Embedded software design task document

Delivered tasks:

- **Keypad & LCD design with non blocking function**
- **Keypad & LCD implementation code**
- **Implement pre-configuration and linking configuration file for the UART driver**

Represented by:
Hazem Ashraf

Keypad design with non blocking function

The design of the keypad function consists of two functions

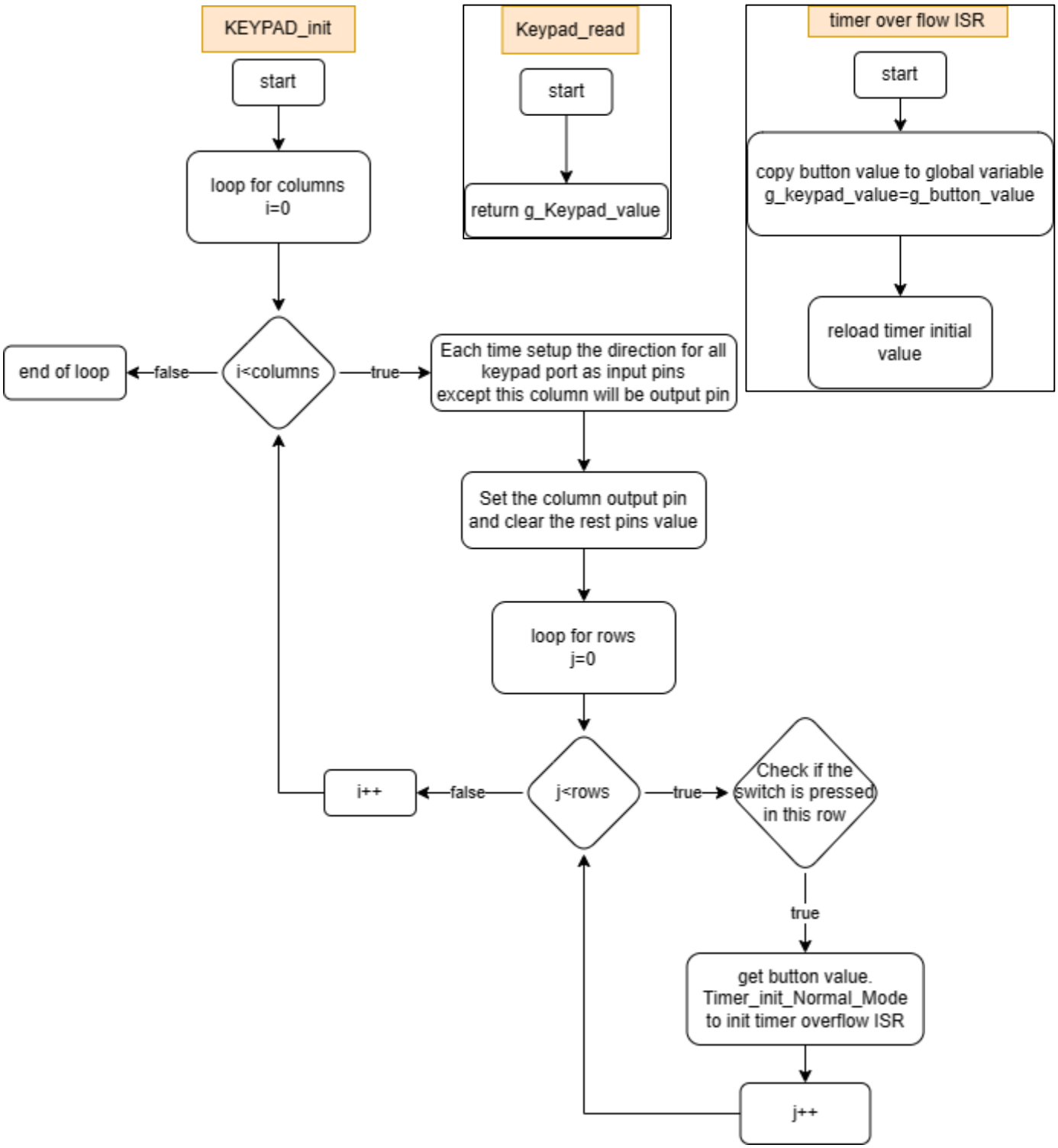
1- KEYPAD_init

Description:

Used to apply keypad button algorithm using timer overflow ISR

2- Keypad read

Used to return the value of pressed key



Keypad design implementation

The design of the keypad function consists of two functions

1- KEYPAD_init

Description:

Used to apply keypad button algorithm using timer overflow ISR

2- Keypad read

Used to return the value of pressed key

```
void  KEYPAD_getPressedKey(void)
{
uint8 col,row;
uint8 keypad_port_value = 0;

for(col=0;col<KEYPAD_NUM_COLS;col++) /* loop for columns */
{
/*
 * Each time setup the direction for all keypad port as input pins,
 * except this column will be output pin
 */
GPIO_setupPortDirection(KEYPAD_PORT_ID,PORT_INPUT);
GPIO_setupPinDirection(KEYPAD_PORT_ID,KEYPAD_FIRST_COLUMN_PIN_ID+col,PIN_OUTPUT
);
/* Set the column output pin and clear the rest pins value */
keypad_port_value = (1<<(KEYPAD_FIRST_COLUMN_PIN_ID+col));
GPIO_writePort(KEYPAD_PORT_ID,keypad_port_value);

for(row=0;row<KEYPAD_NUM_ROWS;row++) /* loop for rows */
{
/* Check if the switch is pressed in this row */
if(GPIO_readPin(KEYPAD_PORT_ID,row+KEYPAD_FIRST_ROW_PIN_ID)==KEYPAD_BUTTON_PRESSED)
{
g_button_value= KEYPAD_4x4_adjustKeyNumber((row*KEYPAD_NUM_COLS)+col+1);
Timer_init_Normal_Mode();
}
}
}
}


void Timer_init_Normal_Mode(void)
{
TCNT0 = 200;
TIMSK = (1<<TOIE0); // Enable Timer0 Overflow Interrupt
TCCR0 = (1<<FOC0) | (1<<CS02);
}
```

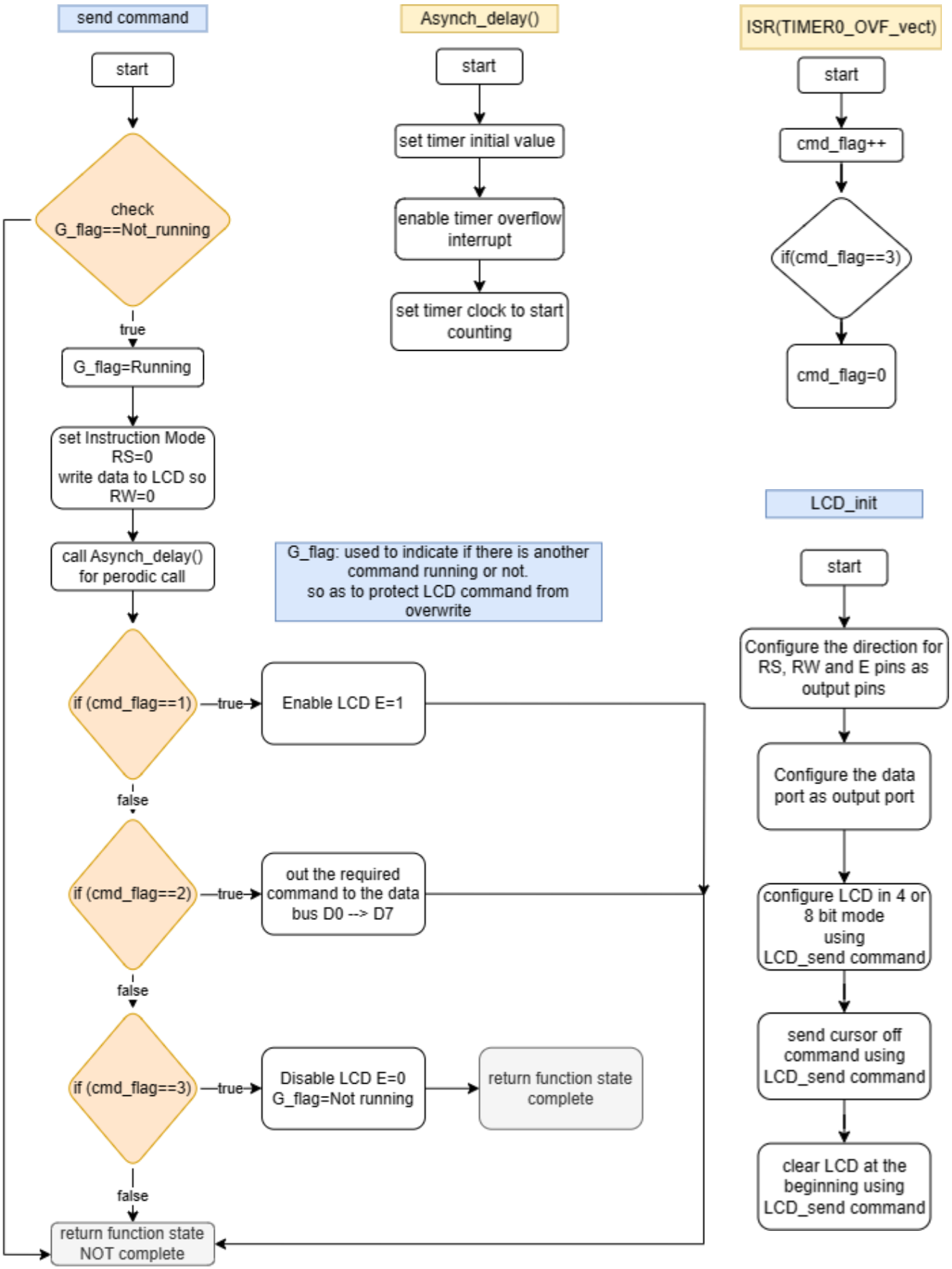
Keypad design with non blocking function

```
uint8 Keypad_read(void)
{
return g_keypad_value;
}


ISR(TIMER0_OVF_vect)
{
g_keypad_value=g_button_value;
TCNT0 = 200;
}
```

LCD design with non blocking function

The design of the LCD function consists of



UART configuration

Configuration pre compile Time

- Can be modified only before compilation
- Implemented using preprocessor instructions

File UART.h

```
/*
 * UART parity mode configuration ,its value should be 0(disabled)
 * or 1(even parity) or 2(odd parity)
 */
#define UART_parity_mode 1

#if (UART_parity_mode == 0)
#define UART_parity UART_parity_Disabled_mask
#elif (UART_parity_mode == 1)
#define UART_parity UART_parity_Even_mask
#elif (UART_parity_mode == 2)
#define UART_parity UART_parity_Odd_mask
#else
#error "Number of parity mode  should be equal to 0 or 1 or 2"
#endif
//define masking bits
#define UART_parity_Disabled_mask    0x00
#define UART_parity_Even_mask        0x20
#define UART_parity_Odd_mask         0x30
```

File UART.c

```
#include "UART.h"
void UART_init(uint32 baud_rate)
{
uint16 ubrr_value = 0;
UCSRB = (1<<RXEN) | (1<<TXEN);
UCSRC = (1<<URSEL) | (1<<UCSZ0) | (1<<UCSZ1);
UCSRC |= UART_mode ;
UCSRC |= UART_parity ;
#if(UART_mode == UART_Sync_mode_mask)
UCSRC |= UART_clk_polarity ;
#endif
UCSRC |= UART_stop_mode ;
/* Calculate the UBRR register value */
ubrr_value = (uint16)(((F_CPU / (baud_rate * 8UL))) - 1);
/* First 8 bits from the BAUD_PRESCALE inside UBRRL and last 4 bits in
UBRRH*/
UBRRH = ubrr_value>>8;
UBRRL = ubrr_value;
}
```

UART configuration

Configuration link compile Time

- Happens during linking stage compilation
- Implemented using structures and enum

File UART.h

```
typedef enum{
Disable,Even_Parity=2,Odd_Parity
}parity_mode;

typedef enum{
_1_bit,_2_bit
}stop_bit;

typedef enum{
bit_5,bit_6,bit_7,bit_8,bit_9=7
}bit_size;

typedef struct{
bit_size No_of_bits;
parity_mode partity_bit;
stop_bit No_of_stop_bit;
uint32 baud_rate;
}UART_configtype;
```

File UART.c

```
#include "UART.h"
void UART_init(UART_configtype *frame)
{
uint16 ubrr_value = 0;
UCSRB = (1<<RXEN) | (1<<TXEN);
UCSRC = (1<<URSEL);
UCSRC = (UCSRC & 0xC7)||(frame->No_of_stop_bit<<3)||(frame>partity_bit<<4);
if(frame->No_of_bits == bit_9)
{
UCSRC = (UCSRC & 0xF9) || ((frame->No_of_bits << 1) & 0xF7);
UCSRB = (UCSRB & 0xFB) || (1<<UCSZ2); //UCSZ2 bit combined with UCSZ1:0 for
9-bit data size mode
}
UCSRC = (UCSRC & 0xF9) || ((frame->No_of_bits) << 1);
ubrr_value = (uint16)(((F_CPU / (frame->baud_rate * 8UL))) - 1);
/* First 8 bits from the BAUD_PRESCALE inside UBRRL and last 4 bits in
UBRRH*/
UCSRC = (UCSRC & 0x7F);  //clear URSEL bit to when writing UBRRH
UBRRH = ubrr_value>>8;
UBRRL = ubrr_value;
}
```