FINAL MOCK PROJECT

THE DIGITAL CLOCK

Presenter : Anh-Tuan Mai

2024 – 10 – 22

EMB05_Module MCP

Outline

REQUIREMENTS

Requirements

Write program: The Digital Clock

SRS 1: After power on, display the time: 00-00-00 (date: 01.01.1971)

SRS 2: Press Button 1

- SRS 2-1: Display the date
- SRS 2-2: Display the time

SRS 3: Press Button 2

- SRS 3-1: Turn off the display mode
- SRS 3-2: Turn on the display mode

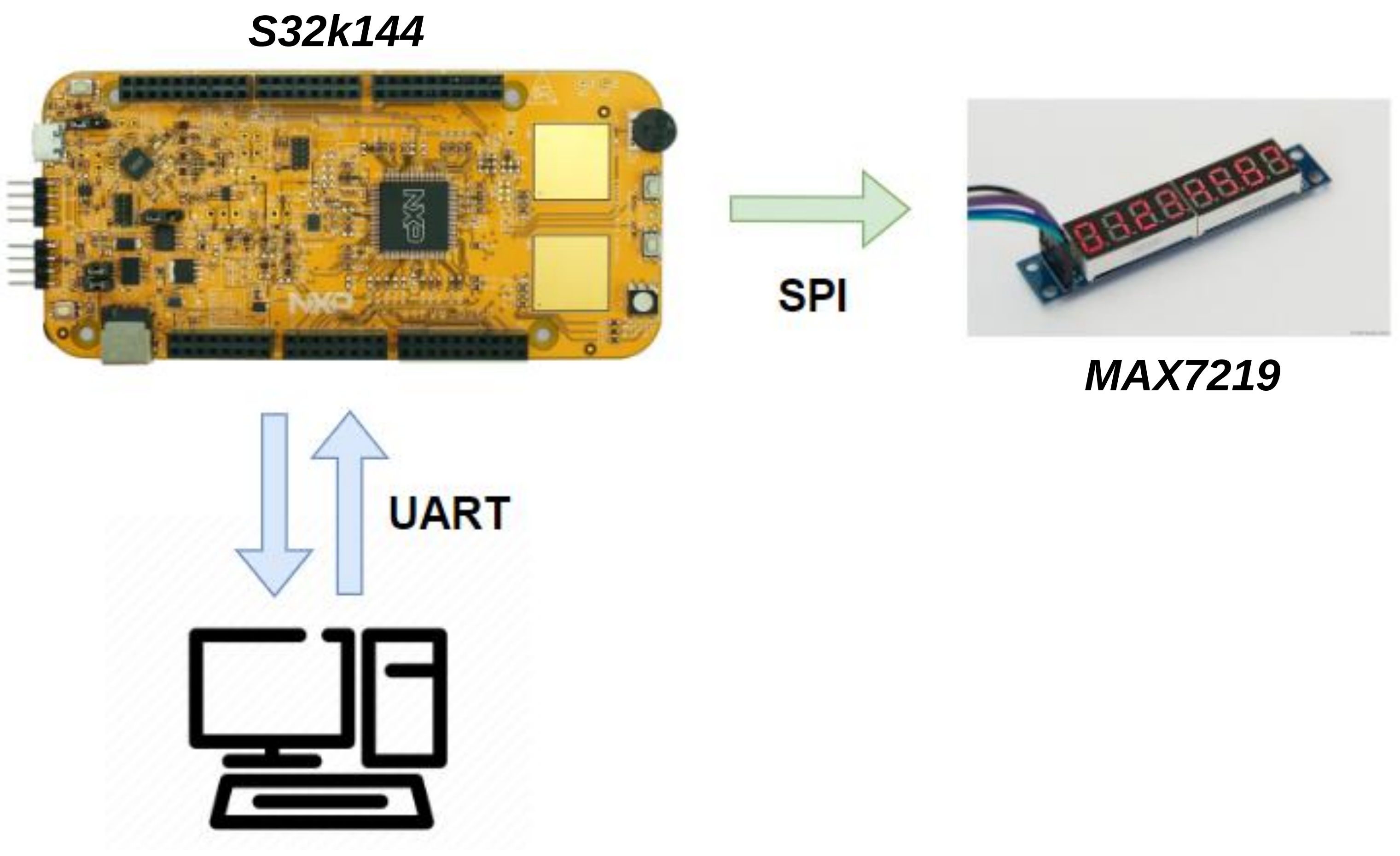
SRS4: Setting date, time by UART serial communication

SRS5: Use potentiometer to control the brightness of LED display(Opt)4

OVERALL SYSTEM

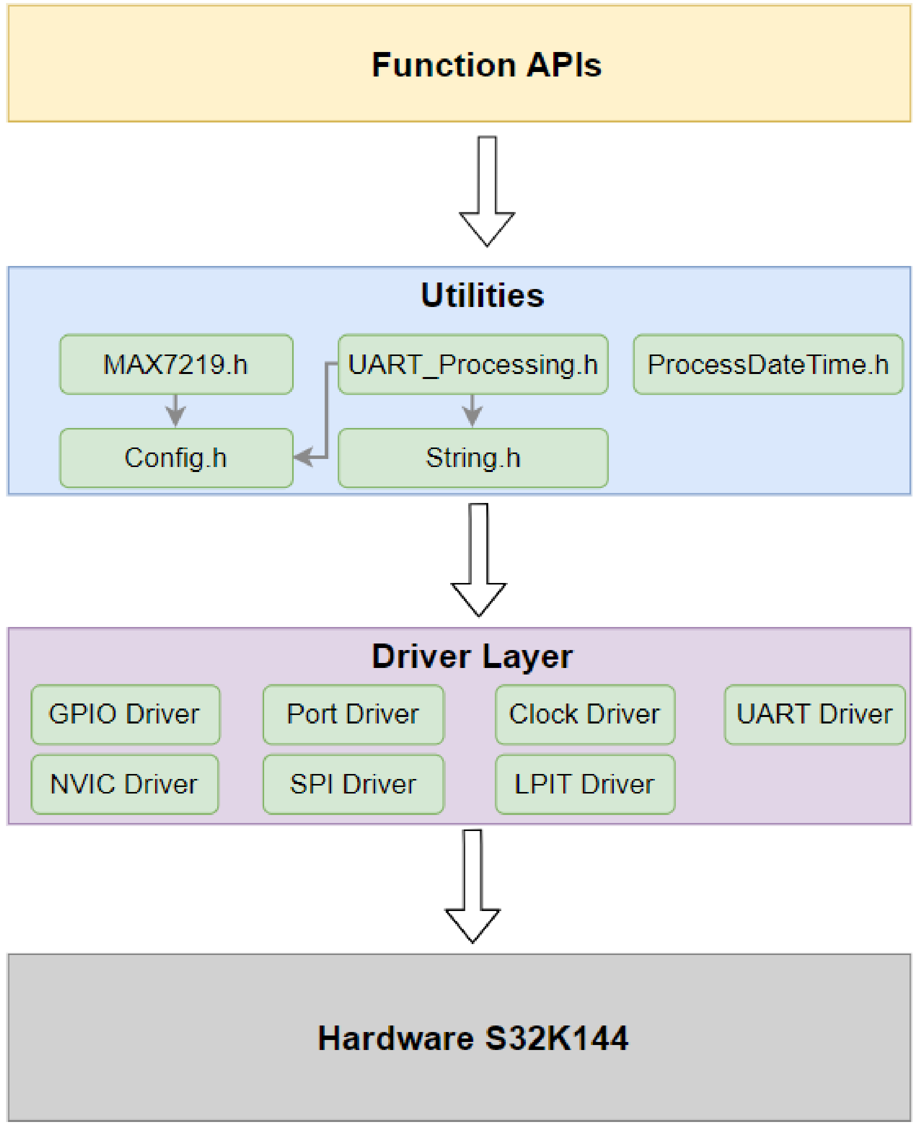
Overall System

Hardware Components



Overall System

Architecture design



Call functions in file main.c

String.h: functions process input string relating to length, format, compare string, convert string to number, split string by token, etc..

UART_Processing.h: functions receive input string, reset data, print string, check format, update date, update time

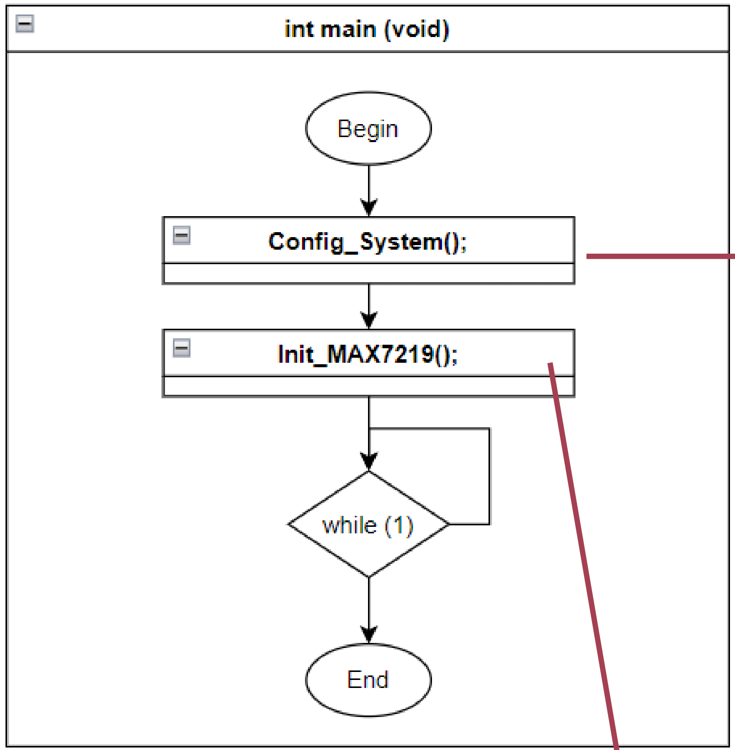
ProcessDateTime.h: functions check days in month, check leap year, process time & date.

Config.h: System Configuration for Peripherals

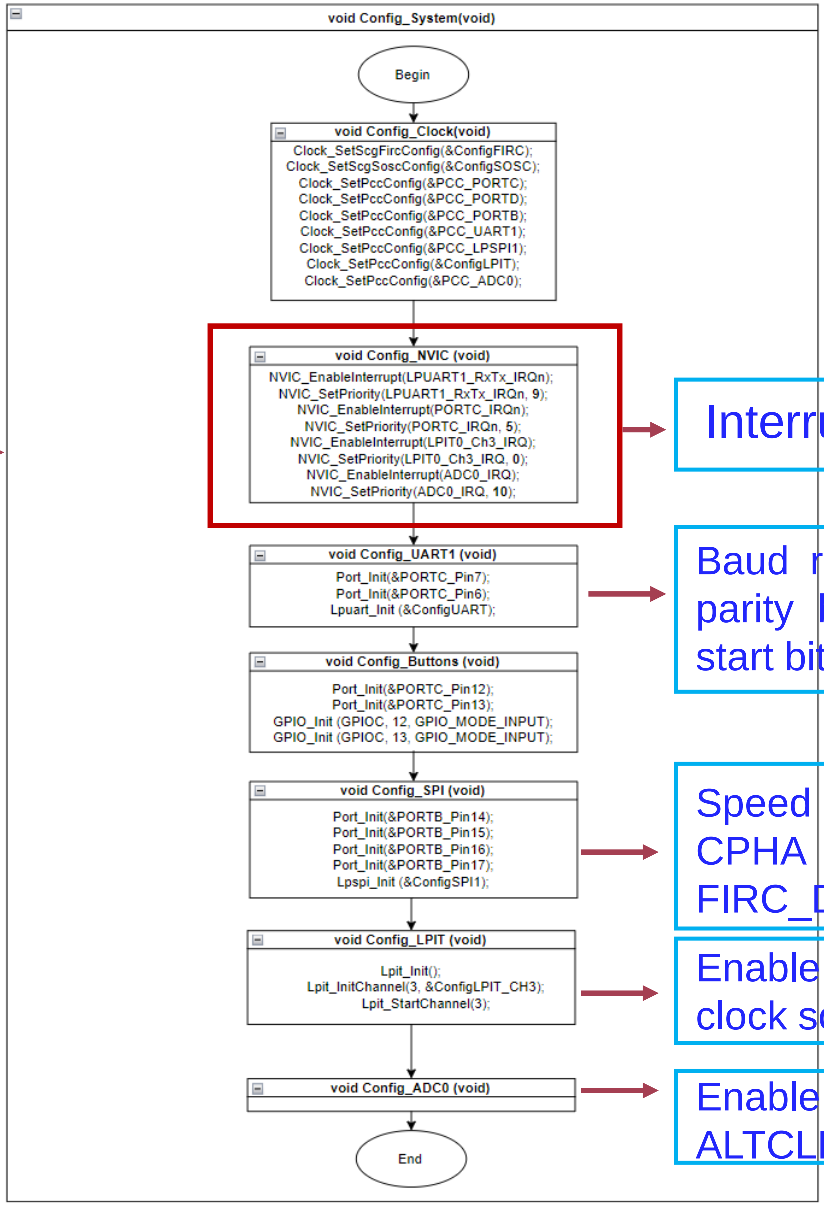
MAX7219.h: MAX7219 Driver Functions

DETAIL DESIGN

Detail design



NORMAL_MODE
NONE_DISPLAY_MODE
SCAN_LIMIT_ALL_DIGITS
DECODE_ALL_DIGITS



```
void Config_Clock(void)
Clock_SetScgFircConfig(&ConfigFIRC);
Clock_SetScgSoscConfig(&ConfigSOSC);
Clock_SetPccConfig(&PCC_PORTC);
Clock_SetPccConfig(&PCC_PORTD);
Clock_SetPccConfig(&PCC_PORTB);
Clock_SetPccConfig(&PCC_UART1);
Clock_SetPccConfig(&PCC_LPSPI1);
Clock_SetPccConfig(&ConfigLPIT);
Clock_SetPccConfig(&PCC_ADC0);

void Config_NVIC (void)
NVIC_EnableInterrupt(LPUART1_RxTx_IRQn);
NVIC_SetPriority(LPUART1_RxTx_IRQn, 9);
NVIC_EnableInterrupt(PORTC_IRQn);
NVIC_SetPriority(PORTC_IRQn, 5);
NVIC_EnableInterrupt(LPIT0_Ch3_IRQ);
NVIC_SetPriority(LPIT0_Ch3_IRQ, 0);
NVIC_EnableInterrupt(ADC0_IRQ);
NVIC_SetPriority(ADC0_IRQ, 10);

void Config_UART1 (void)
Port_Init(&PORTC_Pin7);
Port_Init(&PORTC_Pin6);
Lpuart_Init (&ConfigUART);

void Config_Buttons (void)
Port_Init(&PORTC_Pin12);
Port_Init(&PORTC_Pin13);
GPIO_Init (GPIOC, 12, GPIO_MODE_INPUT);
GPIO_Init (GPIOC, 13, GPIO_MODE_INPUT);

void Config_SPI (void)
Port_Init(&PORTB_Pin14);
Port_Init(&PORTB_Pin15);
Port_Init(&PORTB_Pin16);
Port_Init(&PORTB_Pin17);
Lpspi_Init (&ConfigSPI1);

void Config_LPIT (void)
Lpit_Init();
Lpit_InitChannel(3, &ConfigLPIT_CH3);
Lpit_StartChannel(3);

void Config_ADC0 (void)
```

Interrupt: ADC, LPIT, PORTC, UART

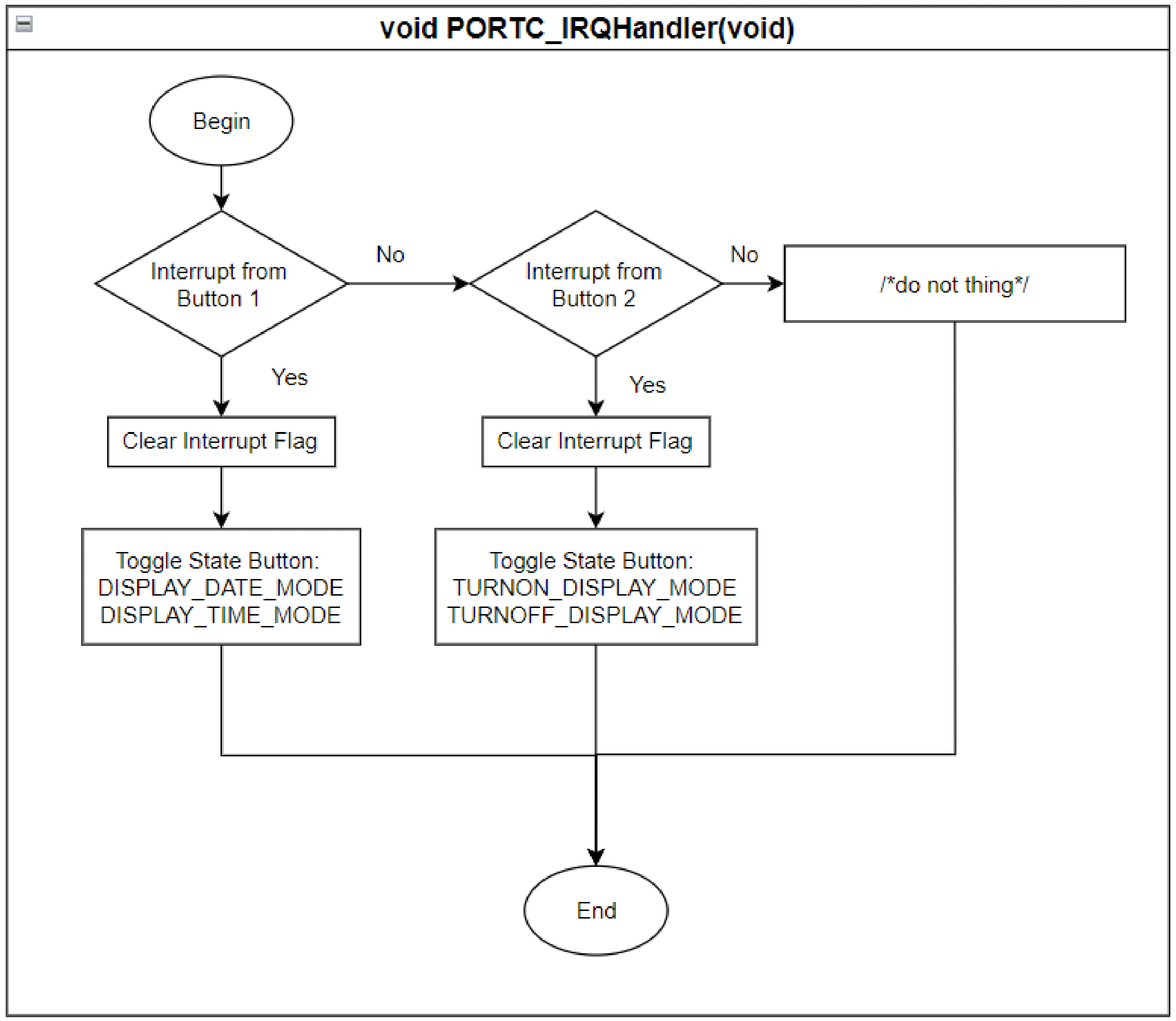
Baud rate 19200, interrupt RX, one stop bit, no parity bit, idle line with 8-character, count from start bit, clock source: FIRC_DIV2

Speed 1MHZ, 16 bits, chip select 3, CPOL = 0, CPHA = 0, MSB data transfer, clock source: FIRC_DIV2

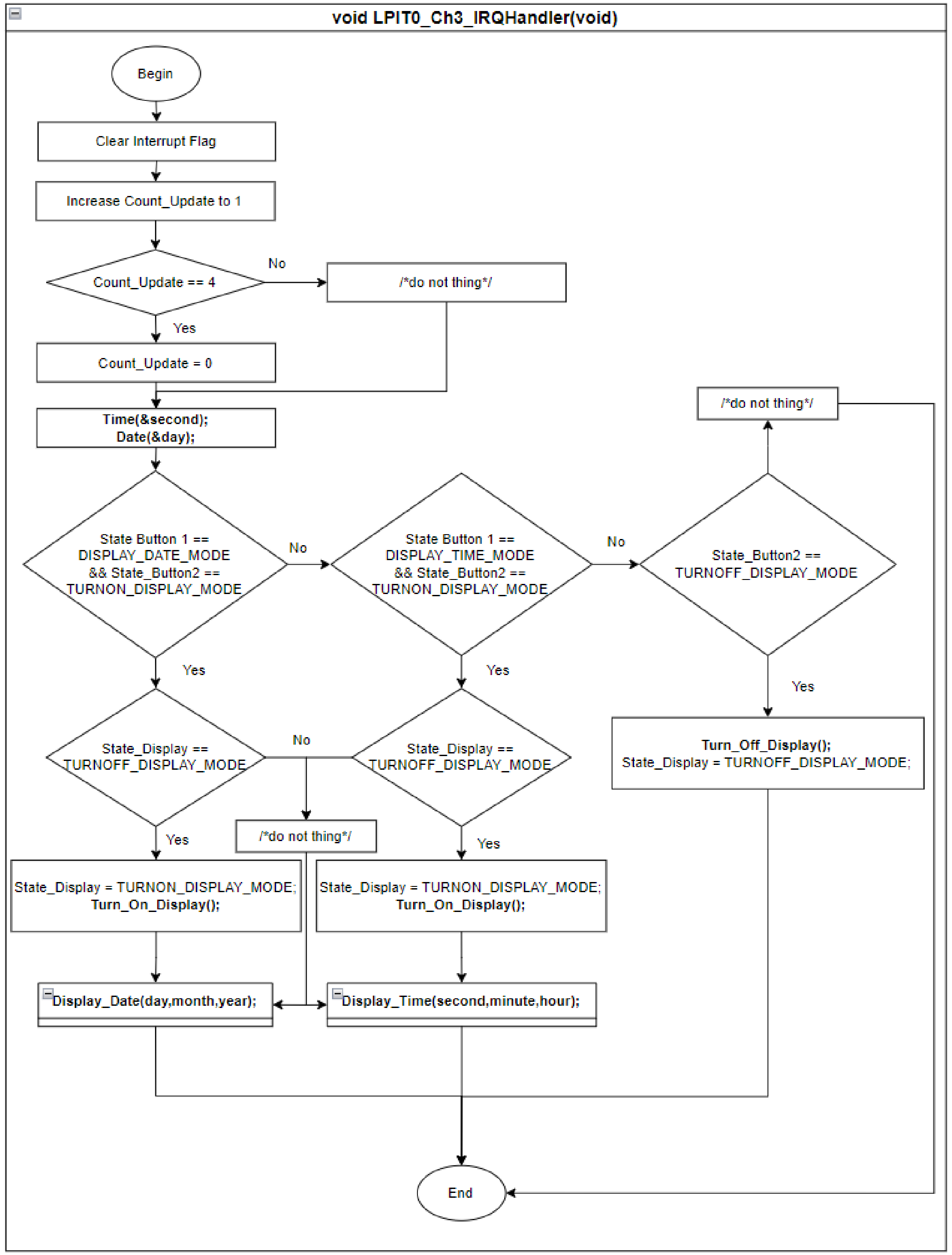
Enable interrupt, period = 250000 (~ T=250ms), clock source: SOSC_DIV2

Enable interrupt, clock source: FIRC_DIV2, ALTCLK1

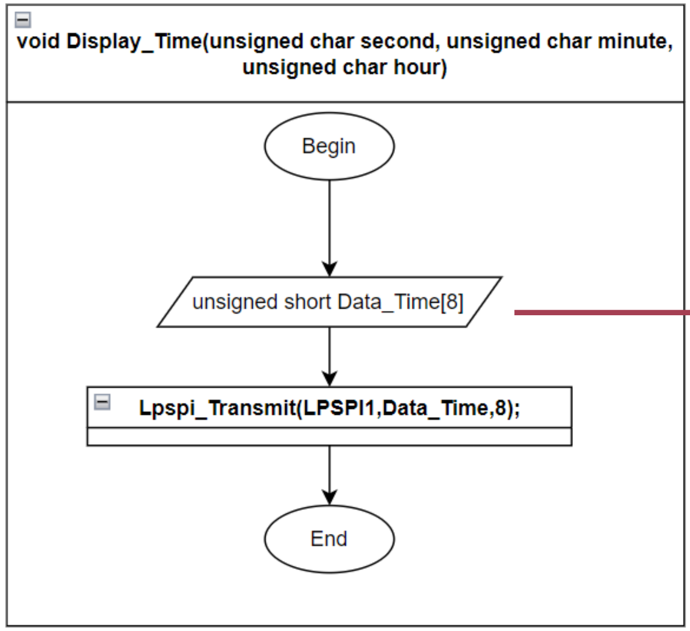
Detail design


void PORTC_IRQHandler(void)
Begin
Interrupt from Button 1
No
Interrupt from Button 2
No
/*do not thing*/
Yes
Yes
Clear Interrupt Flag
Clear Interrupt Flag
Toggle State Button: DISPLAY_DATE_MODE DISPLAY_TIME_MODE
Toggle State Button: TURNON_DISPLAY_MODE TURNOFF_DISPLAY_MODE
End

Detail design


void LPIT0_Ch3_IRQHandler(void)
Begin
Clear Interrupt Flag
Increase Count_Update to 1
Count_Update == 4
No
/*do not thing*/
Yes
Count_Update = 0
Time(&second);
Date(&day);
State Button 1 == DISPLAY_DATE_MODE && State_Button2 == TURNON_DISPLAY_MODE
No
State Button 1 == DISPLAY_TIME_MODE && State_Button2 == TURNON_DISPLAY_MODE
No
State_Button2 == TURNOFF_DISPLAY_MODE
/*do not thing*/
Yes
Yes
Yes
State_Display == TURNOFF_DISPLAY_MODE
No
State_Display == TURNOFF_DISPLAY_MODE
Turn_Off_Display();
State_Display = TURNOFF_DISPLAY_MODE;
Yes
/*do not thing*/
Yes
State_Display = TURNON_DISPLAY_MODE;
Turn_On_Display();
State_Display = TURNON_DISPLAY_MODE;
Turn_On_Display();
Display_Date(day,month,year);
Display_Time(second,minute,hour);
End

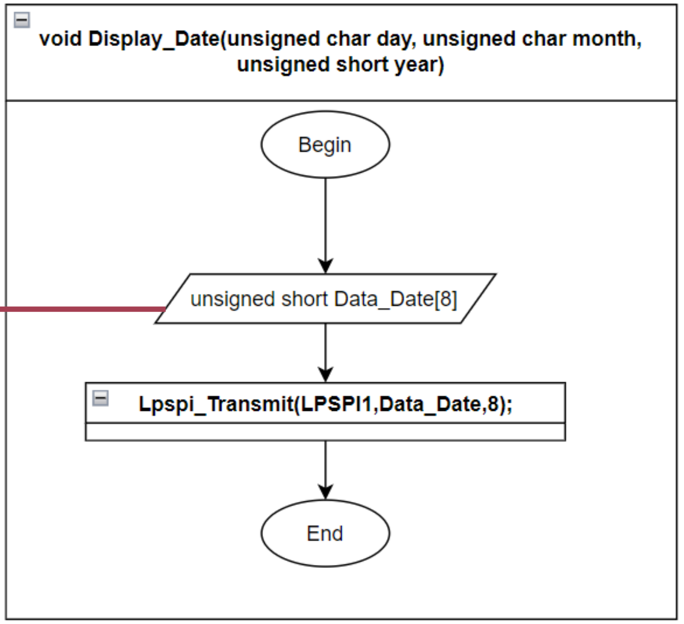
Detail design


void Display_Time(unsigned char second, unsigned char minute, unsigned char hour)
Begin
unsigned short Data_Time[8]
Lpspi_Transmit(LPSPI1,Data_Time,8);
End

```
unsigned short Data_Time[8]={(LED_0 + (second%10))
                            , (LED_1 + (second/10))
                            , (LED_2 + 10)
                            , (LED_5 + 10)
                            , (LED_3 + (minute%10))
                            , (LED_4 + (minute/10))
                            , (LED_6 + (hour%10))
                            , (LED_7 + (hour/10))};
```

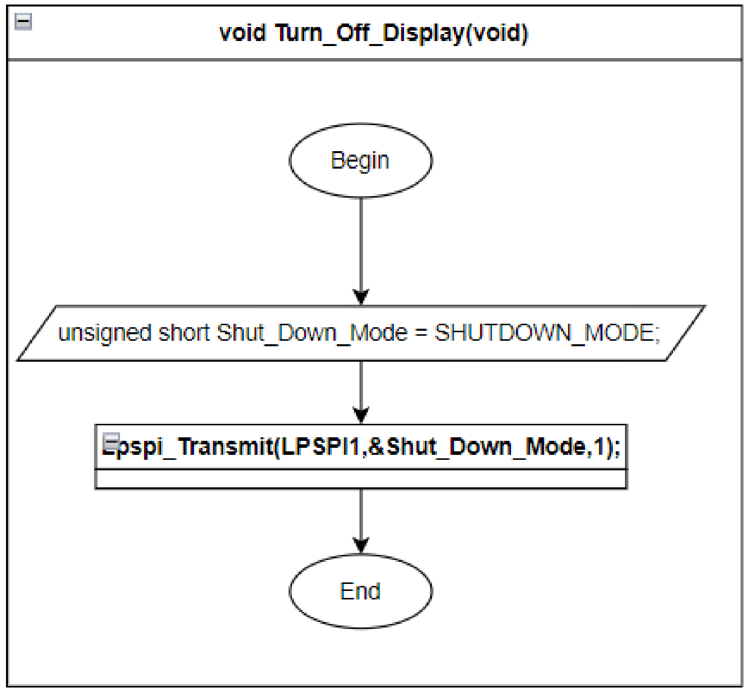
```
#define LED_0                     0x0100
#define LED_1                     0x0200
#define LED_2                     0x0300
#define LED_3                     0x0400
#define LED_4                     0x0500
#define LED_5                     0x0600
#define LED_6                     0x0700
#define LED_7                     0x0800
```

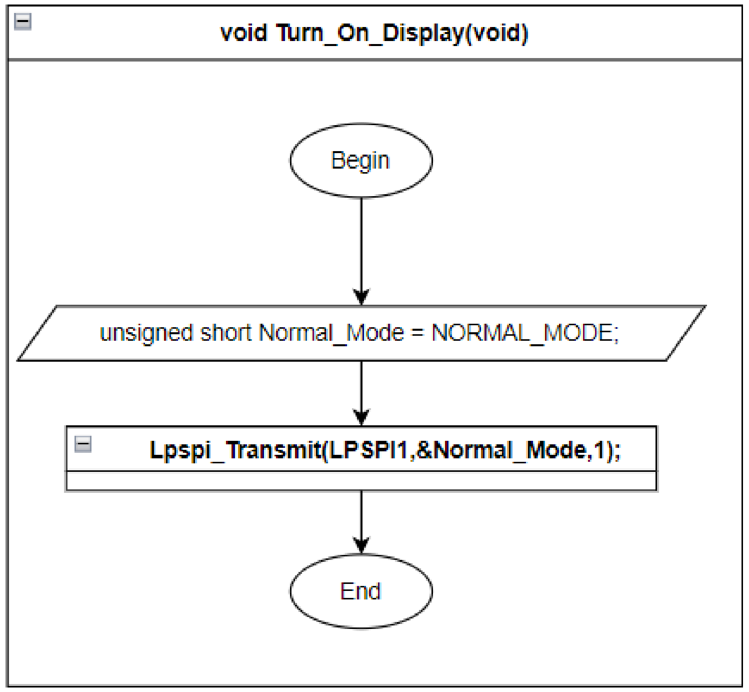
Define address of LEDs


void Display_Date(unsigned char day, unsigned char month, unsigned short year)
Begin
unsigned short Data_Date[8]
Lpspi_Transmit(LPSPI1,Data_Date,8);
End

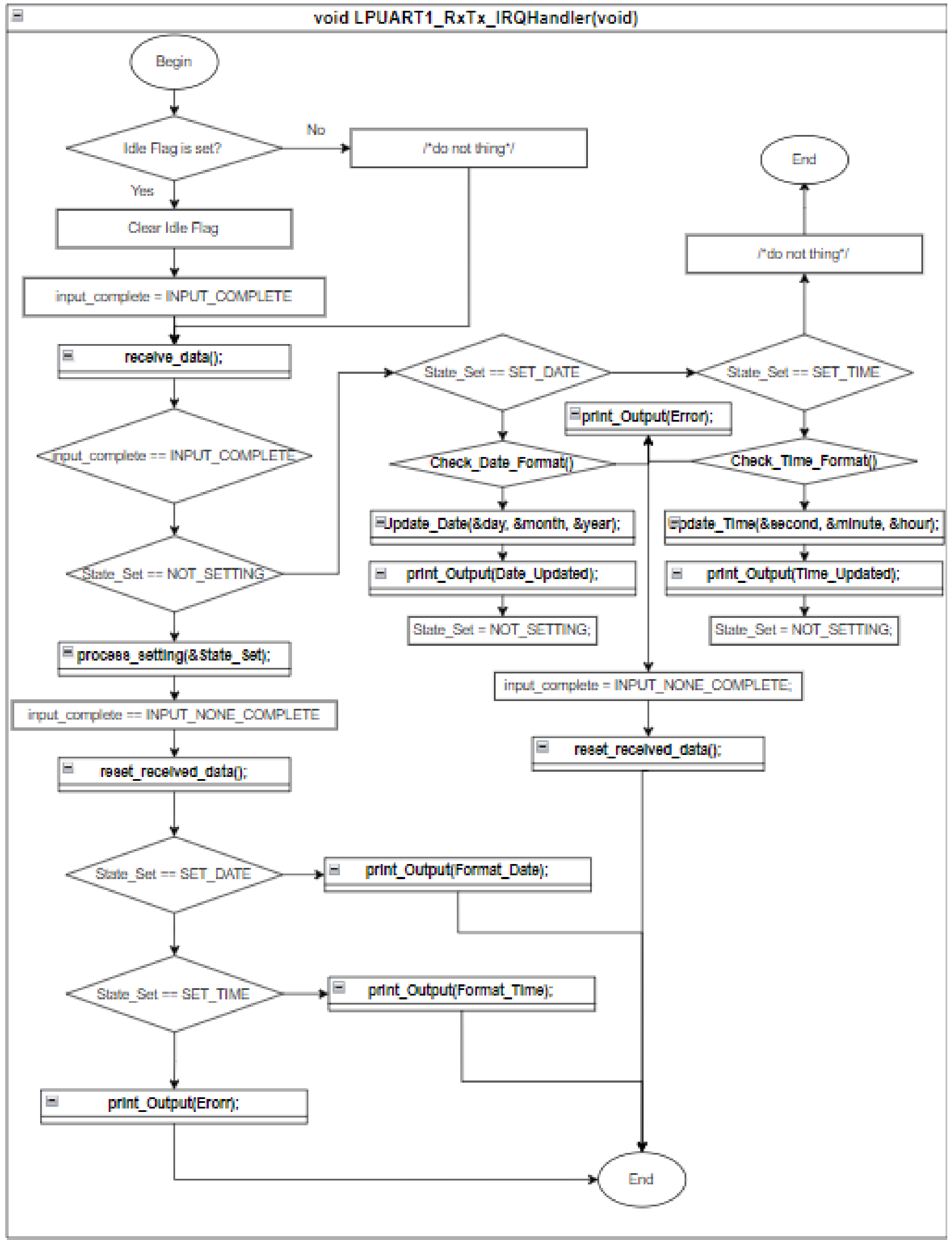
```
unsigned short Data_Date[8]={((LED_6 + (day%10)) | (1U<<7))
                            , (LED_7 + (day/10))
                            , ((LED_4 + (month%10)) | (1U<<7))
                            , (LED_5 + (month/10))
                            , (LED_0 + (year%10))
                            , (LED_1 + ((year/10)%10))
                            , (LED_2 + ((year/100)%10))
                            , (LED_3 + (year/1000))};
```

Detail design


void Turn_Off_Display(void)
Begin
unsigned short Shut_Down_Mode = SHUTDOWN_MODE;
Lpspi_Transmit(LPSPI1,&Shut_Down_Mode,1);
End

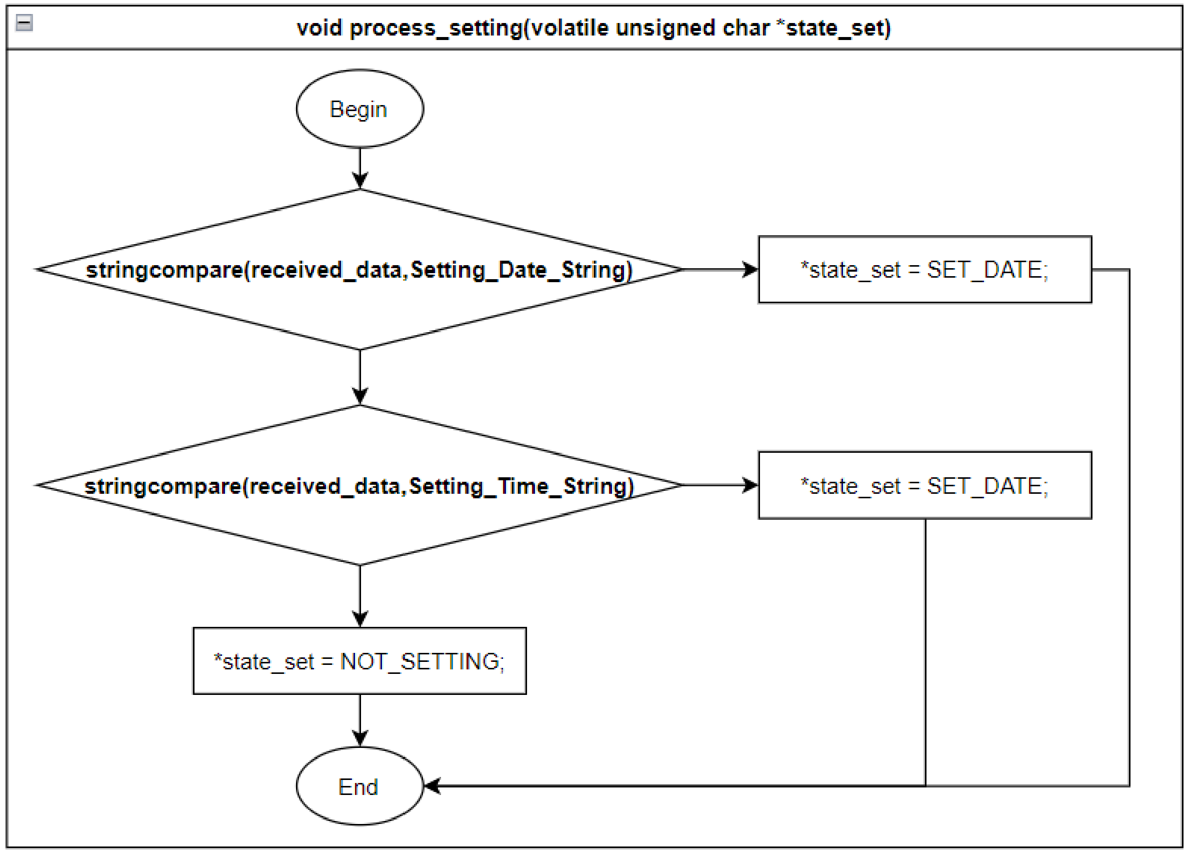

void Turn_On_Display(void)
Begin
unsigned short Normal_Mode = NORMAL_MODE;
Lpspi_Transmit(LPSPI1,&Normal_Mode,1);
End

Detail design

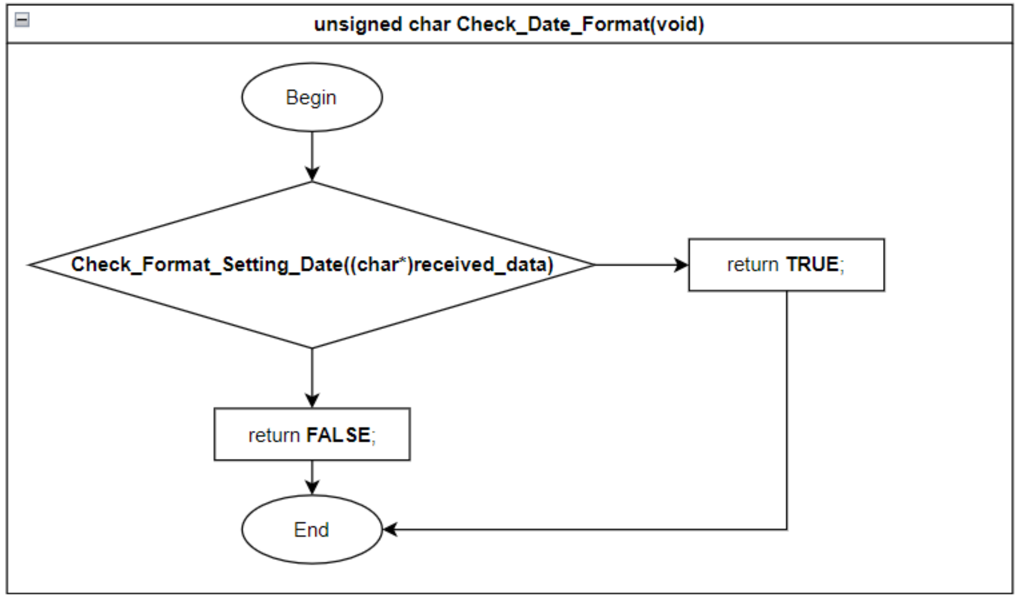


void LPUART1_RxTx_IRQHandler(void)
Begin
Idle Flag is set?
No
/*do not thing*/
Yes
Clear Idle Flag
input_complete = INPUT_COMPLETE
receive_data();
input_complete == INPUT_COMPLETE
State_Set == NOT_SETTING
process_setting(&State_Set);
input_complete == INPUT_NONE_COMPLETE
reset_received_data();
State_Set == SET_DATE
print_Output(Format_Date);
State_Set == SET_TIME
print_Output(Format_Time);
print_Output(Erorr);
End
State_Set == SET_DATE
State_Set == SET_TIME
print_Output(Error);
/*do not thing*/
End
Check_Date_Format()
Check_Time_Format()
Update_Date(&day, &month, &year);
Update_Time(&second, &minute, &hour);
print_Output(Date_Updated);
print_Output(Time_Updated);
State_Set = NOT_SETTING;
State_Set = NOT_SETTING;
input_complete = INPUT_NONE_COMPLETE;
reset_received_data();

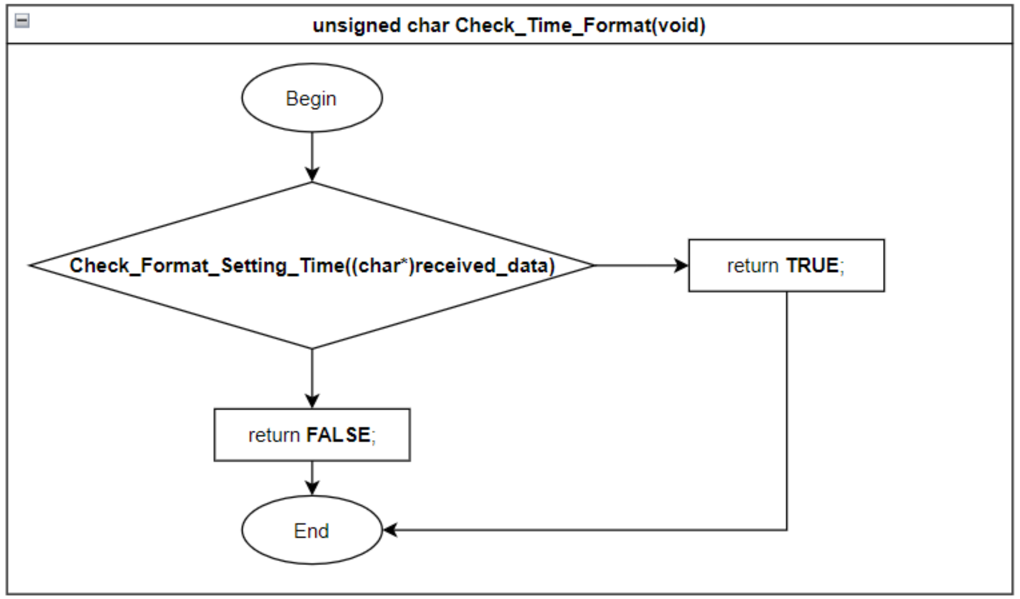
Detail design



void process_setting(volatile unsigned char *state_set)
Begin
stringcompare(received_data,Setting_Date_String)
*state_set = SET_DATE;
stringcompare(received_data,Setting_Time_String)
*state_set = SET_DATE;
*state_set = NOT_SETTING;
End

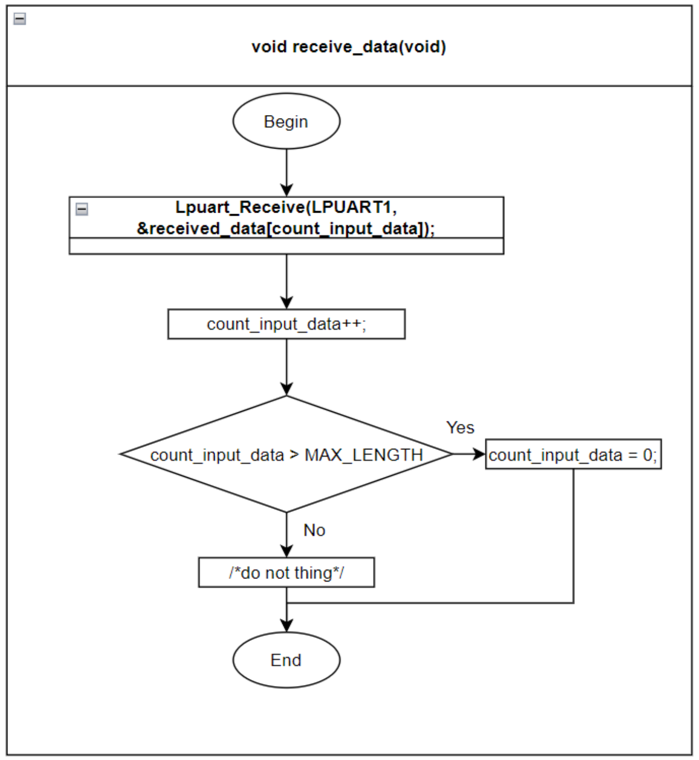


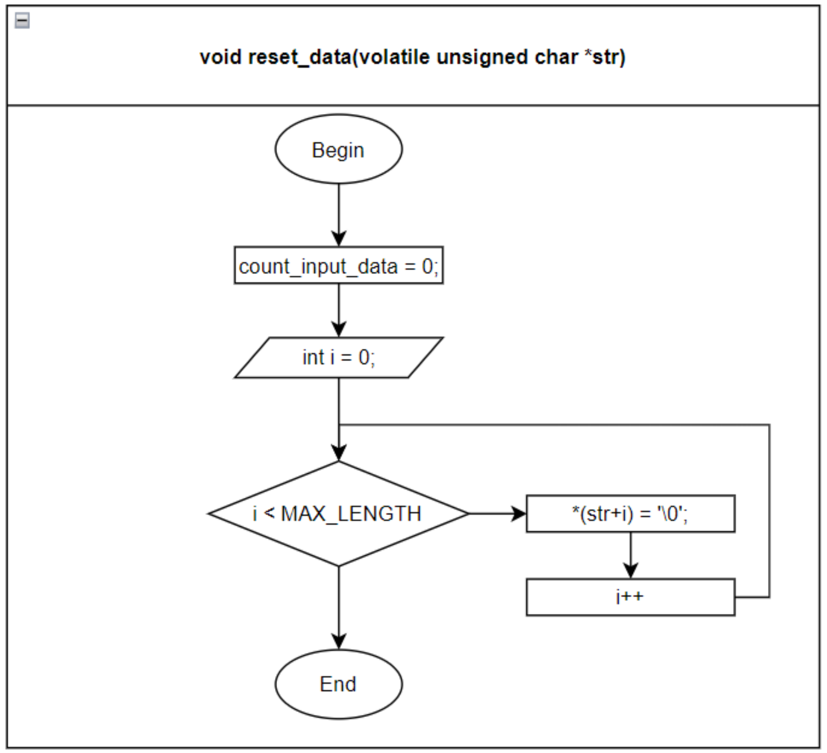
unsigned char Check_Date_Format(void)
Begin
Check_Format_Setting_Date((char*)received_data)
return TRUE;
return FALSE;
End

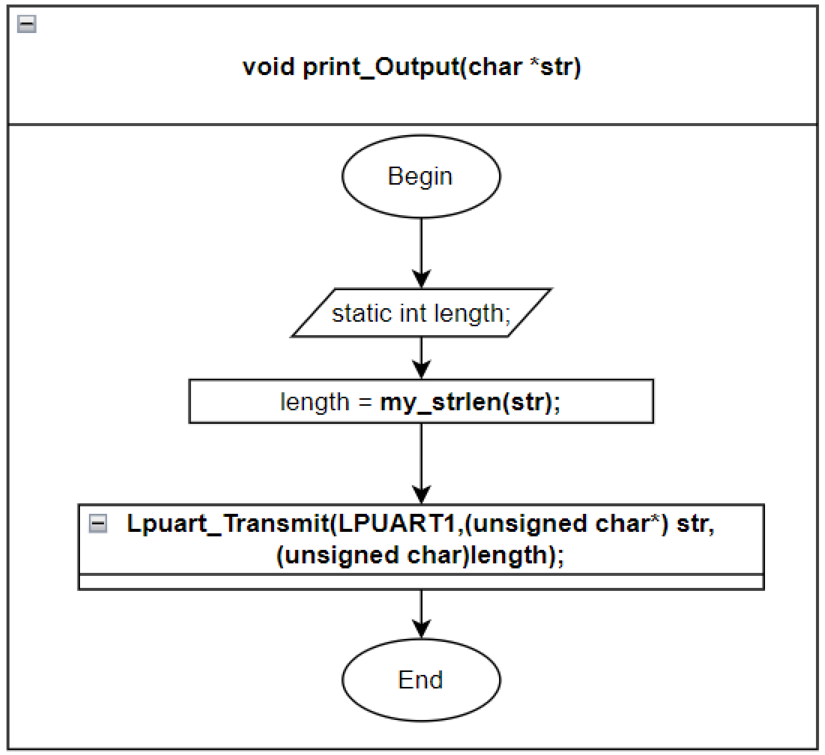


unsigned char Check_Time_Format(void)
Begin
Check_Format_Setting_Time((char*)received_data)
return TRUE;
return FALSE;
End

Detail design

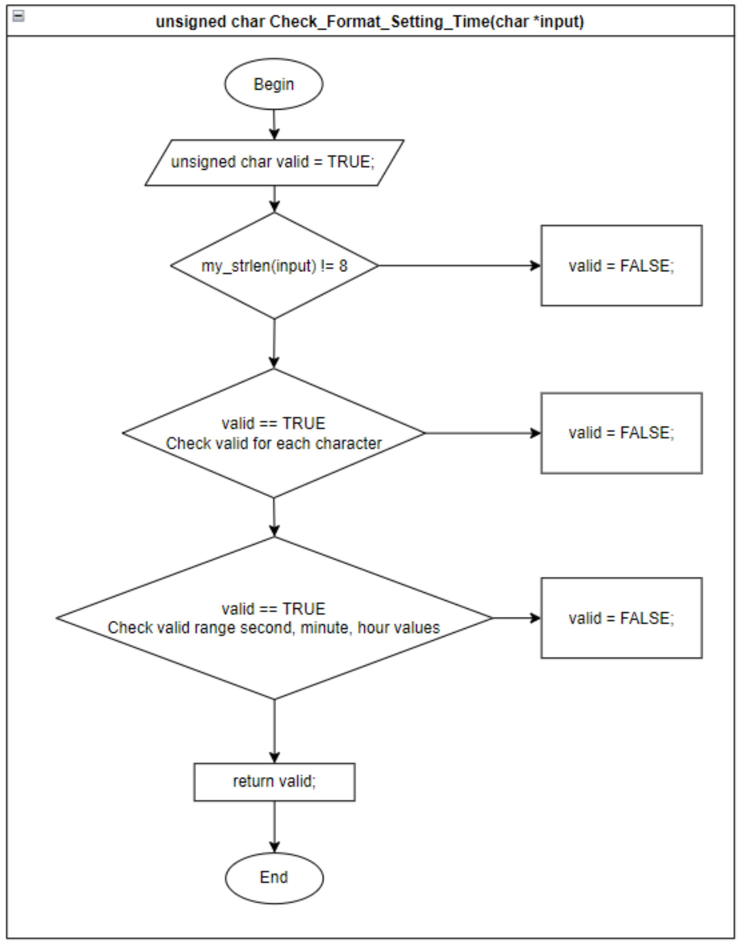




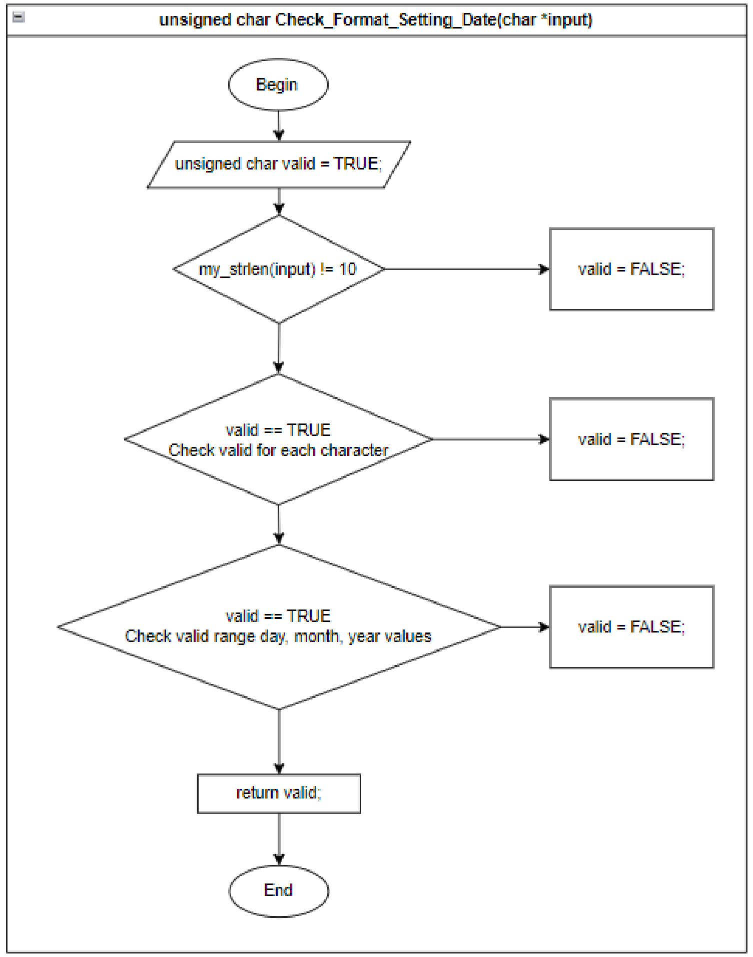


void print_Output(char *str)
Begin
static int length;
length = my_strlen(str);
Lpuart_Transmit(LPUART1,(unsigned char*) str, (unsigned char)length);
End

Detail design



unsigned char Check_Format_Setting_Time(char *input)
Begin
unsigned char valid = TRUE;
my_strlen(input) != 8
valid = FALSE;
valid == TRUE
Check valid for each character
valid = FALSE;
valid == TRUE
Check valid range second, minute, hour values
valid = FALSE;
return valid;
End



unsigned char Check_Format_Setting_Date(char *input)
Begin
unsigned char valid = TRUE;
my_strlen(input) != 10
valid = FALSE;
valid == TRUE
Check valid for each character
valid = FALSE;
valid == TRUE
Check valid range day, month, year values
valid = FALSE;
return valid;
End

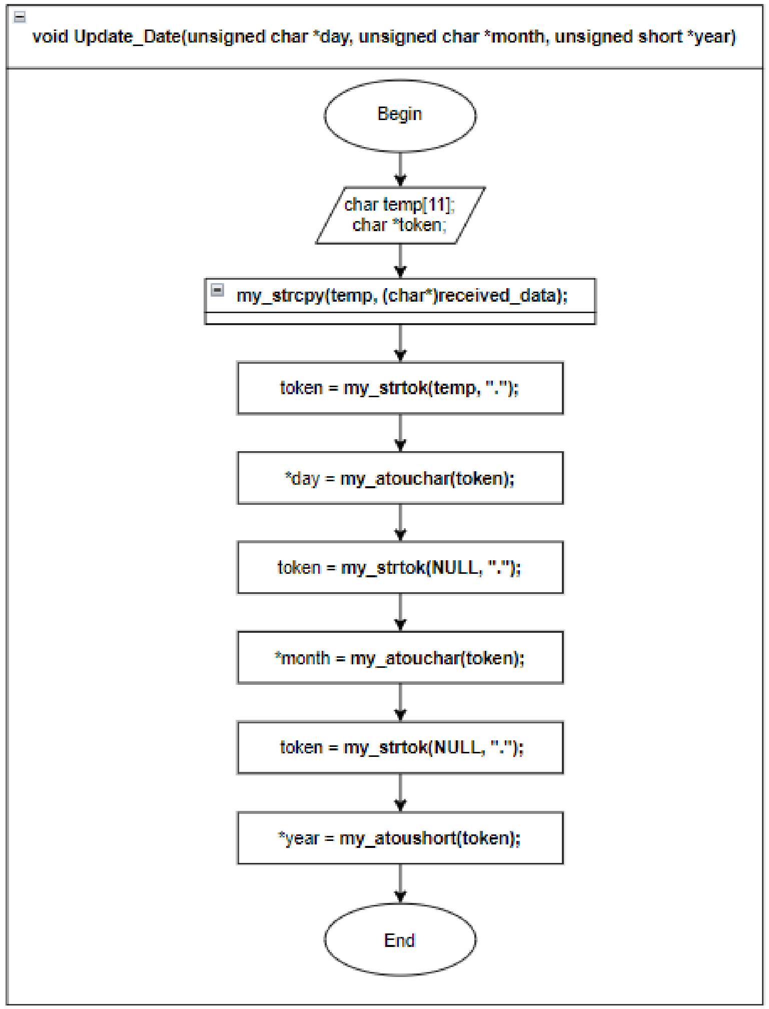
Format Time: XX-XX-XX

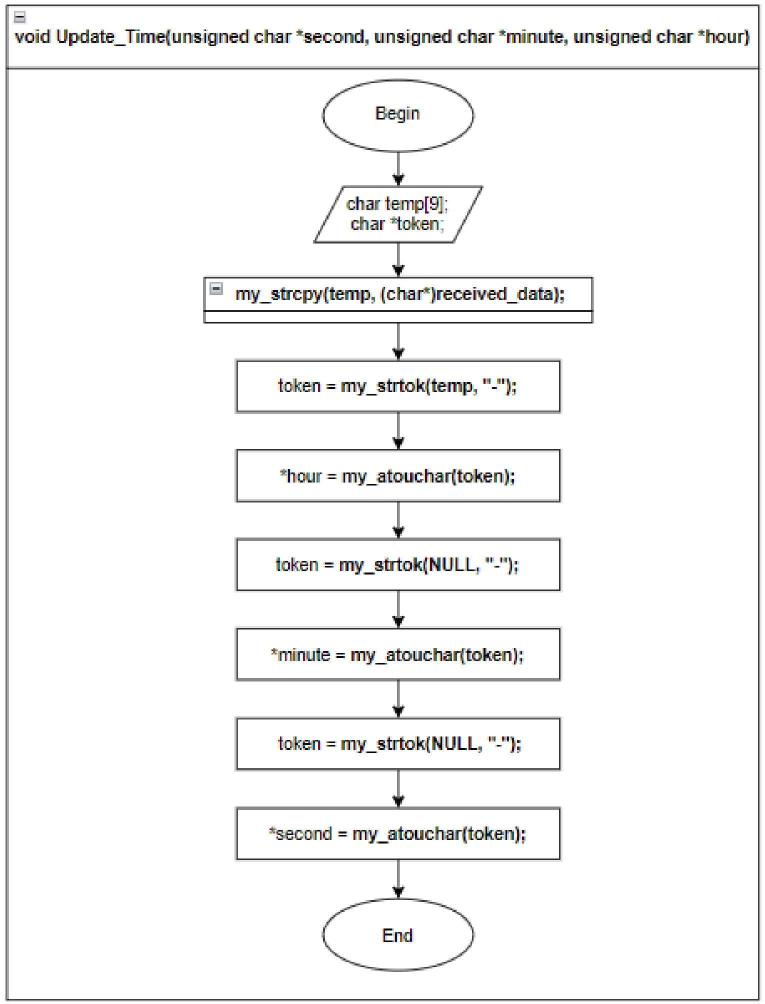
```
Please type right format: XX-XX-XX
```

Format Date: dd.mm.yyyy

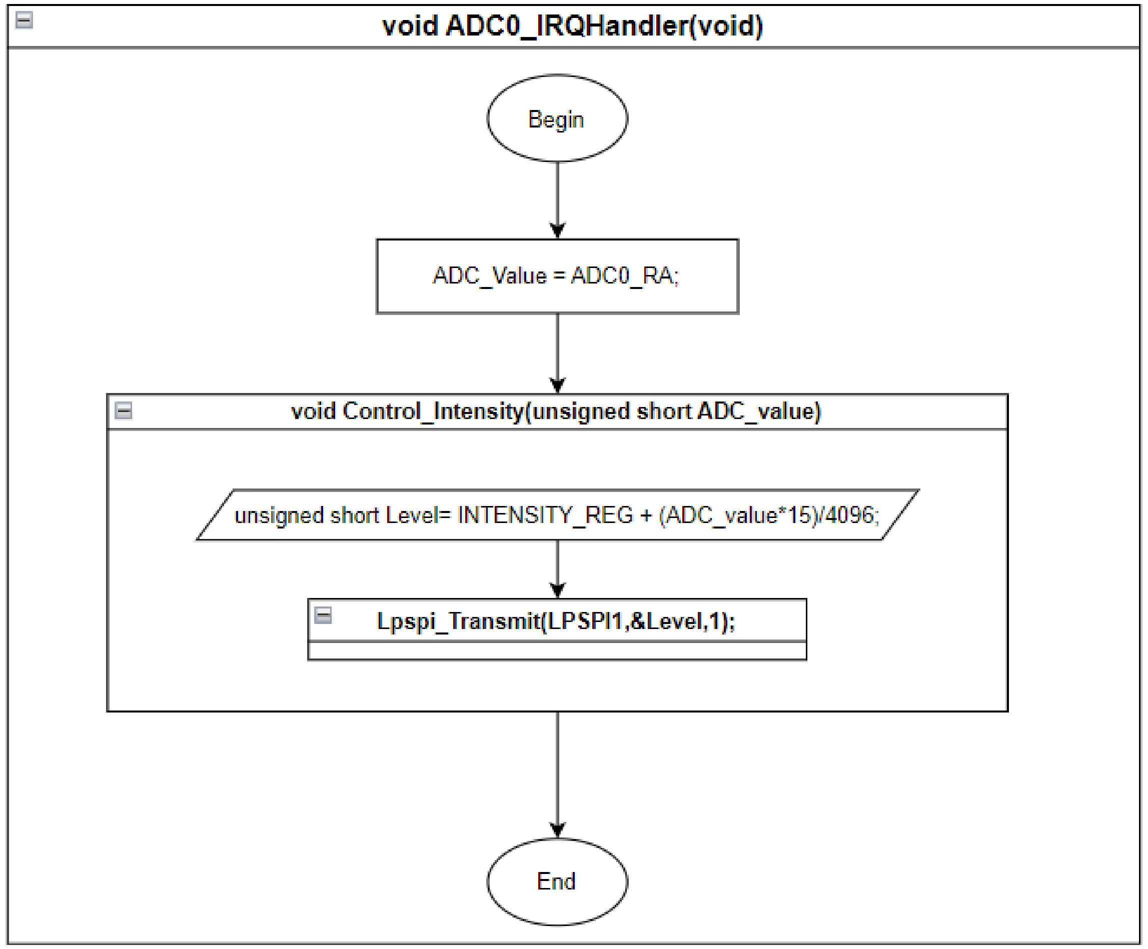
```
Please type right format: dd.mm.yyyy
```

Detail design


void Update_Date(unsigned char *day, unsigned char *month, unsigned short *year)
Begin
char temp[11];
char *token;
my_strcpy(temp, (char*)received_data);
token = my_strtok(temp, ".");
*day = my_atouchar(token);
token = my_strtok(NULL, ".");
*month = my_atouchar(token);
token = my_strtok(NULL, ".");
*year = my_atoushort(token);
End


void Update_Time(unsigned char *second, unsigned char *minute, unsigned char *hour)
Begin
char temp[9];
char *token;
my_strcpy(temp, (char*)received_data);
token = my_strtok(temp, "-");
*hour = my_atouchar(token);
token = my_strtok(NULL, "-");
*minute = my_atouchar(token);
token = my_strtok(NULL, "-");
*second = my_atouchar(token);
End

Detail design


void ADC0_IRQHandler(void)
Begin
ADC_Value = ADC0_RA;
void Control_Intensity(unsigned short ADC_value)
unsigned short Level= INTENSITY_REG + (ADC_value*15)/4096;
Lpspi_Transmit(LPSPI1,&Level,1);
End

RESULTS

CONCLUSION

Conclusion

- Review C program knowledge
- Write driver for most modules (except for ADC)
- Build successfully and logically a project
- Clean Code
- Logic code
- Easy maintenance and reuse
- Increase debug abilities

Thank you

For your attention

Mai Anh Tuan

0939942120

21145309@student.hcmute.edu.vn