Report on

**EMBEDDED SYSTEMS LABORATORY**

**BATCH -11**

NITHISHA.S 15L136

NIVEDHIKA.S 15L137

NIVETHA.P 15L138

Dissertation submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

Branch: **ELECTRONICS AND COMMUNICATION**

**ENGINEERING**



DECEMBER 2017

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PSG COLLEGE OF TECHNOLOGY

COIMBATORE-641004

**EX.NO: 2 SIMULATING A PROJECT USING ARM**

**DATE: 05.12.2017 CONTROLLER IN KEIL IDE**

**AIM:**

To study and simulate a project using ARM controller in keil IDE.

**SOFTWARE USED:**

Keil uVision 4

**THEORY:**

**Types of interrupt:**

**8051** supports level 0 and falling edge triggered interrupts.

**LPC2148** supports all the **4** hardware interrupts.

External interrupts:

ENT0,ENT1,ENT2,ENT3

All the four interrupts either level triggered or edge triggered.

**Software interrupt:**

If interrupt is generated by setting a flag it is software interrupt. There are n number of software interrupt .Hardware interrupt is also a software interrupt.

**Synchronous serial communication:**

It is the one where large amount of data transfer takes place. It is periodic. E.g.: downloading image

**Asynchronous serial communication:**

It is the one where small amount of data transfer takes place. It is random (ie)time interval is not fixed. E.g. data transfer from keyboard to CPU.

In microcontroller we use only UART(Universal Asynchronous Receiver Transmitter) not USART(Universal Synchronous Asynchronous Receiver Transmitter) due to very small memory.

For serial communication there should be start bit and stop bit for both synchronous and asynchronous.

**Start bit:**All start bits in all serial communication are transistion from **1** to **0**.

**Stop bit:**All stop bits in all serial communication are transistion from **0** to **1**.

Microcontroller identifies there is a input by the transistion in the pin.

All transmitters are **Parallel In Serial Out** and all receivers are **Serial In Parallel Out.** When all the 8 bits come in, we can take the data from SBUF.

If data is to be transmitted TI flag should be checked. So TI flag set says that transmission is complete and next byte can be placed in SBUF.Without TI flag there is a possibility of overwriting the data.

Before all bits of data are received there is a possibility of RI to take all data. It have to wait until all the bits are received.

**BAUD RATE:**

Number of bits transferred per second. It is used during serial communication.

If baud rate is 9600 then 9600 bits per second is transmitted.

Then for 1 bit= 1/9600 =104µs.Therefore after every 104µs one bit is shifted in the microcontroller .

**ANALOG TO DIGITAL CONVERTOR(ADC):**

A digital programmable device cannot compute infinite values. i.e., digital programmable device like microprocessor and microcontroller cannot compute integration so this should be summation. Hence infinite values in analog are converted to finite values in digital. For this ADC is used.

ON - CHIP ADC IN LPC2148

* ADC0 - 6 channels
* ADC1 - 8 channels

Total 14 channels.They are 10 bit ADC.

There is one DAC in LPC2148 so signal processing can be done with ADC and DAC.

Hardware multiplier and accumulator helps in digitall signal processing.

**PULSE WIDTH MODULATION:(PWM)**

In mobile phones, brightness can be adjusted by using PWM. Tesla is a manufacturing company which is now producing electric cars whose speed is controlled by using the PWM mechanism since the tyres are dc motors. PWM is the only system available to control the speed..PWM increases or decreases the DC voltage.

TTL means 0 to 5v. CMOS means 0 to 3.3v.

Logic 0🡪0v

Logic 1🡪5 or 3.3v

Through PWM we can get between 0 and 3.3v with high precision.

There are **6** PWM available in LPC2148.

**Real time clock:**

A clock which produces 100% accurately 1 sec delay is called real time clock.It shows date,day,time.

**WATCHDOG TIMER:**

A Watch dog timer is also available in LPC2148. It is the guardian of LPC2148 controller.

**Von Neumann architecture**

Program to be executed, data to be processed and processed results all are stored in one addressable memory .i.e., program and data memory combined.starting address of program memory and data memory will not be same here.

**LPC2148 is**  **Von Neumann** architecture.

**Harvard architecture**

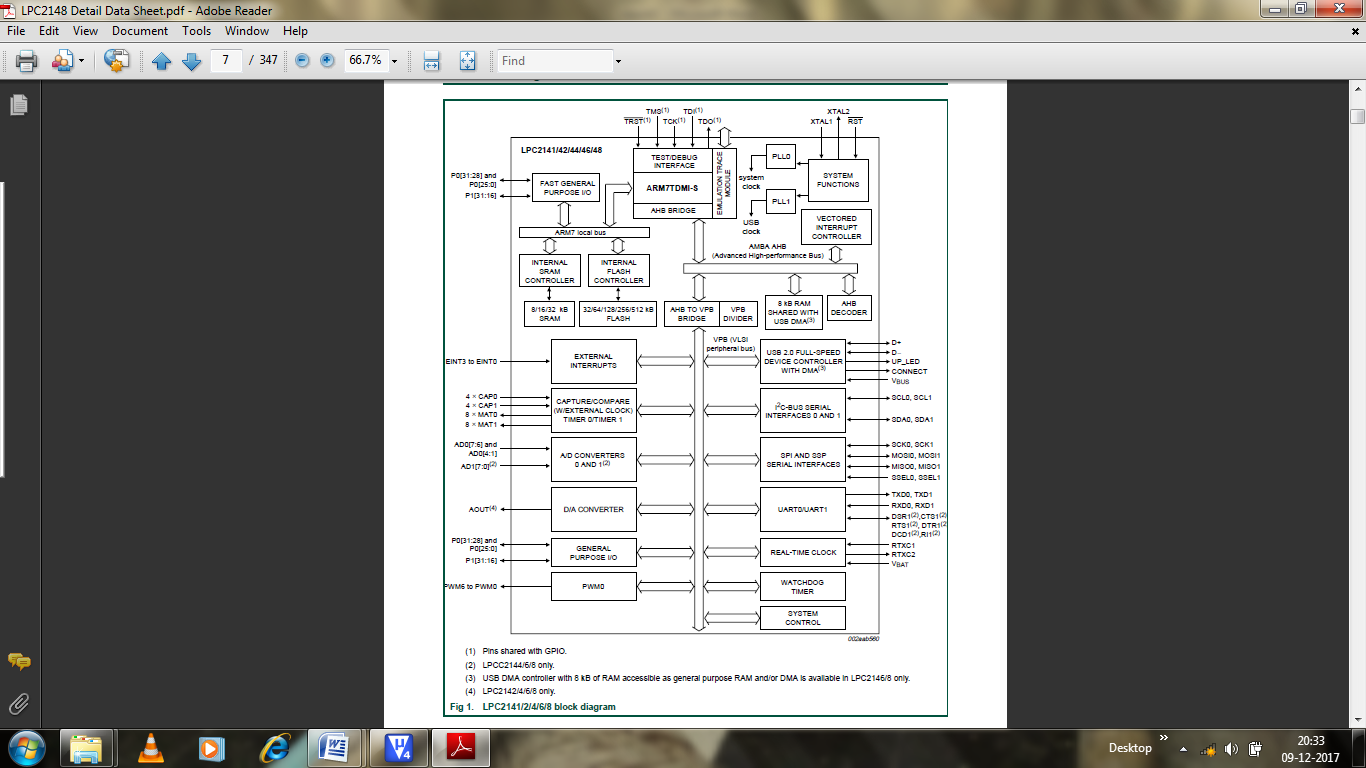
There will be separate program memory and data memory.Program memory starts with 0000H. Data memory starts with 0000H.

**8051** is **Harvard** architecture.

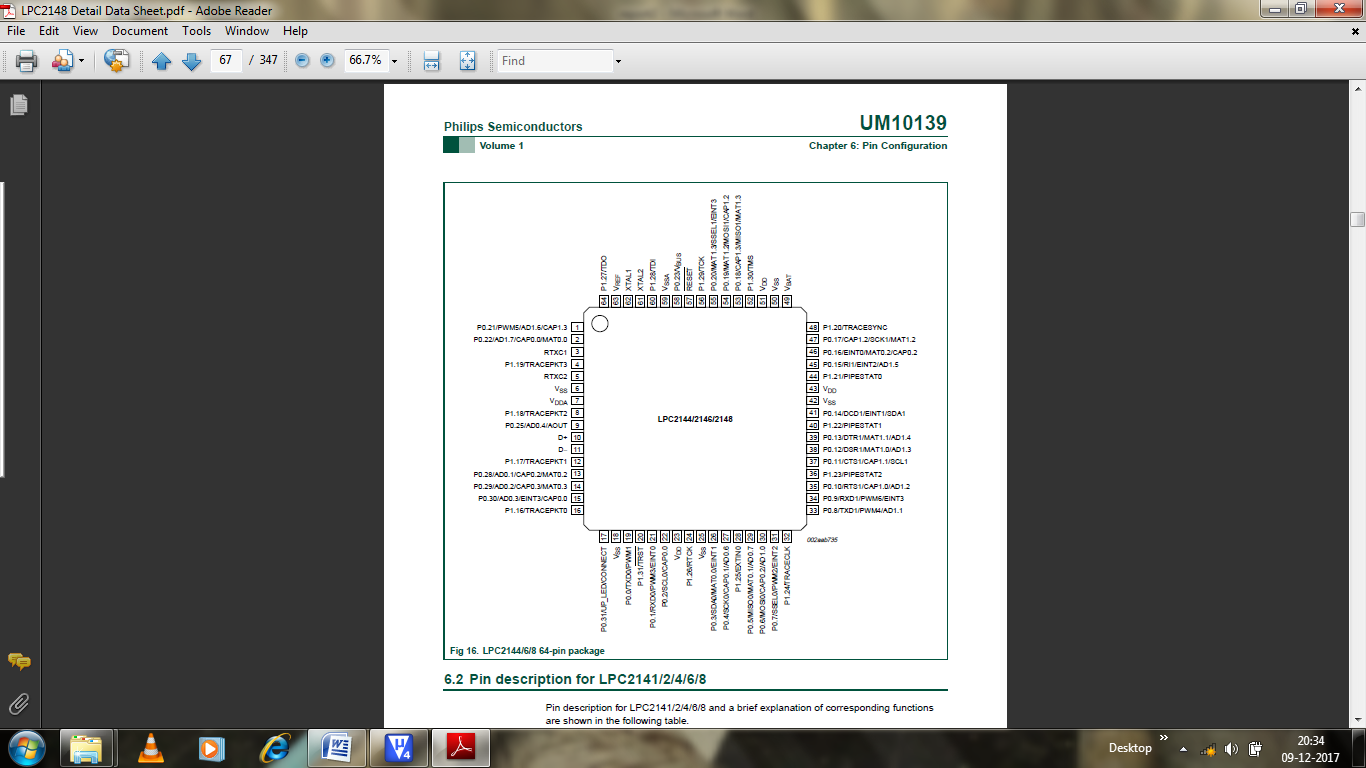
**Timers**:

It is a software interrupt that produces periodic delay. Without timer there is no serial communication.

**BLOCK DIAGRAM OF LPC2141/2/4/6/8**

****

**PIN DIAGRAM OF LPC2148**

****

**PIN CONNECT BLOCK**

**Features**

* Allows individual pin configuration.

**Applications**

* The purpose of the Pin Connect Block is to configure the microcontroller pins to the desired functions.

**Description**

The pin connect block allows selected pins of the microcontroller to have more than one function. Configuration registers control the multiplexers to allow connection between the pin and the on chip peripherals.

Peripherals should be connected to the appropriate pins prior to being activated, and prior to any related interrupt(s) being enabled. Activity of any enabled peripheral function that is not mapped to a related pin should be considered undefined.

The only partial exception from the above rule of exclusion is the case of inputs to the A/D converter. Regardless of the function that is selected for the port pin that also hosts the A/D input, this A/D input can be read at any time and variations of the voltage level on this pin will be reflected in the A/D readings. However, valid analog reading(s) can be obtained if and only if the function of an analog input is selected. Only in this case proper interface circuit is active in between the physical pin and the A/D module. In all other cases, a part of digital logic necessary for the digital function to be performed will be active, and will disrupt proper behaviour of the A/D.

**GENERAL PURPOSE INPUT/OUTPUT PORTS (GPIO)**

**Features**

* Every physical GPIO port is accessible via either the group of registers providing an enhanced feature and accelerated port access or the legacy group of registers.
* Accelerated GPIO functions:
* GPIO registers are relocated to the ARM local bus so that the fastest possible I/O timing can be achieved.
  + Mask registers allow treating sets of port bits as a group, leaving other bits unchanged.
  + All registers are byte and half-word addressable.
  + Entire port value can be written in one instruction.
* Bit-level set and clear registers allow a single instruction set or clear of any number of bits in one port.
* Direction control of individual bits.
* All I/O default to inputs after reset.

**Applications**

* General purpose I/O
* Driving LEDs, or other indicators
* Controlling off-chip devices
* Sensing digital inputs

**Pin description**

|  |  |  |
| --- | --- | --- |
| **Pin** | **Type** | **Description** |
| P0.0 – P0.31  P1.16 – P1.31 | Input /  Output | General purpose input/output. The number of GPIOs actually available depends on the use of alternate functions. |

**UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSMITTER 0 (UART0)**

**Features**

* 16 byte Receive and Transmit FIFOs
* Register locations conform to ‘550 industry standard.
* Receivers FIFO trigger points at 1, 4, 8, and 14 bytes.
* Built-in fractional baud rate generator with auto bauding capabilities.
* Mechanism that enables software and hardware flow control implementation.

**Pin description**

|  |  |  |
| --- | --- | --- |
| **Pin** | **Type** | **Description** |
| RXD0 | Input | **Serial Input.** Serial receive data. |
| TXD0 | Output | **Serial Output.** Serial transmit data. |

**PROGRAM**

1. **BLINK**
2. #include<lpc214x.h> //including the header files for LPC2148

microcontroller

void main()

{

IO1DIR=0xffffffff; //Initializing the direction registers of Port 1

as output

while(1)

{

IO1PIN=~IO1PIN; //Togging the Port 1 i/o pins of

LPC2148

}

}

1. #include<lpc214x.h>

void main()

{

IO1DIR=0xffffffff;

while(1)

{

IO1PIN=0xff00ff00;

}

}

1. **Blink with delay**
2. #include<lpc214x.h>

void delay(unsigned long int x)

{

while(x--);

}

void main()

{

IO1DIR=0xffffffff;

while(1)

{

IO1SET=0xffffffff;

delay(5000000);

IO1CLR=0xffffffff;

delay(5000000);

}

}

1. #include<lpc214x.h>

void delay(unsigned long int x)

{

while(x--);

}

void main()

{

IO1DIR=0xffffffff;

while(1)

{

IO1SET=0xffffffff;

delay(500000);

IO1CLR=0xffffffff;

delay(500000);

IO1SET=0xfffffff0;

delay(500000);

IO1CLR=0xffffffff;

delay(500000);

IO1SET=0xffffff00;

delay(500000);

IO1CLR=0xffffffff;

delay(500000);

IO1SET=0xfffff000;

delay(500000);

IO1CLR=0xffffffff;

delay(500000);

IO1SET=0xffff0000;

delay(500000);

IO1CLR=0xffffffff;

delay(500000);

IO1SET=0xfff00000;

delay(500000);

IO1CLR=0xffffffff;

delay(500000);

IO1SET=0xff000000;

delay(500000);

IO1CLR=0xffffffff;

delay(500000);

IO1SET=0xf0000000;

delay(500000);

IO1CLR=0xffffffff;

delay(500000);

}

}

1. **SET and CLEAR**
2. #include<lpc214x.h>

void main()

{

IO1DIR=0xffffffff;

while(1)

{

IO1SET=0xffffffff;

IO1CLR=0xffffffff;

}

}

1. #include<lpc214x.h>

void main()

{

IO1DIR=0xffffffff;

while(1)

{

IO1SET=0x0f0f0f0f;

IO1CLR=0xffffffff;

IO1SET=0xf0f0f0f0;

IO1CLR=0xffffffff;

IO1SET=0xffff0000;

IO1CLR=0xffffffff;

IO1SET=0x0000ffff;

IO1CLR=0xffffffff;

}

}

1. **UART**

**i**) #include<lpc214x.h>

void delay(unsigned long int x)

{

while(x--);

}

void main()

{

IO1DIR=0xffffffff;

PINSEL0=0x00000005;

U0LCR=0x80;

U0DLL=0x62;

U0LCR=0x03;

while(1)

{

while(U0LSR!=0x60);

U0THR='B';

while(U0LSR!=0x60);

U0THR='Y';

while(U0LSR!=0x60);

U0THR='E';

while(U0LSR!=0x60);

U0THR=' ';

while(U0LSR!=0x60);

U0THR='B';

while(U0LSR!=0x60);

U0THR='Y';

while(U0LSR!=0x60);

U0THR='E';

IO1SET=0xffffffff;

delay(5000000);

IO1CLR=0xffffffff;

delay(5000000);

}}

**ii**) #include<lpc214x.h>

void delay(unsigned long int x)

{

while(x--);

}

void main()

{

IO1DIR=0xffffffff;

PINSEL0=0x00000005;

U0LCR=0x80;

U0DLL=0x62;

U0LCR=0x03;

while(1)

{

while(U0LSR!=0x60);

U0THR='H';

while(U0LSR!=0x60);

U0THR='E';

while(U0LSR!=0x60);

U0THR='L';

while(U0LSR!=0x60);

U0THR='L';

while(U0LSR!=0x60);

U0THR='O';

while(U0LSR!=0x60);

U0THR='W';

while(U0LSR!=0x60);

U0THR='O';

while(U0LSR!=0x60);

U0THR='R';

while(U0LSR!=0x60);

U0THR=' L';

while(U0LSR!=0x60);

U0THR='D';

IO1SET=0xffffffff;

delay(5000000);

IO1CLR=0xffffffff;

delay(5000000);

}}

**RESULT**

Thus, ARM controller LPC2148 has been studied and programs have been simulated and verified using Keil IDE.