(i=0;i<32;i++)
   {
      x = (x << 1)+1;
      xuat_32led_don_1dw(x);
      delay_ms(dl);
   }
   for(i=0;i<32;i++)
   {
      x = (x << 1);
      xuat_32led_don_1dw(x);
      delay_ms(dl);
   }
}

void b309_32led_std_tsp(usi16 dl)
{
   usi8 i; usi32 x;
   x = 0;
   for(i=0;i<32;i++)
   {
      x = (x>>1) + 0x80000000;
      xuat_32led_don_1dw(x);
      delay_ms(dl);
   }
   for(i=0;i<32;i++)
   {
      x = (x>>1);
      xuat_32led_don_1dw(x);
      delay_ms(dl);
   }
}

void b326_32led_std_pst_tsp_tat_3bt()
{
   if(phim_bt0_c1(c_ktnp,0)==co_nhan)
   {
      if(tt_ct<2) tt_ct++;
   }
   if(phim_bt1_c1(c_ktnp,0)==co_nhan)
   {
      if(tt_ct>0) tt_ct--;
   }
   if(tt_ct==0) xuat_32led_don_1dw(0);
   if(tt_ct==1) b308_32led_std_pst(200);
   if(tt_ct==2) b309_32led_std_tsp(200);
}
```

```
void main()
{
   set_up_port();
   tt_ct=0;
   while(true)
   {
     b326_32led_std_pst_tsp_tat_3bt();
   }
}
```

Note: Observe the phenomenon of slow button response when performing control tasks using the for loop.

Observation: When the 32 LEDs are off, pressing button BT0 or BT1 will immediately execute the corresponding gradual light-up and turn-off program. However, when the program for gradually lighting up and turning off the 32 LEDs is running, pressing the button to select another task will only take effect when the current program cycle ends. Pressing the button during the execution has no effect.

Conclusion: The issue with this exercise is that the button response is too slow.

Cause: The execution cycle using the for loop for gradual light-up and turn-off is 64 × 200ms = 12,800ms = 12.8s, which causes slow button response. The following section will discuss ways to address these issues.

Key Scanning Function Library: In **Exercise 326**, the key scanning function phim_bt0_c1(c_ktnp,0) is used, which is located in the library file tv_03_18f6722_key."

- **The definitions for the states of a button**

```
#define co_nhan      1
#define khong_nhan   0
#define c_ktnp       1
#define k_ktnp       0
```

  If pressed, it returns 'pressed', equivalent to level '1'. If not pressed, it returns 'not pressed', equivalent to level '0'. If the button release is checked, it is equivalent to level '1'. If not, it is equivalent to level '0'.

- **Function 309:** Check the BT0 button, return the result of whether it is pressed or not

```
int1 phim_bt0_c1(int1 ktnp,usi16 dl)
{
   if(!input(bt0))
   {
      delay_ms(20);
      if(!input(bt0))
      {
         if(ktnp) while(!input(bt0));
         delay_ms(dl);
         return co_nhan;
      }
      return khong_nhan;
   }
  return khong_nhan;
}
```

- **Explanation:** This function is an extended debounce INV function from Exercise 323. It adds an optional attribute to choose whether to check the button release depending on the 'ktnp' variable and whether there is a delay. The first attribute is for handling the press and release functions, similar to the INV button in Exercise 323. The second attribute is used for the press and hold function to increase or decrease a variable, such as the volume button on a TV. Since the adjustment range is wide, using the press and hold mode is suitable for fine-tuning to the desired value. In this mode, button release checking should be ignored, and a delay must be added to adjust the change speed accordingly.

  The same applies to the buttons BT1, BT2, and BT3.

4. **Resolve issues for quick response**

In the previous lesson, it was analyzed that the slow response when pressing the button was due to control programs having for loops and having to wait for them to complete. There are many ways to address this, but this document presents the most optimal method: instead of using ‘for’ loops, the control programs for 32 LEDs fading in and out should use ‘if’ statements.

Exercise 327: Write a program to control 32 LEDs to fade in and out from right to left using only if statements and without loops. Save the file name as 'bai_327_32led_tat_std_pst_tsp_3btn_nhanh'.

- **Subroutine 'bai_327_tv':**

```
usi8 ttl,tt_ct;
usi32 x;
void h327_reset_tang_tcttd_if()
{
   ttl=0;   x=0;
}

void h327_32led_std_tsp_if()
{
   if(ttl<32)
   {
      x=(x>>1)+0x80000000;
      xuat_32led_don_1dw(x);
      ttl++;
   }
   else  if(ttl<64)
   {
      x=(x>>1);
      xuat_32led_don_1dw(x);
      ttl++;
    }
    else  h327_reset_tang_tcttd_if();
}

void h327_32led_std_pst_if()
{
   if(ttl<32)
   {
      x=(x<<1)+1;
      xuat_32led_don_1dw(x);
      ttl++;
   }
   else  if(ttl<64)
   {
      x=(x<<1);
      xuat_32led_don_1dw(x);
      ttl++;
    }
    else  h327_reset_tang_tcttd_if();
}
```

- **Main program:**

```
#include <tv_kit_vdk_18f6722_all.c>
#include <bai_327_tv.c>

void b327_32led_std_pst_tsp_tat_3bt_if()
{
   if(phim_bt0_c1(c_ktnp,0)==co_nhan)
   {
      if(tt_ct<2)
      {
         tt_ct++;
         h327_reset_tang_tcttd_if();
      }
   }
   if(phim_bt1_c1(c_ktnp,0)==co_nhan)
   {
      if(tt_ct>0)
      {
         tt_ct--;
         h327_reset_tang_tcttd_if();
      }
   }
}

void main()
{
   set_up_port();
   tt_ct=0;
   h327_reset_tang_tcttd_if();
   while(true)
   {
      b327_32led_std_pst_tsp_tat_3bt_if();
      if(tt_ct==0) xuat_32led_don_1dw(0);
      if(tt_ct==1) h327_32led_std_pst_if();
      if(tt_ct==2) h327_32led_std_tsp_if();
      delay_ms(200);
   }
}
```

Exercise 328: Similar to Exercise 327, but uses method 2 for checking button presses. Save the file name as 'bai_328_32led_3y_cau_up_dw_if_c2'.

```
#include <tv_kit_vdk_18f6722_all.c>
#include <bai_327_tv.c>

void b328_32led_std_pst_tsp_tat_3bt_if()
{
   if(phim_bt0_c2(2)==co_nhan)
   {
      if(tt_ct<2)
      {
         tt_ct++;
         h327_reset_tang_tcttd_if();
      }
   }
   if(phim_bt1_c2(2)==co_nhan)
   {
      if(tt_ct>0)
      {
         tt_ct--;
         h327_reset_tang_tcttd_if();
      }
   }
}

void main()
{
   set_up_port();
   tt_ct=0;
   h327_reset_tang_tcttd_if();
   while(true)
   {
      b328_32led_std_pst_tsp_tat_3bt_if();
      if(tt_ct==0) xuat_32led_don_1dw(0);
      if(tt_ct==1) h327_32led_std_pst_if();
      if(tt_ct==2) h327_32led_std_tsp_if();
      delay_ms(200);
    }
}
```