Switch PCB

Mentor: Mr. Gulfan Ahamad

Aim: To glow two particular set of LEDs according to the configuration of switches by creating a testing jig for Switch PCB

Walkthrough:

Set of LEDs: [O] [O] [O] [O] [O] [O] [O] [O]

POWER LOW MED HIGH TUB FLAT SMF LOCAL

These LEDs were to be controlled using a R5F10277 micro-controller having 24-pins. We would be using 8 output pins and one input pin for Analog to Digital Convertor. Let's declare the output pins for the LEDs first. The eight pins are connected to LEDs in the given manner:

```
#define LED_POWER (P2_bit.no1) // 1 POWER
#define LED_LOW (P0_bit.no0)   // 2 LOW
#define LED_MED (P0_bit.no1)   // 3 MED
#define LED_HIGH (P0_bit.no2)  // 4 HIGH
#define LED_TUB (P0_bit.no3)   // 5 TUB
#define LED_FLAT (P6_bit.no0)  // 6 FLAT
#define LED_SMF (P6_bit.no1)   // 7 SMF
#define LED_LOCAL (P2_bit.no0) // 8 LOCAL
```

Where `P2_bit.no1` means PIN21 of the controller.

After this initialization, for glowing a particular LED we need to send HIGH signal to that pin. For example, if PIN21 is set HIGH, POWER LED will glow.

[Θ] [O] [O] [O] [O] [O] [O] [O]

POWER LOW MED HIGH TUB FLAT SMF LOCAL

Here, Θ denotes glowing LED.

Working of Analog to Digital Convertor

ADC is used to analyse and differentiate between analog signals. It takes a value and returns a 10 bits value, according to the highest value. When analysing a set of signals, like from 0 to 5V, ADC assigns 1023 to 5V and 0 to 0. So, we obtain a least count of 0.0048V. The ANI17 pin is connected to the Switch PCB.

Important functions for the ADC

```
void R_ADC_Create(void);
void R_ADC_Start(void);
void R_ADC_Set_OperationOn(void);
void R_ADC_Get_Result(uint16_t * const buffer);
```

1. Create

```
void R_ADC_Create(void)
{
    ADCEN = 1U;  /* supply AD clock */
    ADM0 = _00_AD_ADM0_INITIALVALUE;  /* disable AD conversion and clear ADM0
register */
    ADMK = 1U;  /* disable INTAD interrupt */
    ADIF = 0U;  /* clear INTAD interrupt flag */
    /* Set INTAD low priority */
    ADPR1 = 1U;
    ADPR0 = 1U;
    /* The reset status of ADPC is analog input, so it's unnecessary to set. */
    /* Set ANI17 pin */
    PMC1 |= 0x02U;
    PM1 |= 0x02U;
    ADM0 = _08_AD_CONVERSION_CLOCK_32 | _00_AD_TIME_MODE_NORMAL_1 |
_00_AD_OPERMODE_SELECT;
    ADM1 = _00_AD_TRIGGER_SOFTWARE | _00_AD_CONVMODE_CONSELECT;
    ADM2 = _00_AD_POSITIVE_VDD | _00_AD_NEGATIVE_VSS | _00_AD_AREA_MODE_1 |
_00_AD_RESOLUTION_10BIT;
    ADUL = _FF_AD_ADUL_VALUE;
    ADLL = _00_AD_ADLL_VALUE;
    ADS = _11_AD_INPUT_CHANNEL_17;
}
```

2. Start

```
void R_ADC_Start(void)
{
    ADIF = 0U;  /* clear INTAD interrupt flag */
    ADMK = 0U;  /* enable INTAD interrupt */
    ADCS = 1U;  /* enable AD conversion */
}
```

3. Set Operation

```
void R_ADC_Set_OperationOn(void)
{
    ADCE = 1U;  /* enable AD comparator */
}
```

4. Get Result

```
void R_ADC_Get_Result(uint16_t * const buffer)
{
   *buffer = (uint16_t)(ADCR >> 6U);
}
```

Now, according to the values given in battery[][] matrix, LEDs will be glown

```
uint16_t battery[NO_OF_BATTERIES][NO_OF_CURRENT] = {

    /*H    M    L */
    {691, 851, 1023}, /*Tubular*/
    {506, 623, 750}, /*Flat*/
    {303, 374, 450}, /*SMF*/
    {166, 204, 246} /*Local*/

};
```

So, if ADC gives 691, the LED status would be:

[Θ] [O] [O] [Θ] [Θ] [O] [O] [O]

POWER LOW MED HIGH TUB FLAT SMF LOCAL

We will add a buffer of 10 to the values, so above configuration can be obtained at values between 691 ± 10.

```
#define SELECTION_HYS_COUNT 10   // defining buffer
```

Foe the logic of LEDs, if-else blocks would be used, and with the help of these led_test() function is created, which takes a 16 bit number and glow the appropriate LED.

```
void led_test(uint16_t x){
  WDTE = 0xACU; /* restart watchdog timer */

    if (x>=battery[0][0] - SELECTION_HYS_COUNT && x<=battery[0][0]+SELECTION_HYS_COUNT)
    {
        LED_TUB = 1;
        LED_FLAT = 0;
        LED_SMF = 0;
        LED_LOCAL = 0;
        LED_HIGH = 1;
        LED_MED = 0;
        LED_LOW = 0;
    }

    else if (x>=battery[1][0] - SELECTION_HYS_COUNT && x<=battery[1][0]+SELECTION_HYS_COUNT)
    {
        LED_TUB = 0;
        LED_FLAT = 1;
        LED_SMF = 0;
        LED_LOCAL = 0;
        LED_HIGH = 1;
        LED_MED = 0;
        LED_LOW = 0;
    }
    else if (x>=battery[2][0] - SELECTION_HYS_COUNT && x<=battery[2][0]+SELECTION_HYS_COUNT)
    {
        LED_TUB = 0;
        LED_FLAT = 0;
        LED_SMF = 1;
        LED_LOCAL = 0;
```

```
    LED_HIGH = 1;
    LED_MED = 0;
    LED_LOW = 0;
}

else if (x>=battery[3][0] - SELECTION_HYS_COUNT && x<=battery[3][0]+SELECTION_HYS_COUNT)
{
    LED_TUB = 0;
    LED_FLAT = 0;
    LED_SMF = 0;
    LED_LOCAL = 1;
    LED_HIGH = 1;
    LED_MED = 0;
    LED_LOW = 0;
}

else if (x>=battery[0][1] - SELECTION_HYS_COUNT && x<=battery[0][1]+SELECTION_HYS_COUNT)
{
    LED_TUB = 1;
    LED_FLAT = 0;
    LED_SMF = 0;
    LED_LOCAL = 0;
    LED_HIGH = 0;
    LED_MED = 1;
    LED_LOW = 0;
}

else if (x>=battery[1][1] - SELECTION_HYS_COUNT && x<=battery[1][1]+SELECTION_HYS_COUNT)
{
    LED_TUB = 0;
    LED_FLAT = 1;
    LED_SMF = 0;
    LED_LOCAL = 0;
    LED_HIGH = 0;
    LED_MED = 1;
    LED_LOW = 0;
}




else if (x>=battery[2][1] - SELECTION_HYS_COUNT && x<=battery[2][1]+SELECTION_HYS_COUNT)
{
    LED_TUB = 0;
    LED_FLAT = 0;
    LED_SMF = 1;
    LED_LOCAL = 0;
    LED_HIGH = 0;
    LED_MED = 1;
    LED_LOW = 0;
}

else if (x>=battery[3][1] - SELECTION_HYS_COUNT && x<=battery[3][1]+SELECTION_HYS_COUNT)
```

```
{
    LED_TUB = 0;
    LED_FLAT = 0;
    LED_SMF = 0;
    LED_LOCAL = 1;
    LED_HIGH = 0;
    LED_MED = 1;
    LED_LOW = 0;
}

else if (x>=battery[0][2] - SELECTION_HYS_COUNT && x<=battery[0][2]+SELECTION_HYS_COUNT)
{
    LED_TUB = 1;
    LED_FLAT = 0;
    LED_SMF = 0;
    LED_LOCAL = 0;
    LED_HIGH = 0;
    LED_MED = 0;
    LED_LOW = 1;
}

else if (x>=battery[1][2] - SELECTION_HYS_COUNT && x<=battery[1][2]+SELECTION_HYS_COUNT)
{
    LED_TUB = 0;
    LED_FLAT = 1;
    LED_SMF = 0;
    LED_LOCAL = 0;
    LED_HIGH = 0;
    LED_MED = 0;
    LED_LOW = 1;
}

else if (x>=battery[2][2] - SELECTION_HYS_COUNT && x<=battery[2][2]+SELECTION_HYS_COUNT)
{
    LED_TUB = 0;
    LED_FLAT = 0;
    LED_SMF = 1;
    LED_LOCAL = 0;
    LED_HIGH = 0;
    LED_MED = 0;
    LED_LOW = 1;
}

else if (x>=battery[3][2] - SELECTION_HYS_COUNT && x<=battery[3][2]+SELECTION_HYS_COUNT)
{
    LED_TUB = 0;
    LED_FLAT = 0;
    LED_SMF = 0;
    LED_LOCAL = 1;
    LED_HIGH = 0;
    LED_MED = 0;
    LED_LOW = 1;
}
```

```
    else
    {
        LED_TUB = 0;
        LED_FLAT = 0;
        LED_SMF = 0;
        LED_LOCAL = 0;
        LED_HIGH = 0;
        LED_MED = 0;
        LED_LOW = 0;
    }
}
```

Now, we just have to call these functions to get the output. As, this program would be running continuously, we have to use infinite loop.

```
void main(void)
{
    R_MAIN_UserInit();
    R_ADC_Set_OperationOn();
    while (1U)
    {
        WDTE = 0xACU; /* restart watchdog timer */
        out = R_ADC_Get_Conversion(_11_AD_INPUT_CHANNEL_17);
        led_test(out);
    }
}
```

Result: Testing jig V1.0 for Switch PCB was created and then further it was further improved by my team-mate.