**EEET2096 Laboratory 4: A Mini Project**

**Home Management System (HMS)**

**The Objectives :**

1. Learn to program Cortex-M4 ADC,
2. Learn to program Cortex-M4 Serial Port,
3. Learn to program Cortex-M4 Timer
4. Learn to use interrupt
5. Embedded System Design

**Project Description:**

You are required to design and implement a home management system by using the Cortex-M4 as the central microcontroller of the system. The project has two parts:

Part A: Basic system (weight : 80% )

Part B: Extra features (weight: 20%)

Part A (ie the basic system) functionalities are given. Your task is to design and implement them. Part B ( the extra features) is open. You can add any extra features you want to enhance the system.

Smart home also known as home automation is an application in which home devices are connected to the Internet and the control and monitoring of these devices can be done remotely over the Internet. The aim of this project is to let you to design and implement a simple automated home by using Cortex-M4 as the central microcontroller. The project is named “home management system” , instead of the popular term “smart home” to avoid the confusion as generally wireless interfaces ( eg IoT) are used to connect the devices in smart homes.

The HMS basic functions block diagram is shown below.

UART

Cortex-M4

Humidity

sensor

Light

sensor

Ventilation Fan

PC

Light

Switch

Fan

Switch

Figure 1 Basic HMS Function Diagram

The basic functionalities ( Part A) are listed below

1. Every second, the following system status parameters will be sent out via UART for monitoring
   * The Humidity
   * The light status (on/off)
   * The Fan status (on/off)
2. Light switch works in toggle mode, ie … press ON, press OFF, press ON, Press OFF, … [ note: the press has to last more than one second. ]
3. Fan switch works in toggle mode, ie press ON, press OFF, press ON, Press OFF, …

[ note: the press has to last more than one second. ]

1. If the Humidity is above a high threshold, the fan will be on.
2. Through UART, the light can be turned on or off. If light turn on/off command and light switch happen in the same time ( ie within 1 second), the command from UART has the priority.
3. Through UART, the fan can be turned on or off. If fan turn on/off command and fan switch happen in the same time ( ie within 1 second), command from UART has the higher priority. If the humidity is above the high threshold, the fan command will be ignored.
4. The PC side design is not included in Part A.

The following Cortex-M4 interfaces are recommended to use

The Humidity Sensor -------ADC1 IN0 ( PortA Pin0)

The light ---------------------- GPIOB Pin8 ( 0 for ON, 1 for OFF)

The Fan ----------------------- GPIOF Pin8 ( 0 for ON, 1 for OFF)

Light Switch ----------------- GPIOB Pin0 ( 0 for pressed, 1 for un-pressed)

Fan Switch --------------------GPIOB Pin1 ( 0 for pressed, 1 for un-pressed)

Serial Port ---------------------USART6 ( PortC pin6 ---TX, PortC pin7---RX)

**References:**

* Lecture Notes:
* Cortex-M4 Reference Manual (RM0090)
  + Chapter 13 ADC
  + Chapter 30 USART
  + Chapter 20 Basic Timers

**Preparation:**

1. **Review ADC and find out how to do the following.**
   1. ADC configuration ( eg simplest scenario, single channel, discontinuous, 8 bit ADC)
   2. ADC polling mode or interrupt mode
   3. ADC interrupt routine
   4. ADC status checking
2. **Review Serial Port and find out how to do the following**
   1. UART configuration ( eg simplest scenario, 1 stop bits, 8 bits data, baud rate = 19.2kbps, no parity check)
   2. UART transmission and receiving
   3. UART interrupt routine for receiving and transmitting
   4. UART polling for transmitting
3. **Review the Systick Timer and find out how to do the following**
   1. Configure Systick to have time interval 1 second
   2. Design the tasks in SysTick interrupt routine ( eg, transmit the system parameters, start the ADC, …)
   3. Check if the any of the switch button is pressed.
4. **System design**

System design or program structure need to be designed first. You are encouraged to have your own design. The following design can be a reference for you. You can use the this design if it meets your requirement. For students who will do ( Part B, extra features), the design have to be modified or enhanced further.

/\* some global system variables to be declared. These variables could be accessed by interrupt routines and other functions defined. \*/

// for example

int current\_Humidity; // current\_Humidity is used for the current humidity. Its value

// should be updated at least once in a second.

int light; // light status, 0 for On, 1 for OFF

bool lightCommandFlag; // this flag will indicate if a light command

// is received from the UART

int lightCommand; // lightCommand record the light command from UART

bool lightToggleFlag; // if it is 1 it means the button is pressed and released, 0 no switch

// press action is detected

// this flag can be changed in the SysTick interrupt routine

int lightSwitchStatus; // 0 means the button pressed, 1 means unpressed

/\* ADC interrupt routine \*/

// check if the targeted trigger occurred

// if yes do ADC reading and update the current\_Humidity

/\* SysTick Interrupt routines \*/

/\* UART interrupt routines \*/

main( )

{

// Declaration of local variables

…

// System configuration. This should include ADC, Serial Port and the SysTick timer…

…

for ( ; ; ) // forever loop

{

…

// check if the light command flag is set.

// if yes execute the command from UART

// else check if the light toggle flag is set

// if yes, toggle the light

// clear light command flag

// clear light toggle flag

…

// check if the fan command flag is set.

// if yes execute the command from UART

// else check if the fan toggle flag is set

// if yes, toggle the fan

// clear fan command flag

// clear fan toggle flag

…

}

1. **Detection of Switch Press**

The light switch and fan switch work as toggle buttons. Assume the current status of light is on, if a press of the light switch is detected, the light will be turned off. if another press is detected, the light will be on. So light changes from ON to OFF or from OFF to ON only if a button press is detected. The following is the voltage level of an input pin from a switch.

Not pressed

Not pressed

pressed

Rising edge

x

x + T

Figure 2. Button press and rising edge

The button press detection is a detection of the rising edge ( or falling edge) from the input pin. If we assume the button press has to last more than 1 second, and the input pin is read frequently with time interval T. If T is equal or less than 1 second, then the rising edge will be detected, ie the edge wont be missed. If T is less than 1 second, there is no guarantee that the edge will be detected.

To detect the rising edge , the program has to keep the previous input pin reading s(x). If the previous reading s(x) is LOW and the current reading s(x+T) is HIGH, then a button press is detected. The switch toggle flag shall be set.

1. **UART Communication Protocol Design**

Generally a communication protocol is needed for communication between two parties or among multi parties. In this project, a serial communication protocol between Cortex-M4 and PC is needed. The following simple protocol is an example you can adopt in your project. You can also design your own if you have more devices in your system.

From Cortex-M4 to PC

Every second, a set of parameters including the humidity, the light status, the fan status is transmitted from Cortex-M4 to PC. The data is three bytes long and is arranged in the following format.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | x | x | x | x | x | x | x | x | 0 | 0 | 0 | 0 | 0 | 0 | x | y |
| First byte is a header  0xAD | | | | | | | | Humidity Value | | | | | | | | x--- light status ( 0 or 1)  y---- fan status ( 0 or 1) | | | | | | | |

A header is needed to indicate a new set of parameter is coming.

From PC to Cortex-M4

The command from PC could come at any time. The command could be

the light ON or OFF

the Fan ON or OFF

Both the light ON/OFF and the Fan ON/OFF

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | X | Y | x | y |
| First byte is a header  0xAD | | | | | | | | X is 1 ->  a light command included  Y is 1 ->  a fan command included  x--- light command ( 0/1)  y---- fan command ( 0/1) | | | | | | | |

Your program has to listen to the serial port all the time. For example, if a light command is received, the lightCommandFlag should be set and the lightCommand should record x. The command should be executed in the for loop in main( ). After execution, the lightCommandFlag should be cleared and the variable light should keep the current ON/OFF status.

**Coding and Debugging**

Without the real board, the debugging and testing become very difficult. Therefore, you are required to provide clear comments for your code. The compiler should be able to pick all the syntax errors. You should test as much as you can for the logic of your code. Your testing should be included in your report. You are required to zip your project and submit with your report to canvas. The comments and testing will be part of the assessment. Submit your .7z file to “Lab4 Zipped Project”.

**Report**

You don’t need to prepare two parts for part A and Part B if you have tried part B. You can combine then into one. But you need to clearly indicate that if you have tried part B.

The report should have

**A system level design which includes**

* all input and output interfaces descriptions.
* All function ( including interrupt routines) descriptions
* Serial port communication protocols
* Program structures

**Your C files**

Attach your C files as Appendixes

**A Testing report**

Present any testing you have done on your code. There is no clear guide on the testing, you just do as much as you can. The test is mainly on some design logic.

**Discussion**

You should include a discussion on challenges you have had in this project.

**Demonstration**

A demonstration is scheduled on week 12. Ten minutes per group. The schedule is shared with you in Canvas Collaborate [ Note: Collaborate is different from Collaborate Ultra.] You need to book a time slot for your group. The Demo session is basically a Q&A session. Collaborate Ultra will be used. You can show me your project and code. Questions will be asked on your code.