**Inverter Mode Selection PCB (L107\_V5)**

Mentor: Mr. Gulfan Ahamad

Aim: To develop a Testing jig which would be used in the production line to check the working of Inverter Mode Selection PCB. In this Testing jig, we have used different LEDs for indication of different modes being selected on Selection PCB.

Walkthrough:

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

POWER LOW MED HIGH TUB FLAT SMF LOCAL

These LEDs are controlled using a R5F10277 micro-controller having 24-pins. We would be using 8 output pins for 8 selection modes LEDs, one input pin for Analog Input from Selection PCB and one for digital input pin from Selection PCB . 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)**

Basically an analogue to digital converter takes a snapshot of an analogue voltage at one instant in time and produces a digital output code which represents this analogue voltage. The number of binary digits, or bits used to represent this analogue voltage value depends on the resolution of an A/D converter . In this microcontroller, we have a 10 bits resolution ADC which assigns any analog signal in range from 0 to 5V a digital value from 1023 to 0 where 1023 indicates 5V and 0 indicates 0V. So, Here we get a least count of 0.0048V. We have used ANI17 pin here to read analog input.

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);

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;

}

Start

void R\_ADC\_Start(void)

{

ADIF = 0U; /\* clear INTAD interrupt flag \*/

ADMK = 0U; /\* enable INTAD interrupt \*/

ADCS = 1U; /\* enable AD conversion \*/

}

Set Operation

void R\_ADC\_Set\_OperationOn(void)

{

ADCE = 1U; /\* enable AD comparator \*/

}

Get Result

void R\_ADC\_Get\_Result(uint16\_t \* const buffer)

{

\*buffer = (uint16\_t)(ADCR >> 6U);

}

After making our ADC module ready for taking Analog input, we made a table of range of ADC values for different modes of operations. For this purpose, we have used a well working Selection PCB and read ADC values for different modes. We got a battery [][] matrix given below -:

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\*/

};

Here H,M,L represents High, medium and Low current charging respectively and Tabular, Flat, SMF, Local represents different batteries types.

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 variable

For 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);

}

}

Further, we have worked on solving the hardware issues evolving in the Testing Jig PCB while its working. It involves correcting the polarity of zener diodes in the circuit, identifying the issue of shorting in PCB circuit using a multimeter and resolving it etc.

I have worked with my team-mate Kuldeep to make this project done successfully.