EMBEDDED SYSTEMS LAB MANUAL

(On ARM microcontroller)

LPC2148 (ARM) MICROCONTROLLER

INNOVATION NOT IMITATION

ARK AUTOMATION SYSTEMS
Comprehensive Solutions for all Lab Requirements
F-2, Block-A, “Ponni Illam” No : 2, Ponniamman Koil street, Puzhuthivakkam, Chennai - 91
Mail : arkautomationsystems@gmail.com, jesudhassmith@gmail.com, 94440 35048,98416 88767

Sl.no	LIST OF EXPERIMENTS	Remarks
1.	Study of ARM evaluation system – KEIL µVision 4 Tool – Flash Magic Tool	
2.	Interfacing ADC and DAC	
3.	Interfacing LED and PWM	
4.	Interfacing real time clock and serial port	
5.	Interfacing keyboard and LCD	
6.	Interfacing EPROM and interrupt	
7.	Mailbox	
8.	Interrupt performance characteristics of ARM	
9.	Flashing of LEDS	
10.	Interfacing stepper motor and temperature sensor	
11.	Implementing zigbee protocol with ARM	

.

.

Ex.No: 1 **STUDY OF ARM EVALUATION SYSTEM**

AIM:

To study the Performance ARM2148 evaluation system.

1. INTRODUCTION TO ARM BOARD (LPC2148)

This section of the document introduces LPC2148 microcontroller board based on a 16-bit/32-bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combine microcontrollers with embedded high-speed flash memory ranging from 32 kB to 512 kB. A 128-bit wide memory interface and unique accelerator architecture enable 32-bit code execution at the maximum clock rate. For critical code size applications, the alternative 16-bit Thumb mode reduces code by more than 30% with minimal performance penalty. The meaning of LPC is Low Power Low Cost microcontroller. This is 32 bit microcontroller manufactured by Philips semiconductors (NXP).

Due to their tiny size and low power consumption, LPC2148 is ideal for applications where miniaturization is a key requirement, such as access control and point-of-sale.

1.1 Features of ARM Microcontroller

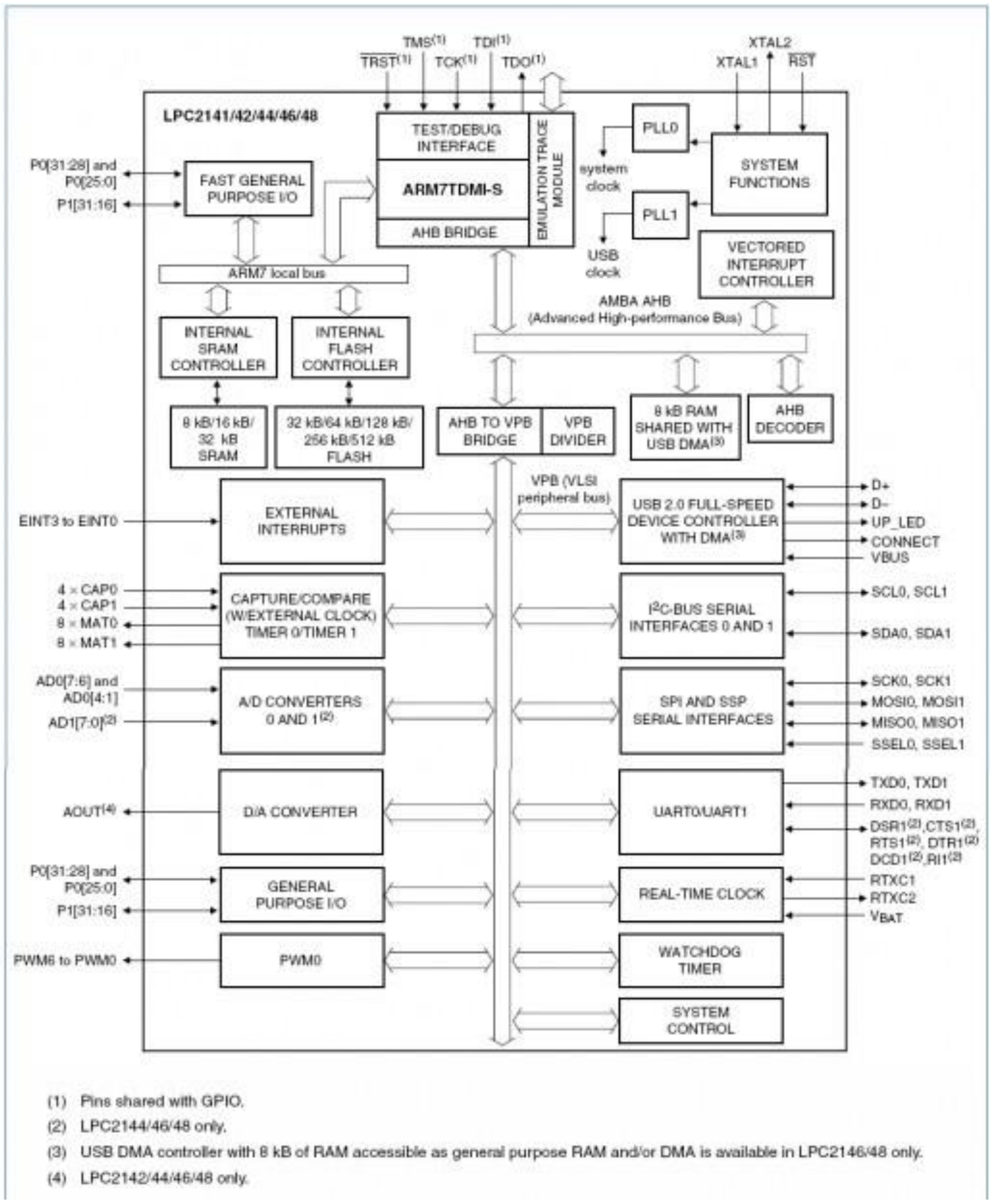
- 16-bit/32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package.
- 8 kB to 40 kB of on-chip static RAM and 32 kB to 512 kB of on-chip flash memory; 128-bit wide interface/accelerator enables high-speed 60 MHz operation.
- In-System Programming/In-Application Programming (ISP/IAP) via on-chip boot loader software, single flash sector or full chip erase in 400 ms and programming of 256 Bytes in 1 ms Embedded ICE RT and Embedded Trace interfaces offer real-time debugging with the on-chip Real Monitor software and high-speed tracing of instruction execution.
- USB 2.0 Full-speed compliant device controller with 2kB of endpoint RAM. In addition, the LPC2148 provides 8 kB of on-chip RAM accessible to USB by DMA.
- One or two (LPC2141/42 vs, LPC2144/46/48) 10-bit ADCs provide a total of 6/14 analog inputs, with conversion times as low as 2.44 ms per channel.
- Single 10-bit DAC provides variable analog output (LPC2148 only)

- Two 32-bit timers/external event counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.
- Low power Real-Time Clock (RTC) with independent power and 32 kHz clock input.
- Multiple serial interfaces including two UARTs , two Fast I2C-bus (400 kbit/s),SPI and SSP with buffering and variable data length capabilities.
- Vectored Interrupt Controller (VIC) with configurable priorities and vector addresses.
- Up to 45 of 5 V tolerant fast general purpose I/O pins in a tiny LQFP64 package. Up to nine edge or level sensitive external interrupt pins available.
- 60 MHz maximum CPU clock available from programmable on-chip PLL with settling time of 100 ms.
- Power saving modes include Idle and Power-down
- Individual enable/disable of peripheral functions as well as peripheral clock scaling for additional power optimization.
- Processor wake-up from Power-down mode via external interrupt or BOD. Single power supply chip with POR and BOD circuits: CPU operating voltage range of 3.0 V to 3.6 V (3.3 V ± 10 %) with 5 V tolerant I/O.

1.1 Brief overview of ARM7 Architecture

The ARM7TDMI core is a 32-bit embedded RISC processor delivered as a hard macrocell optimized to provide the best combination of performance, power and area characteristics. The ARM7TDMI core enables system designers to build embedded devices requiring small size, low power and high performance. The ARM7 family also includes the ARM7TDMI processor, the ARM7TDMI-S processor, the ARM720T processor and the ARM7EJ- S processors, each of which has been developed to address different market requirements.

The market for microprocessors continues to diversify, based on the evolving demands of applications including wireless, home entertainment, automotive and microcontrollers. ARM core families sharing the ARMv7 architecture will cover the widening spectrum of embedded processing. The ARM architecture is based on Reduced Instruction Set Computer (RISC) principles. The RISC instruction set and related decode mechanism are much simpler than those of Complex Instruction Set Computer (CISC) design.



(1) Pins shared with GPIO.
(2) LPC2144/46/48 only.
(3) USB DMA controller with 8 kB of RAM accessible as general purpose RAM and/or DMA is available in LPC2146/48 only.
(4) LPC2142/44/46/48 only.

This simplicity gives:

· A high instruction throughput

· An excellence real-time interrupts response

· A small, cost-effective, processor macrocell

The ARM7TDMI core is the industry’s cost widely used 32-bit embedded RISC microprocessor solution. Optimized for cost and power-sensitive application, the ARM7TDMI solution provides low power consumption, small size, and high performance needed in portable, embedded application.

The ARM7DMI-S is synthesizable version of ARM7TDMI core. The ARM720T hard macrocell contain the ARM7DMI core, 8KB unified cache and MMU (Memory Management Unit) that allows the use of protected execution space and virtual memory.

The ARM7EJ-S processor is synthesizable core that provides all the benefit of ARM7DMI, while also incorporating ARM’s latest DSP extensions and jazelle technology, enabling acceleration of Java-based applications.

Architecture:

The ARM7 core is based on the von Neumann architecture with 32-bit data bus that carries both instruction and data. Data can be of 8 bits, 16 bits, 32 bits. It has following features:

- Instruction pipeline
- Memory format
- Operation modes
- Coprocessor
- Debugging feature

Instruction pipeline:

The ARMv7 core uses a three stage pipeline to increase the flow of instructions to the processor. This allows multiple simultaneous operations to take place and continuous operations and memory systems. The instructions are executed in three stages:

- Fetch
- Decode
- Execute

During normal operation, while one instruction is being executed, its successor is being decoded, and third instruction is being fetched from the memory. The program counter (PC) value used in an executing instruction is always two instructions ahead of the address.

Memory Format:

The ARM7 memory interface is design to allow optimum performance potential and minimize memory usage. Speed critical control signals are pipelined to allow system control function to exploit the fast burst access modes supported by many memory technologies. ARM7 has four basics types of cycle:

- Internal
- Non sequential
- Sequential
- Coprocessor transfer

The ARM7 can be configured to store the words as either in little-endian or big-endian format.

The ARM7 processor supports the following data types:

- Word, 32-bit
- Half word, 16-bit
- Byte, 8-bit

You must align this as follow:

- Word quantities must be aligned to four-byte boundaries.
- Half word quantities must be aligned to two-byte boundaries. · Byte quantities can be placed on any boundary.

The ARM core supports two operating states and instruction sets

- ARM state for 32 bit word aligned instruction

- Thumb state for 16-bit half word aligned instruction

Operating modes:

The ARMv7 core has seven modes of operation:

- User mode – normal ARM program execution mode and used for executing most application programs.
- Fast Interrupt (FIQ) – mode supports data transfer or channel processes to allow very fast interrupt
- Interrupt (IRQ) – mode is used for general purpose interrupt handling.
- Supervisor (SVC) – is protected mode for operating system.
- Abort (ABT) – mode is entered after a data or instruction fetch is aborted.
- Undefined (UND) – mode is entered when an undefined instruction is executed.
- System (SYS) – is a privileged user mode for the operating system.
- Modes other than user mode are collectively known as privileged modes. Privileged modes are used to service interrupts or exceptions, or to access protected resources.
- The ARMv7 has 37 register all are 32bit wide, not all the registers are available for a given modes. R15 is program counter. R14 is link register. R13 stack pointer.

CPSR – current program status register. SPSR – saved program status register.

Coprocessor:

Up to 16 coprocessors can be connected to an ARMv7 system. Coprocessors are separate processing unit that tightly coupled to the ARM processor. Typical coprocessor contains:

- An instruction pipeline
- Instruction decode logic
- Handshake logic
- A register bank
- Special processing logic with its own data path

Debugging Feature:

Internal state of the ARM core can be examined using a JTAG interface to allow the insertion of instructions into core pipeline and avoid using external data bus.
ARM7TDMI core includes an internal functional unit known as the Embedded ICE logic. The embedded ICE logic is configured to monitor the ARM7TDMI core actively for specific instruction fetches and data accesses.

Applications:

Using the ARMv7 architecture, ARM can strengthen its position as a low-power/performance leader while conquering new markets to carry its cores up in high performance and down in the low-cost high-volume domain of the microcontroller ARM designs the technology that lies at the heart of advanced digital products, from wireless, networking and consumer entertainment solutions to imaging, automotive, security and storage devices. ARM's comprehensive product offering includes 16/32-bit RISC microprocessors, data engines, 3D processors, digital libraries, embedded memories, peripherals, software and development tools, as well as analog functions and high-speed connectivity products.

1.3 ARM operating modes

The processor mode determines which registers are active and the access rights to the cpsr register itself. Each processor mode is either privileged or non privileged; A privileged mode allows full read-write access to the cpsr. Conversely, a non privileged mode only allows read access to the control field in the cpsr but still allows read-write access to the condition flags.

There are seven processor/operating modes in total: six privileged modes Abort mode

Fast interrupt request mode	
Interrupt request mode	Privileged
Supervisor mode	modes
System mode	
Undefined mode	
User mode	Nonprivileged mode

The processor enters

Abort mode	:	when there is a failed attempt to access memory.
Fast interrupt	:	Two interrupt levels
Interrupt	:	available on the ARM processor

Supervisor : The processor is in after reset and is generally the mode that an operating system kernel operates in.

System mode : A special version of user mode that allows full read-write access to the cpsr.

Undefined mode: When the processor encounters an instruction that is undefined or not supported by the implementation.

User mode : The mode is used for programs and applications.

1.4 System initialisation (Runtime Environment)

Most ARM applications begin by executing an assembly start up file. This

file could be linked to the bottom of the on-chip memory (Flash (0x0)or SRAM (0x400 0000)) depending from where the application is targeted to run.

The following should be covered in this startup file:

1. Interrupt Vector table

2. Stack pointers

3. Branch to Main

After the above basic assembly initialization code is executed, a branch is done to C main().

The following steps could be carried out in C code:

1. Enable the Memory Accelerator Module (MAM) if the application is run from on-chip Flash. It provides accelerated execution at higher frequencies and also helps in reducing power consumption. The MAM is only available in devices with on-chip Flash.
2. Set the System clock and peripheral clock. The system clock can be boosted using the PLL to 60 MHz or 75 MHz depending upon the input frequency. The peripheral clock can be set using the VPB Divider register . Please refer to Application note AN10331 to get detailed information about the PLL.

3. Set the Memory Mapping Control register (MEMMAP at address 0xE01F C040) accordingly. The MEMMAP register gives the application the

   flexibility of executing interrupts from different memory regions. For instance, if MEMAP is set to 0x2, the

   Interrupt vectors would be mapped to 0x4000 0000 (bottom of on-chip SRAM).

4. Disable unused peripherals using the Power Control for Peripherals register (PCONP at address 0xE01F C0C4).
5. Configure GPIO's using the respective IODIR, IOSET and IOCLR registers. On the LPC2000, there are certain pins that should not be held low on reset. For instance, by driving P0.14 low on reset would make the on-chip bootloader to take control of the part after reset.

6. Depending upon the peripherals being used set the port functions accordingly using the appropriate Pin Function Select register (PINSELx).

7. Initialize the peripherals which are still enabled in PCONP and enable interrupts for them if needed.

8. Configure the Vectored Interrupt Controller (VIC) to handle the different interrupt sources and classify them as IRQ and FIQ. It is recommended that only one interrupt source should be classified as an FIQ.

9. It is always safe to program the Default Vector Address Register (VICDefVectAddr) with a dummy ISR address wherein the VIC would be updated

(by performing a write operation on the VIC Vector Address register (VICVectAddr) to avoid any spurious interrupts.

1.5 Arm Board description

Power:

DC 6.5V with power LEDOn-board linear regulators generate +3.3V/500mA and +5v/500mA from power supply. USB connector (as alternate power source).

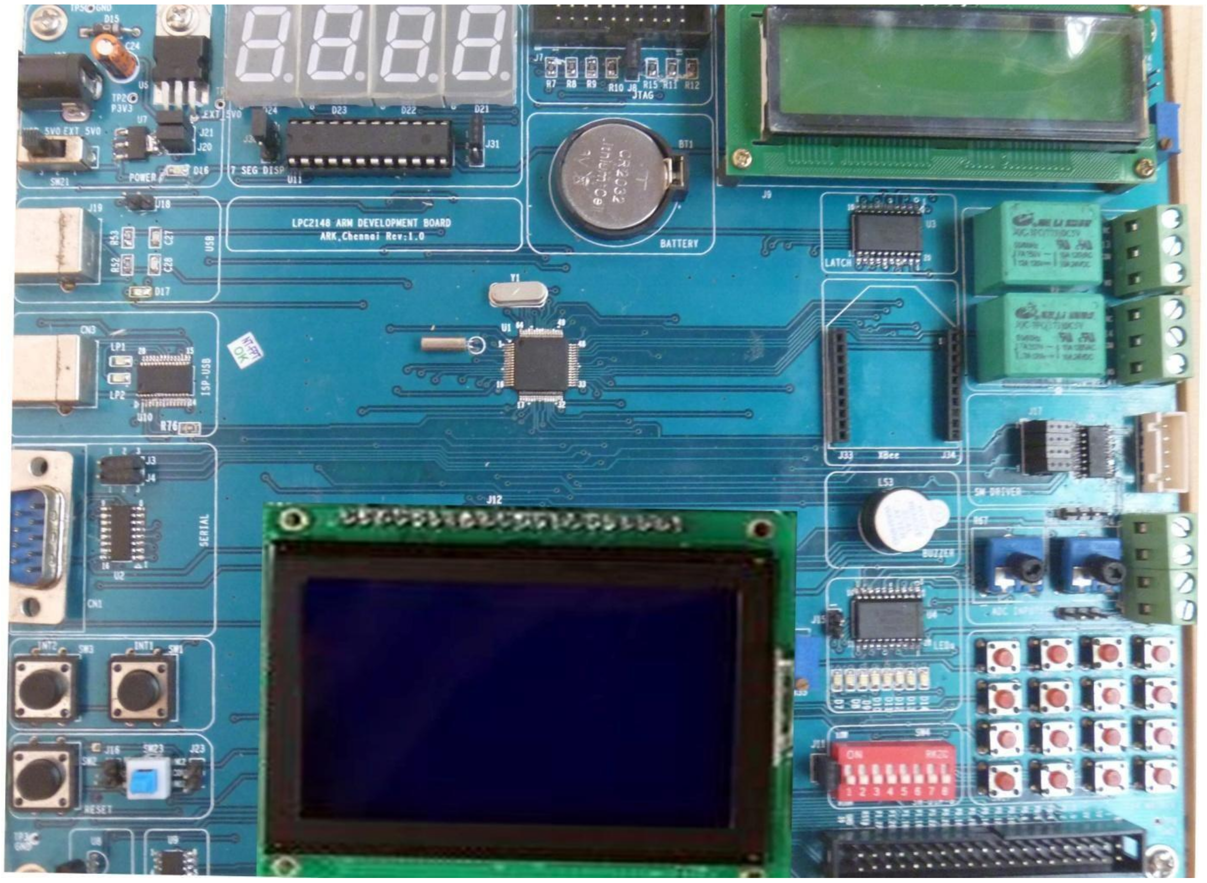
Connectors:

- Extension headers for all microcontroller pins.
- RS232 connector.
- VGA connector.
- PS/2 connector.
- JTAG connector.
- USB B-type connector with Link-LED.
- All peripheral configurable via jumpers

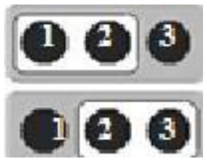
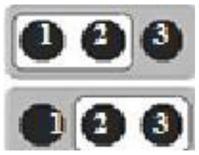
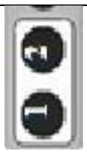
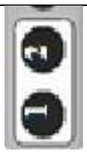
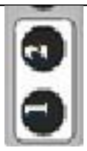
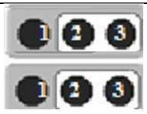
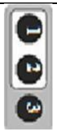
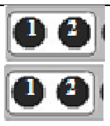
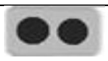
Other Peripherals:

- 256Kb I2C based EEPROM .
- 2 line X 16 character LCD with back light control.
- Configurable for manual and automatic program download (ISP) via serial port.
- 8 controllable LEDs on SPI using 74HC595.

LPC2148 (ARM7)



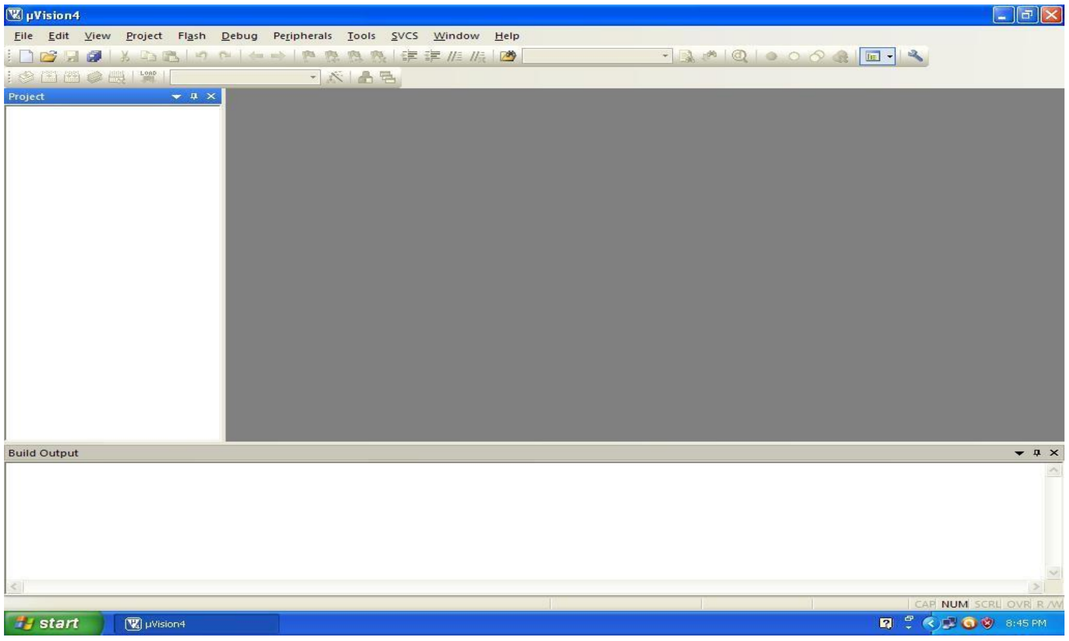
Modules and Jumpers Relationship:

Analog I/P (P0.29) **J24**		On-Board Analog Input External Analog Input-1 select
Analog I/P (P0.30) **J25**		On-Board Analog Input External Analog Input-1 select
J8		Disable JTAG Power
J15		LED driver selection
J11		DIP switch Enable
J3 & J4		Serial port Enable
J30 & J31		7 Segment Display Enable
J20 & J21		Enable Power conversion from 5v to 3.3v
J10, J16, J19 & J23		Keep it in NO condition

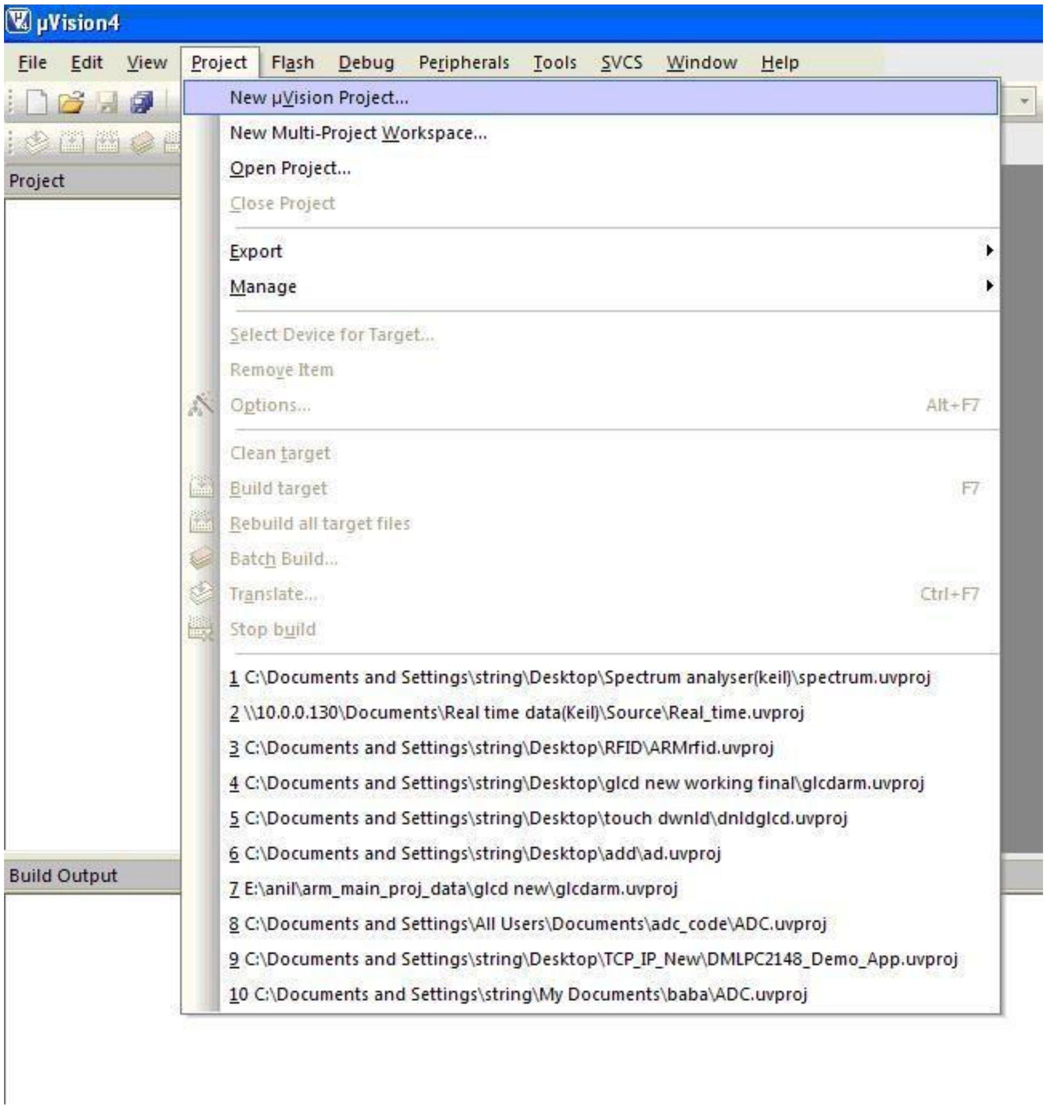
- **How to work with keil?**

2.1.1 How to create a new µProject?

Step 1: Give a double click on µvision 4 icon on the desk top, it will generate a window as shown below.

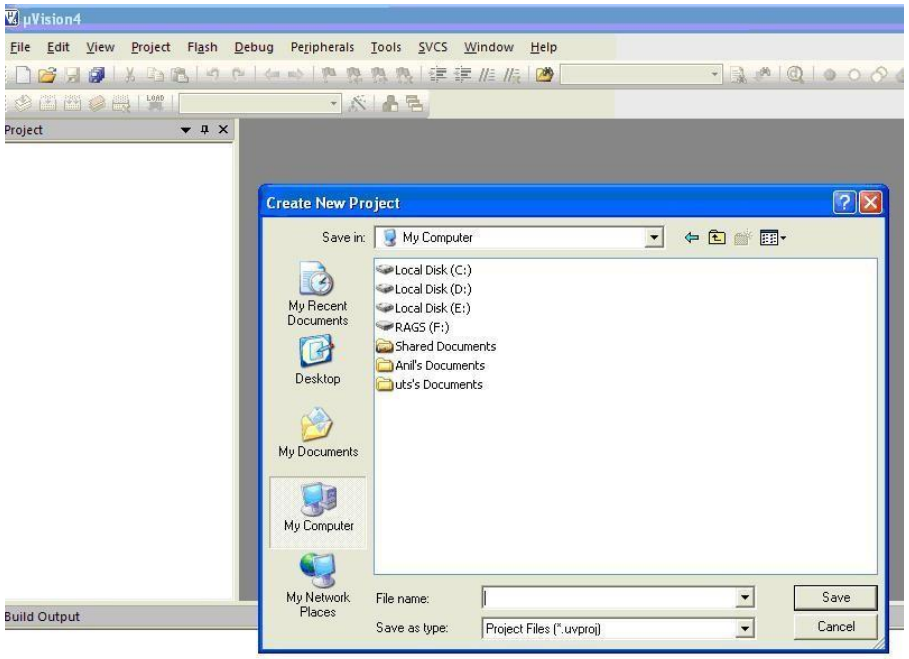

µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project
Build Output
CAP NUM SCRL OVR R/W
start
µVision4
8:45 PM

Step 2: To create new project go to project select new micro vision project.



1 C:\Documents and Settings\string\Desktop\Spectrum analyser(keil)\spectrum.uvproj
2 \\10.0.0.130\Documents\Real time data(Keil)\Source\Real_time.uvproj
3 C:\Documents and Settings\string\Desktop\RFID\ARMrfid.uvproj
4 C:\Documents and Settings\string\Desktop\glcd new working final\glcdarm.uvproj
5 C:\Documents and Settings\string\Desktop\touch dwnld\dnldglcd.uvproj
6 C:\Documents and Settings\string\Desktop\add\ad.uvproj
7 E:\anil\arm_main_proj_data\glcd new\glcdarm.uvproj
8 C:\Documents and Settings\All Users\Documents\adc_code\ADC.uvproj
9 C:\Documents and Settings\string\Desktop\TCP_IP_New\DMLPC2148_Demo_App.uvproj
10 C:\Documents and Settings\string\My Documents\baba\ADC.uvproj

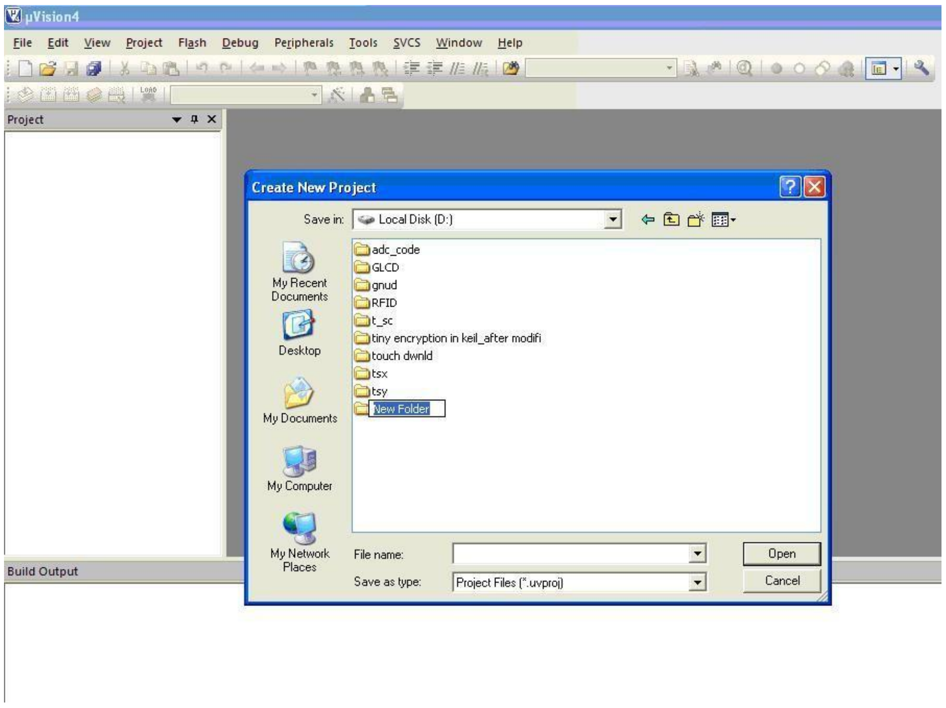
Step 3: select a drive where you would like to create your project.



Save in: My Computer

Save as type: Project Files (*.uvproj)

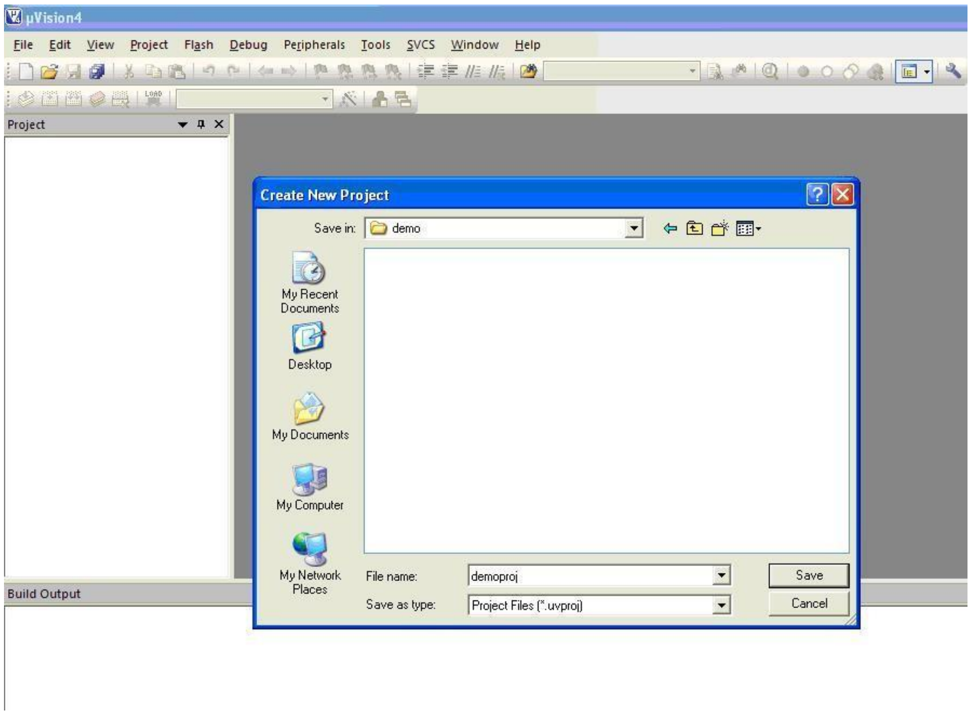
Step 4: Create a new folder and name it with your project name.



Save in: Local Disk (D:)

Save as type: Project Files (*.uvproj)

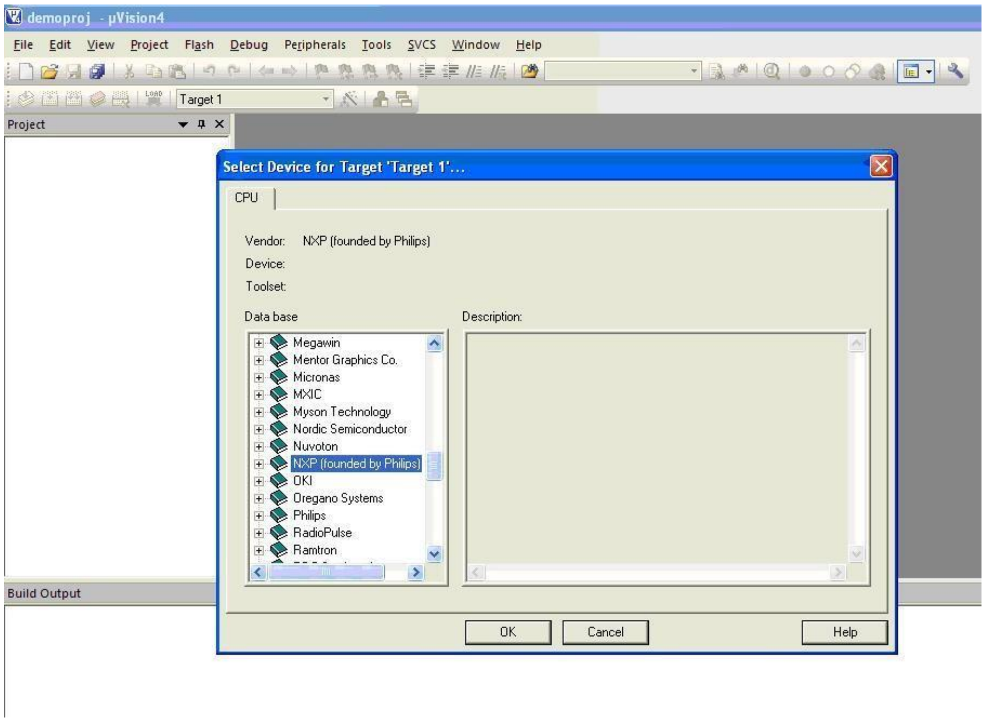
Step 5: Open that project folder and give a name of your project executable file and save it.



File name: demoproj

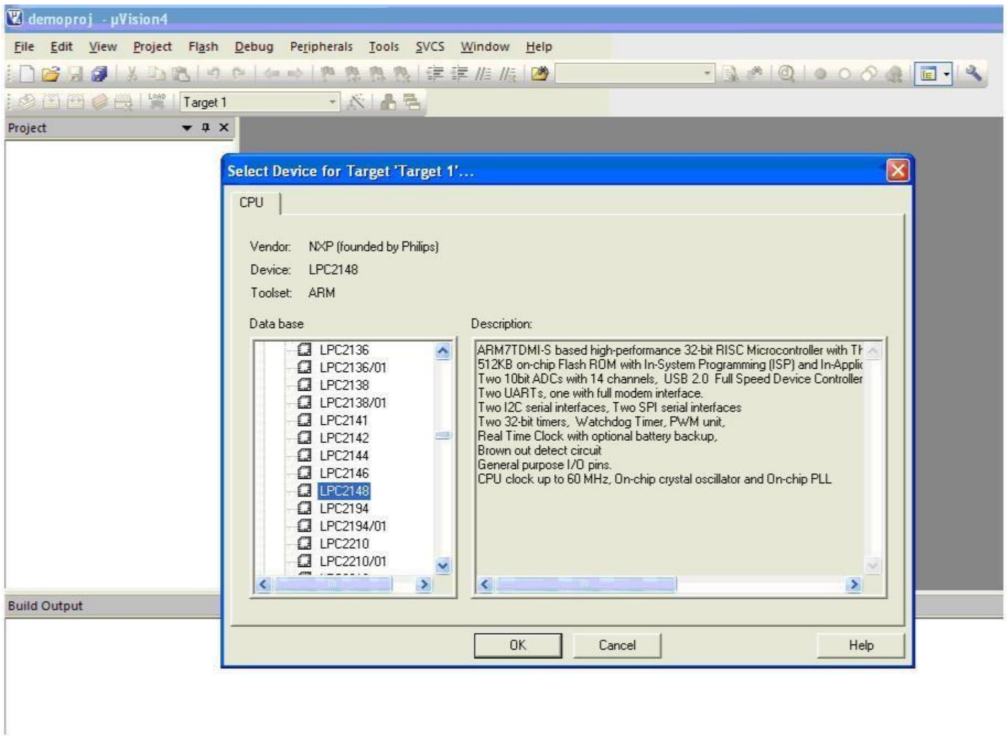
Save as type: Project Files (*.uvproj)

Step 6: After saving it will show some window there you select your microcontroller company i.e NXP from Phillips.



Vendor: NXP (founded by Philips)
Device:
Toolset:
Data base
Description:

Step 7: Select your chip as LPC2148



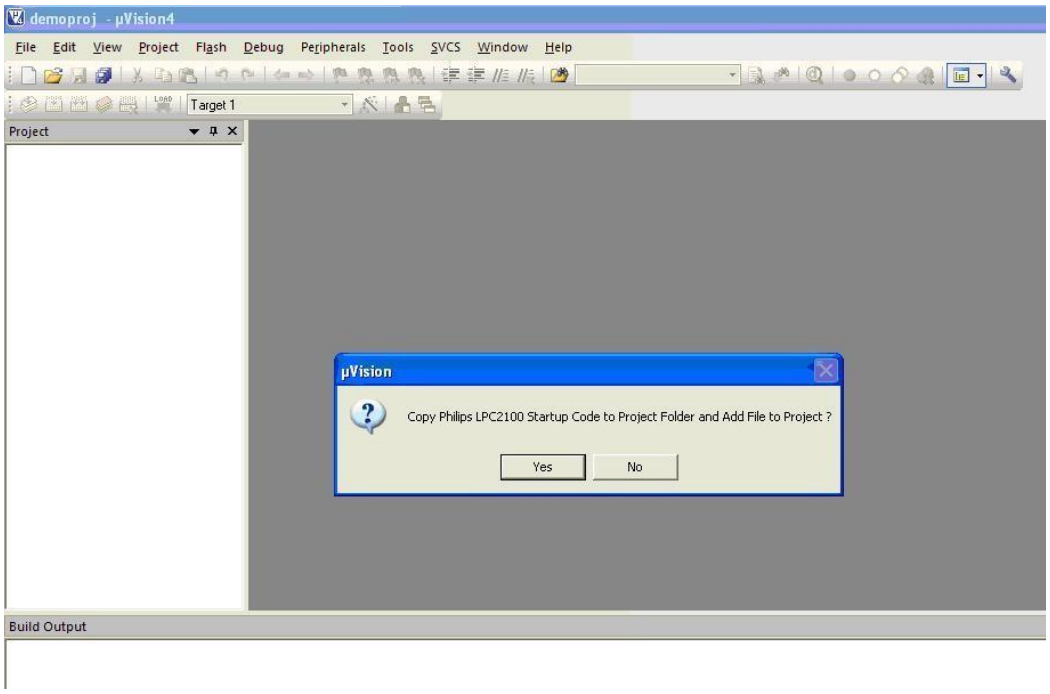
Select Device for Target 'Target 1'...

Vendor: NXP (founded by Philips)
Device: LPC2148
Toolset: ARM

Description:

```
ARM7TDMI-S based high-performance 32-bit RISC Microcontroller with Th
512KB on-chip Flash ROM with In-System Programming (ISP) and In-Applic
Two 10bit ADCs with 14 channels,  USB 2.0  Full Speed Device Controller
Two UARTs, one with full modem interface.
Two I2C serial interfaces, Two SPI serial interfaces
Two 32-bit timers,  Watchdog Timer, PWM unit,
Real Time Clock with optional battery backup,
Brown out detect circuit
General purpose I/O pins.
CPU clock up to 60 MHz, On-chip crystal oscillator and On-chip PLL
```

Step 8: After selecting chip click on OK then it will display some window asking to add STARTUP file. Select YES.

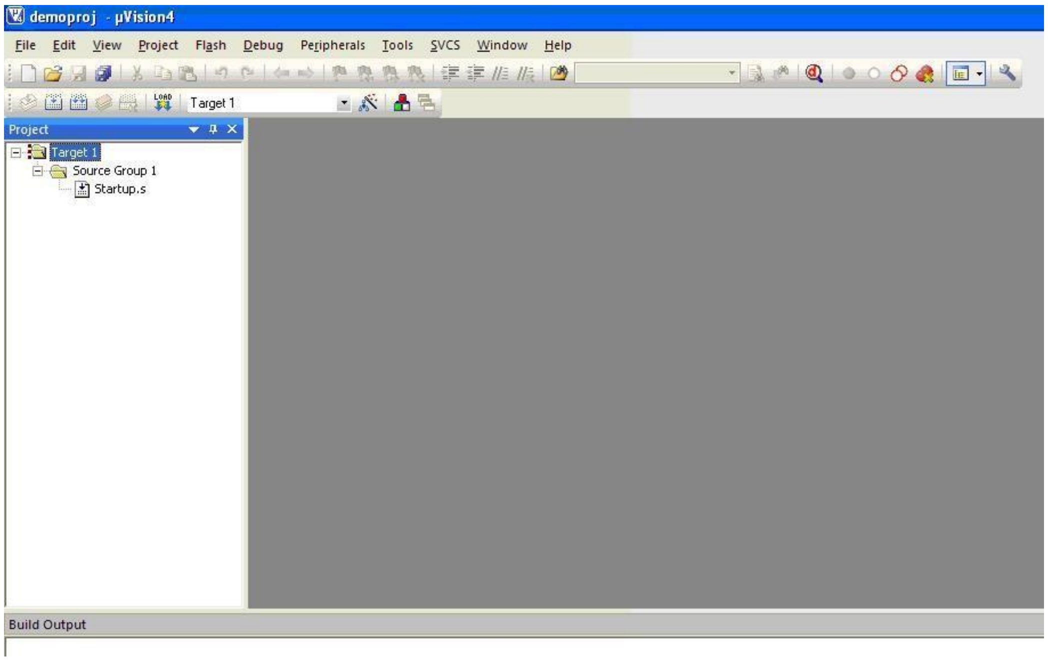


µVision

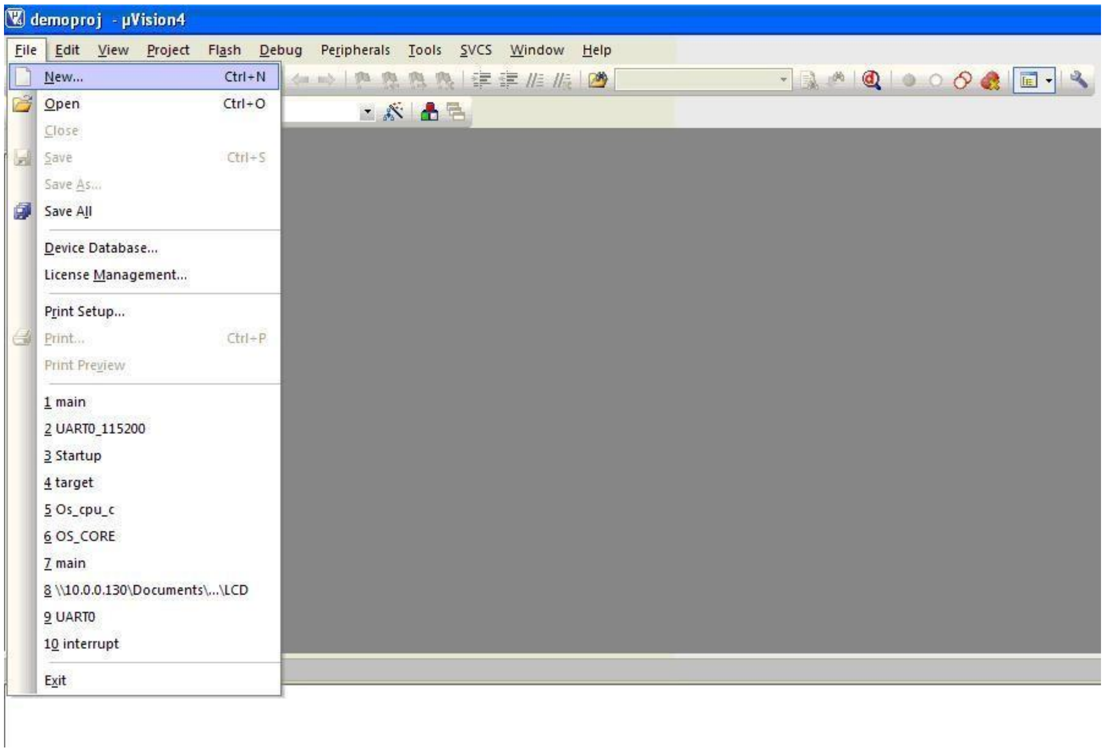
Copy Philips LPC2100 Startup Code to Project Folder and Add File to Project ?

demoproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Build Output
Yes
No

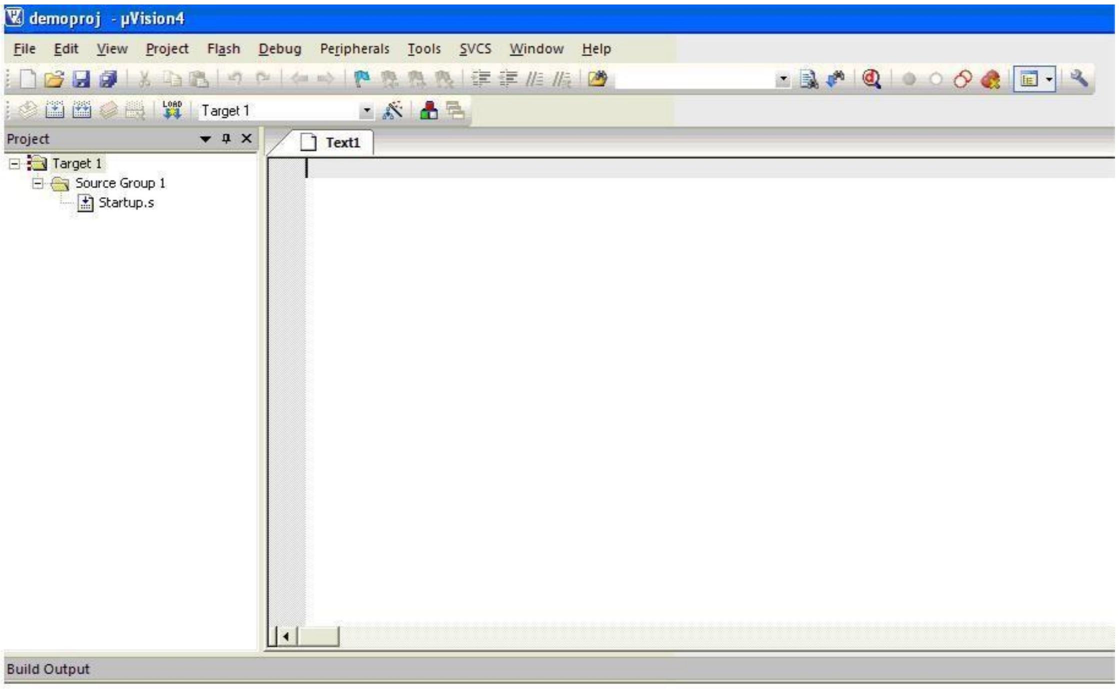
Step 9: A target is created and startup filoe is added to your project target and is shown below.



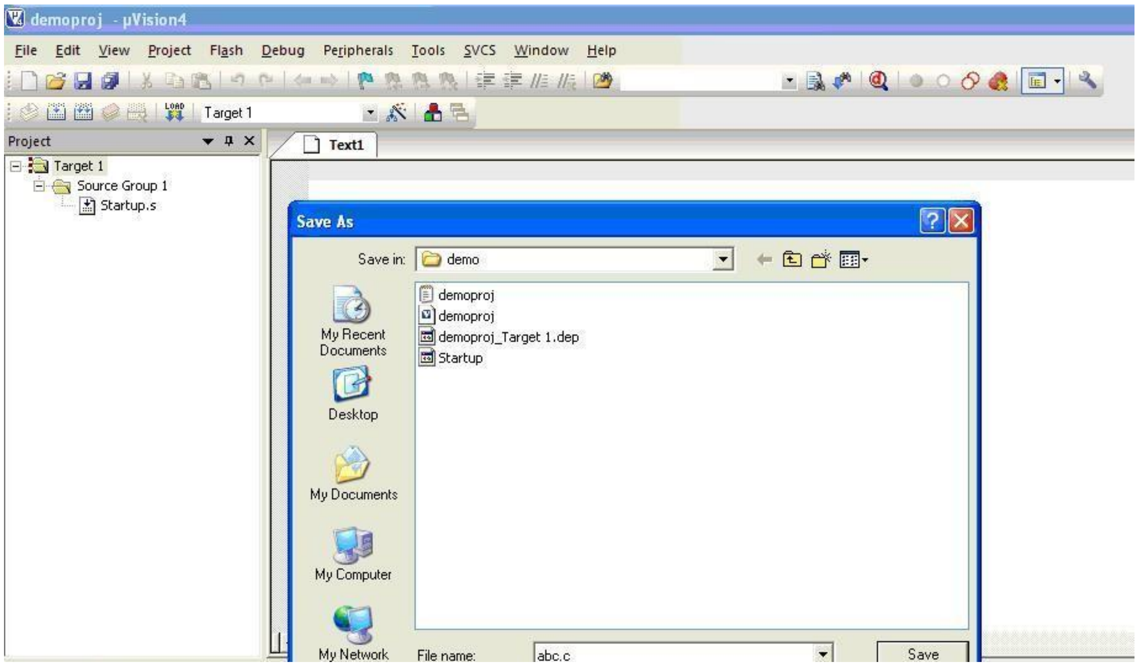
Step 10: To write your project code select a new file from FILE menu bar.



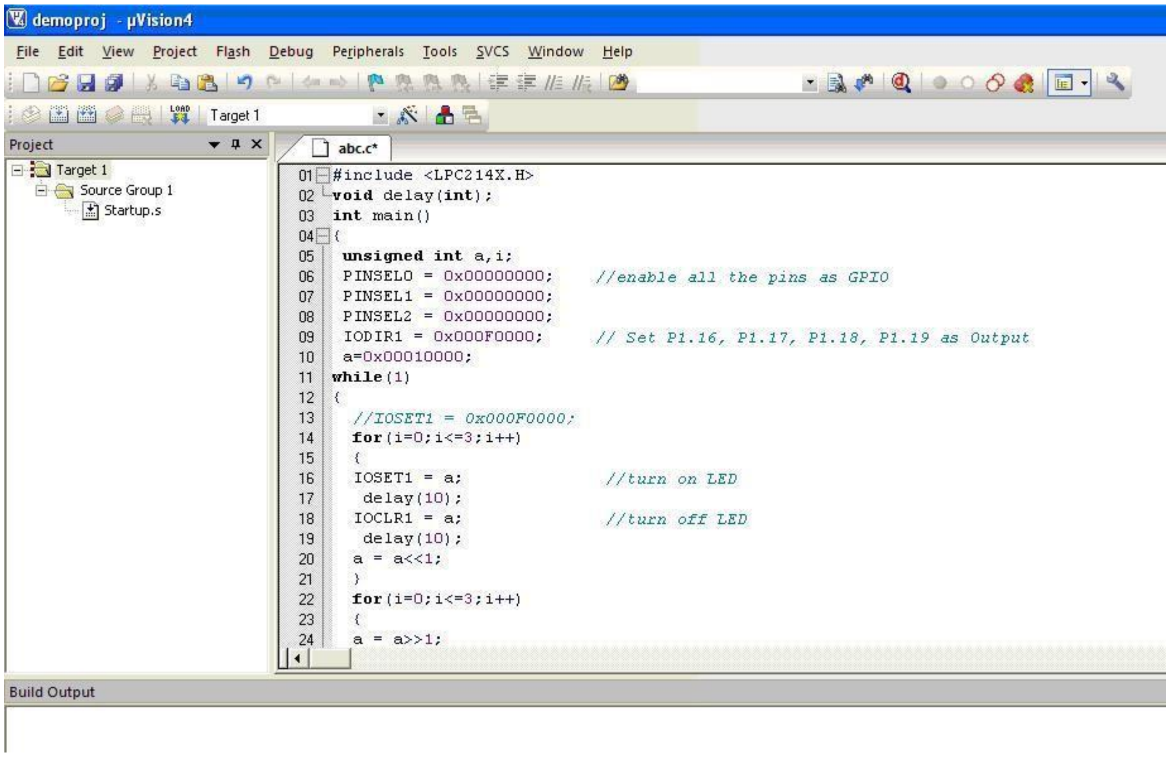
Step 11: It will display some text editor, to save that file select SAVE option from FILE menu bar.



Step 12: By giving a file name lwith extension .C for c files and save it.

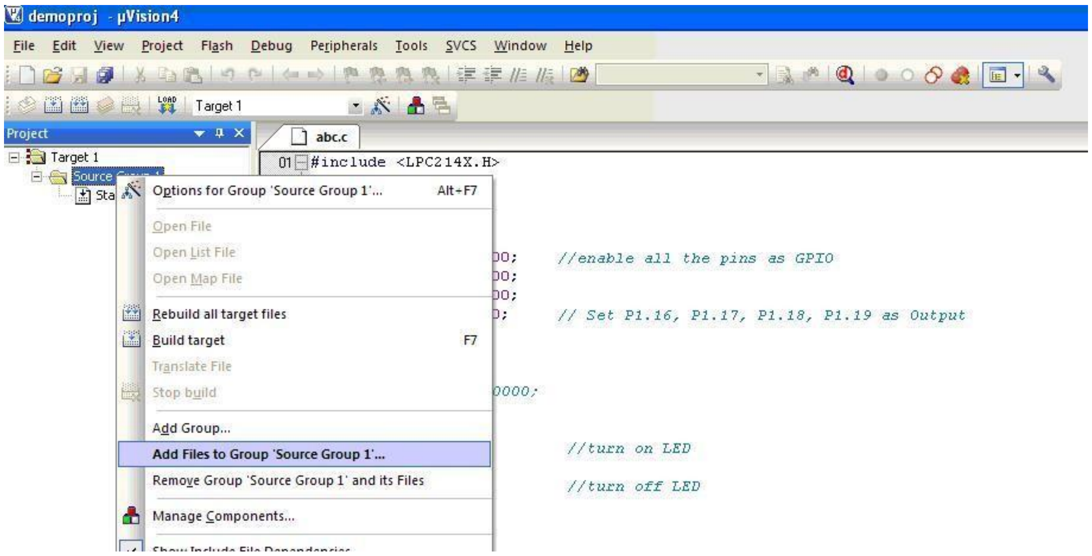


Step 13: Write the code of your projct and save it.



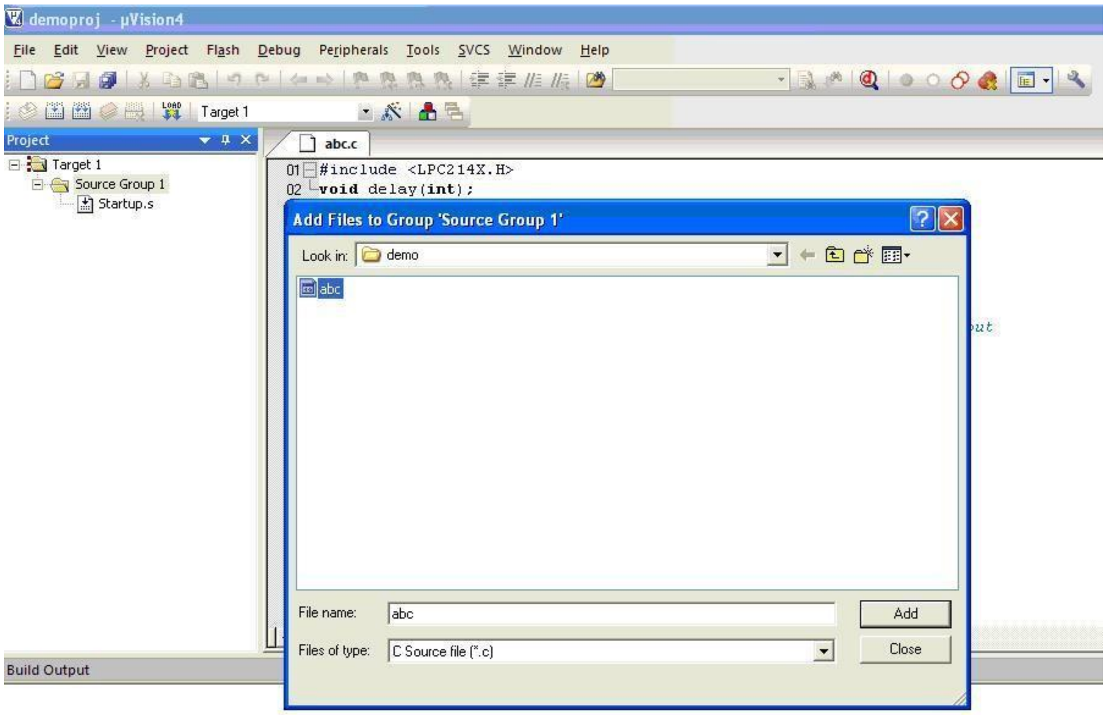
```c
#include <LPC214X.H>
void delay(int);
int main()
{
 unsigned int a,i;
 PINSEL0 = 0x00000000;    //enable all the pins as GPIO
 PINSEL1 = 0x00000000;
 PINSEL2 = 0x00000000;
 IODIR1 = 0x000F0000;     // Set P1.16, P1.17, P1.18, P1.19 as Output
 a=0x00010000;
while(1)
{
  //IOSET1 = 0x000F0000;
  for(i=0;i<=3;i++)
  {
  IOSET1 = a;               //turn on LED
   delay(10);
  IOCLR1 = a;               //turn off LED
   delay(10);
  a = a<<1;
  }
  for(i=0;i<=3;i++)
  {
  a = a>>1;
```

Step 14: To add our c file to target give a right click on Source Group, choose “ADD files to Group” option.


demoproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source
Sta
abc.c
Options for Group 'Source Group 1'... Alt+F7
Open File
Open List File
Open Map File
Rebuild all target files
Build target F7
Translate File
Stop build
Add Group...
Add Files to Group 'Source Group 1'...
Remove Group 'Source Group 1' and its Files
Manage Components...

```c
#include <LPC214X.H>
//enable all the pins as GPIO
// Set P1.16, P1.17, P1.18, P1.19 as Output
//turn on LED
//turn off LED
```

Step 15: It will displays some window there select the file you have to add and click on ADD option.



```
#include <LPC214X.H>
void delay(int);
```

Add Files to Group 'Source Group 1'

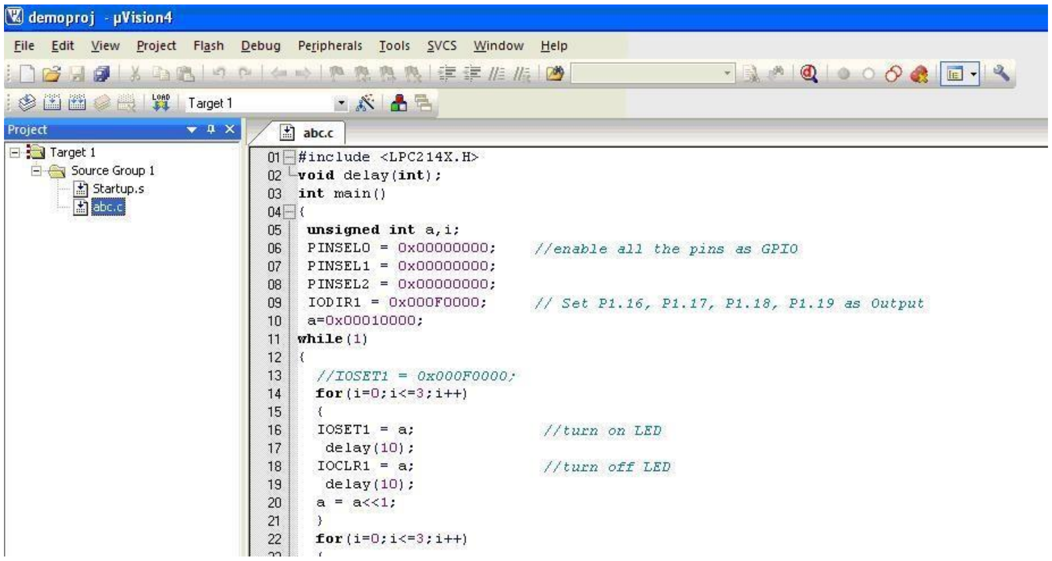
Look in: demo

abc

File name: abc

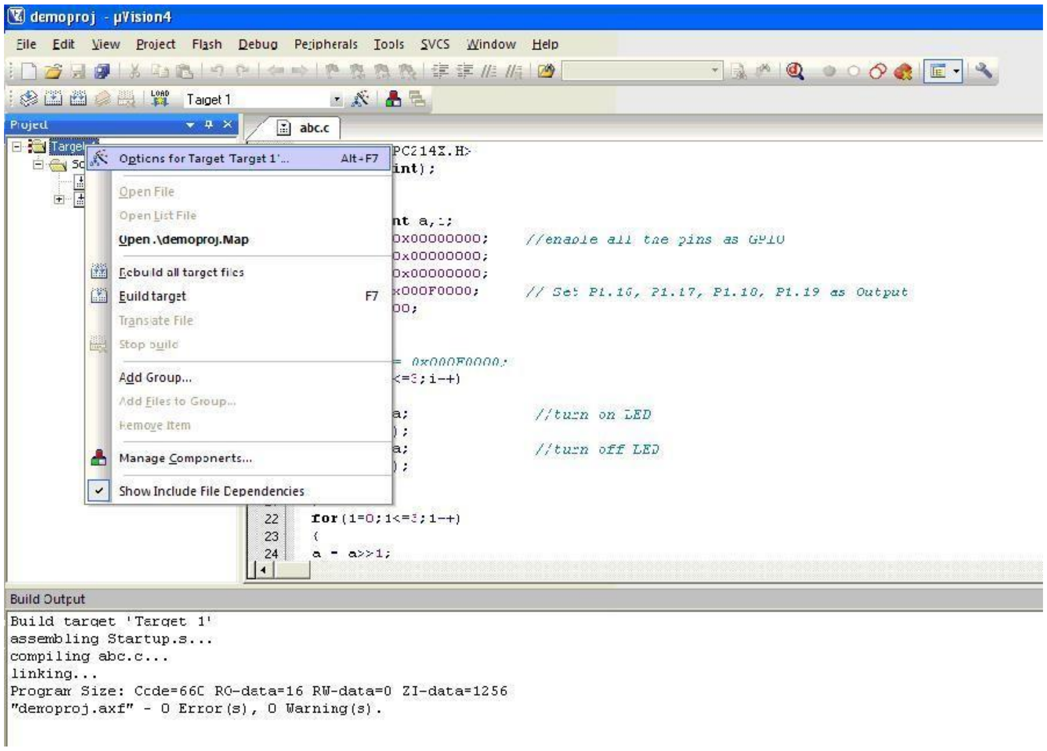
Files of type: C Source file (*.c)

Step 16: The file will be added to our target and it shows in the project window.



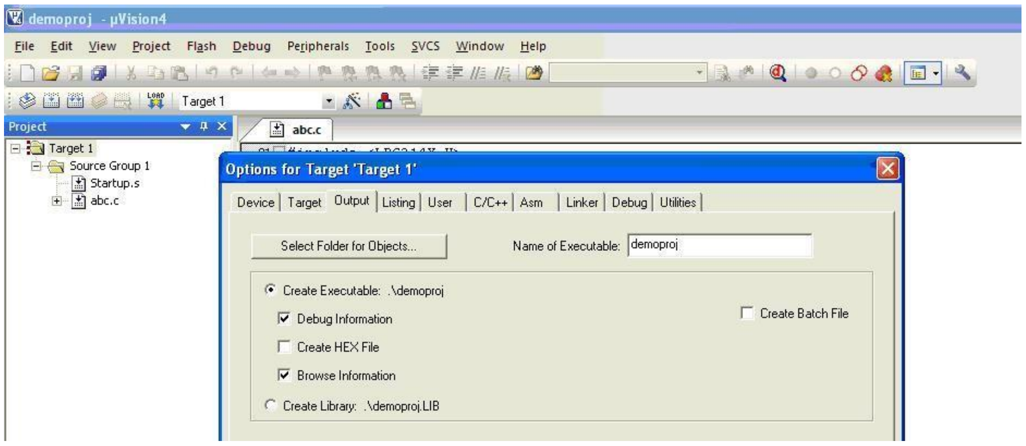
```
#include <LPC214X.H>
void delay(int);
int main()
{
 unsigned int a,i;
 PINSEL0 = 0x00000000;     //enable all the pins as GPIO
 PINSEL1 = 0x00000000;
 PINSEL2 = 0x00000000;
 IODIR1 = 0x000F0000;      // Set P1.16, P1.17, P1.18, P1.19 as Output
 a=0x00010000;
while(1)
{
  //IOSET1 = 0x000F0000;
  for(i=0;i<=3;i++)
  {
  IOSET1 = a;               //turn on LED
   delay(10);
  IOCLR1 = a;               //turn off LED
   delay(10);
  a = a<<1;
  }
  for(i=0;i<=3;i++)
```

Step 17: Now give a right click on target in the project window and select "Options for Target".



```
Build target 'Target 1'
assembling Startup.s...
compiling abc.c...
linking...
Program Size: Code=660 RO-data=16 RW-data=0 ZI-data=1256
"demoproj.axf" - 0 Error(s), 0 Warning(s).
```

Step 18: It will shoe some window, in that go to output option and choose Create Hex file option by selecting that box.



Name of Executable: demoproj

(●) Create Executable: .\demoproj

[✓] Debug Information

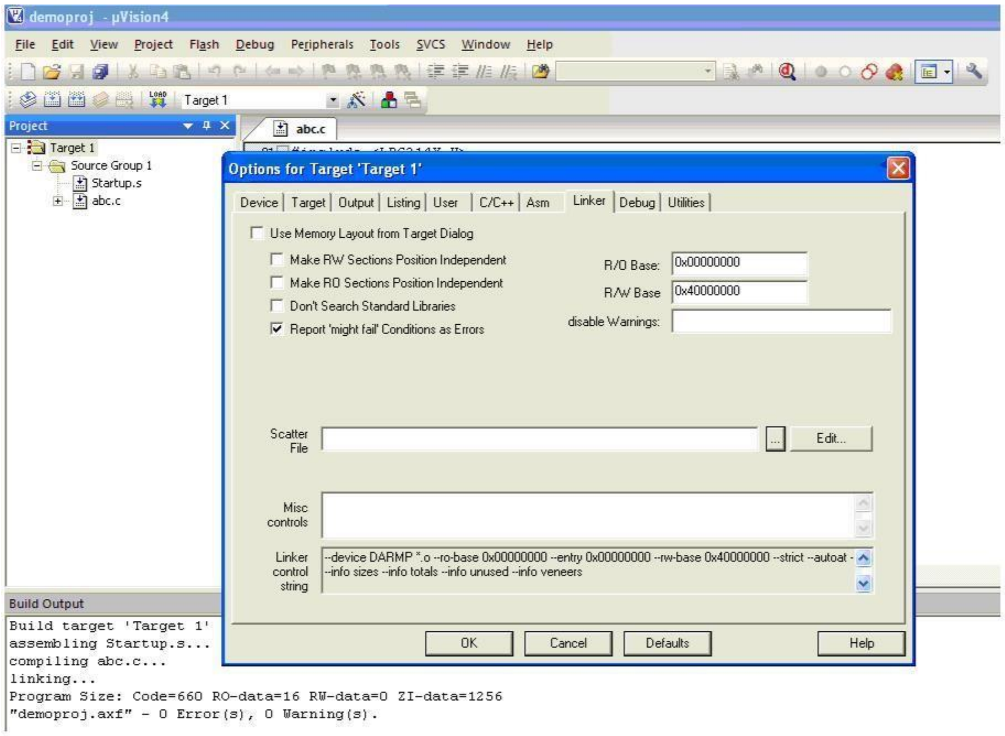
[ ] Create HEX File

[✓] Browse Information

( ) Create Library: .\demoproj.LIB

[ ] Create Batch File

Step 19: In the same window go to Linker option and choose Use Memory Layout from Target Dialog by selecting the box, and click OK.


demoproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
abc.c
Target 1
Source Group 1
Startup.s
abc.c
Options for Target 'Target 1'
Device Target Output Listing User C/C++ Asm Linker Debug Utilities
Scatter File
Edit...
Misc controls
OK Cancel Defaults Help
Build Output

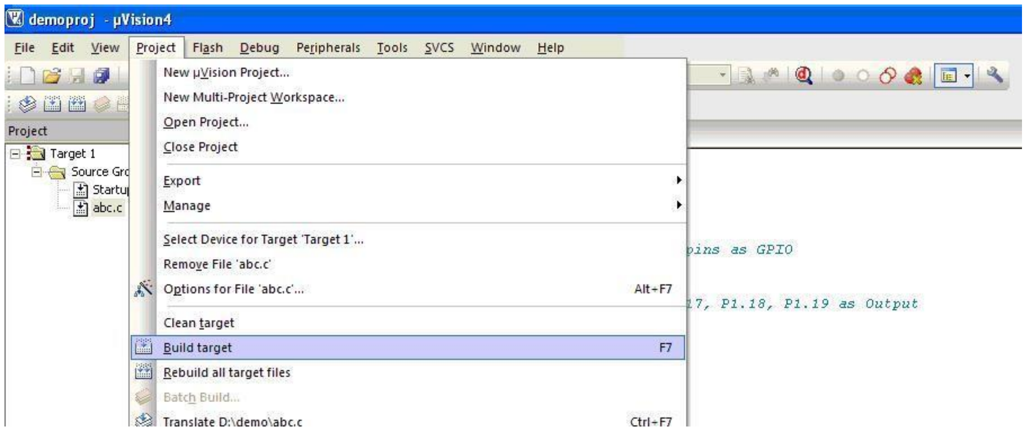
Use Memory Layout from Target Dialog
Make RW Sections Position Independent
Make RO Sections Position Independent
Don't Search Standard Libraries
✓ Report 'might fail' Conditions as Errors

R/O Base: 0x00000000
R/W Base 0x40000000
disable Warnings:

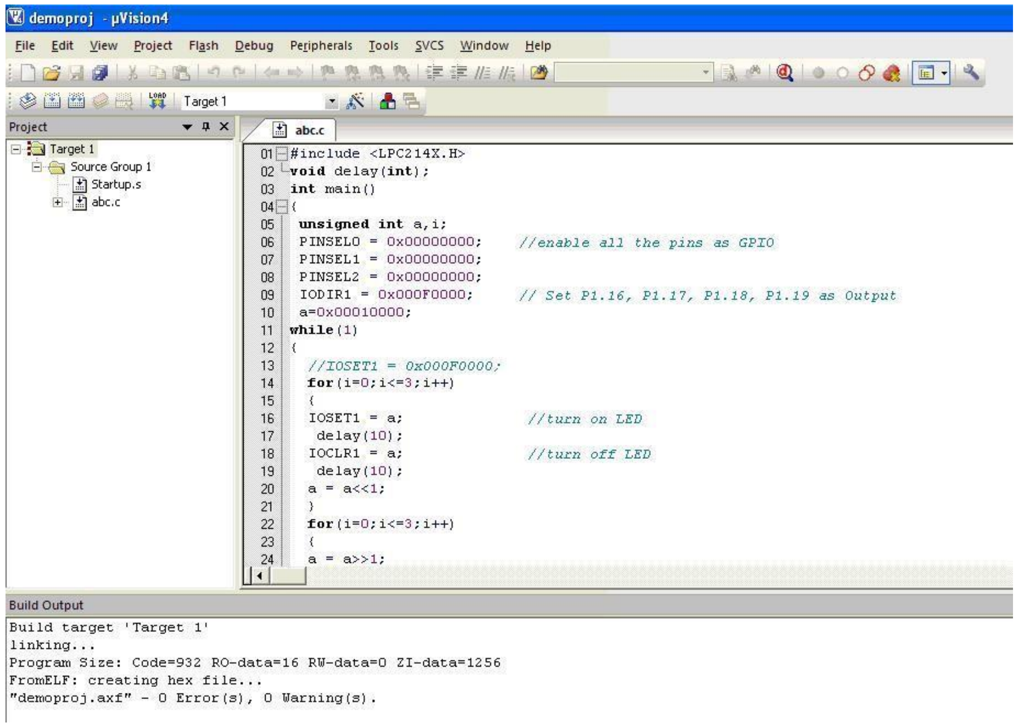
Linker control string: --device DARMP *.o --ro-base 0x00000000 --entry 0x00000000 --rw-base 0x40000000 --strict --autoat --info sizes --info totals --info unused --info veneers

```
Build target 'Target 1'
assembling Startup.s...
compiling abc.c...
linking...
Program Size: Code=660 RO-data=16 RW-data=0 ZI-data=1256
"demoproj.axf" - 0 Error(s), 0 Warning(s).
```

Step 20: Now to Compile your project go to Project select Build Target option or press F7.


demoproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project
Target 1
Source Gro
Startu
abc.c
New µVision Project...
New Multi-Project Workspace...
Open Project...
Close Project
Export
Manage
Select Device for Target 'Target 1'...
Remove File 'abc.c'
Options for File 'abc.c'... Alt+F7
Clean target
Build target F7
Rebuild all target files
Batch Build...
Translate D:\demo\abc.c Ctrl+F7
pins as GPIO
17, P1.18, P1.19 as Output

Step 21: In the build OUT PUT window you can see the errors and warnings if there in your code. And here Your project Hex file will be created.



```
#include <LPC214X.H>
void delay(int);
int main()
{
 unsigned int a,i;
 PINSEL0 = 0x00000000;    //enable all the pins as GPIO
 PINSEL1 = 0x00000000;
 PINSEL2 = 0x00000000;
 IODIR1 = 0x000F0000;     // Set P1.16, P1.17, P1.18, P1.19 as Output
 a=0x00010000;
while(1)
{
  //IOSET1 = 0x000F0000;
  for(i=0;i<=3;i++)
  {
  IOSET1 = a;              //turn on LED
   delay(10);
  IOCLR1 = a;              //turn off LED
   delay(10);
  a = a<<1;
  }
  for(i=0;i<=3;i++)
  {
  a = a>>1;
```

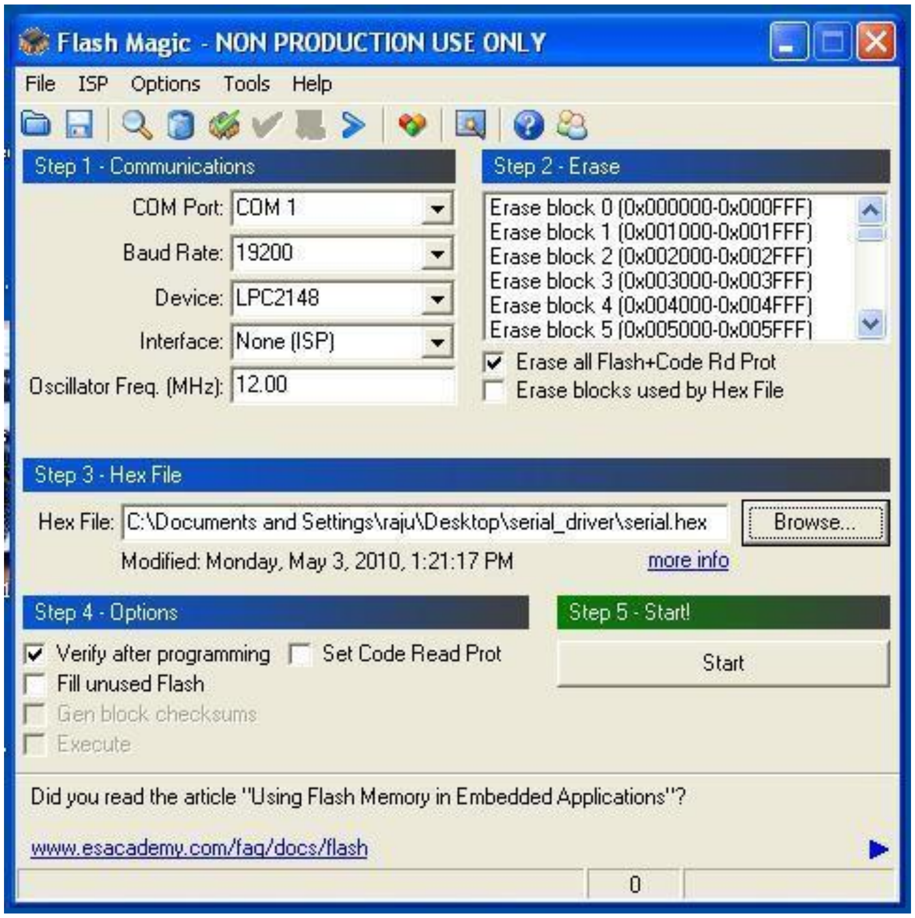
```
Build target 'Target 1'
linking...
Program Size: Code=932 RO-data=16 RW-data=0 ZI-data=1256
FromELF: creating hex file...
"demoproj.axf" - 0 Error(s), 0 Warning(s).
```

Downloading Hex file onto ARM microcontroller

3.1 Flash Magic Tool

To program the Microcontroller, Flash Magic tool is used. Generally, the microcontroller is in one of the two modes. One is RUN mode and the other is PROGRAMMING mode. In RUN mode microcontroller executes the application present in the microcontroller flash memory. In PROGRAMMING mode, microcontroller programs its flash memory in synchronization with Flash Magic.

To enter in to the programming mode, Hold down SW2(isp) and SW3(reset), then release SW3 first and finally SW2 . To enter in to Run Mode,press the SW3(reset) after programming is over.



COM Port: COM 1
Baud Rate: 19200
Device: LPC2148
Interface: None (ISP)
Oscillator Freq. (MHz): 12.00

Erase block 0 (0x000000-0x000FFF)
Erase block 1 (0x001000-0x001FFF)
Erase block 2 (0x002000-0x002FFF)
Erase block 3 (0x003000-0x003FFF)
Erase block 4 (0x004000-0x004FFF)
Erase block 5 (0x005000-0x005FFF)

[✓] Erase all Flash+Code Rd Prot
[ ] Erase blocks used by Hex File

Hex File: C:\Documents and Settings\raju\Desktop\serial_driver\serial.hex
Modified: Monday, May 3, 2010, 1:21:17 PM

[✓] Verify after programming
[ ] Set Code Read Prot
[ ] Fill unused Flash
[ ] Gen block checksums
[ ] Execute

Did you read the article "Using Flash Memory in Embedded Applications"?

www.esacademy.com/faq/docs/flash

Snapshot of the Flash Magic Tool.

3.2 Downloading Hex file onto microcontroller

To program the flash memory, first keep the microcontroller in PROGRAMMING mode. Launch the Flash Magic Tool. Select the COM1, Baud rate as 19200, device as LPC2148; Oscillator Freq (MHz) as 12, in Communication block. Select the box erase all Flash + Code Rd Prot in Erase block. Select the box Verify after programming in Options Block. Select the hex file in Hex File block. Hold down SW2 (isp) and SW3 (reset), then release SW3 first and finally SW2 .Then click Start Button in Start Block.

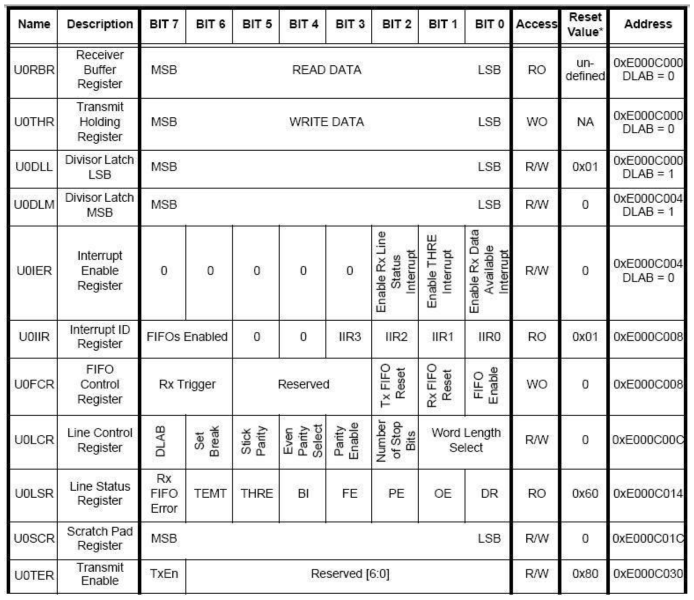
Serial Communication Drivers for ARM Processors

5.1 Study of serial communication architecture on LPC2148 ARM architecture

Introduction: LPC2148 microcontroller has two UARTs. The microcontroller has to communicate to external world through the UARTs only. There is an importance for developing serial port driver.

- 16 byte Receive and Transmit FIFOs.
- Register locations conform to '550 industry standard.
- Receiver FIFO trigger points at 1, 4, 8, and 14 bytes.
- Built-in baud rate generator.
- LPC2148 contains mechanism that enables software flow control implementation.

The below fig. shows the register mapping. For complete description of individual register refer to user guide.



Name	Description	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	Access	Reset Value*	Address
U0RBR	Receiver Buffer Register	MSB			READ DATA				LSB	RO	un-defined	0xE000C000 DLAB = 0
U0THR	Transmit Holding Register	MSB			WRITE DATA				LSB	WO	NA	0xE000C000 DLAB = 0
U0DLL	Divisor Latch LSB	MSB							LSB	R/W	0x01	0xE000C000 DLAB = 1
U0DLM	Divisor Latch MSB	MSB							LSB	R/W	0	0xE000C004 DLAB = 1
U0IER	Interrupt Enable Register	0	0	0	0	0	Enable Rx Line Status Interrupt	Enable THRE Interrupt	Enable Rx Data Available Interrupt	R/W	0	0xE000C004 DLAB = 0
U0IIR	Interrupt ID Register	FIFOs Enabled		0	0	IIR3	IIR2	IIR1	IIR0	RO	0x01	0xE000C008
U0FCR	FIFO Control Register	Rx Trigger		Reserved			Tx FIFO Reset	Rx FIFO Reset	FIFO Enable	WO	0	0xE000C008
U0LCR	Line Control Register	DLAB	Set Break	Stick Parity	Even Parity Select	Parity Enable	Number of Stop Bits	Word Length Select		R/W	0	0xE000C00C
U0LSR	Line Status Register	Rx FIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60	0xE000C014
U0SCR	Scratch Pad Register	MSB							LSB	R/W	0	0xE000C01C
U0TER	Transmit Enable	TxEn	Reserved [6:0]							R/W	0x80	0xE000C030

5.2 Implementing serial communication drivers for ARM processors

The major steps in carrying out the experiment:

i) write a c program.

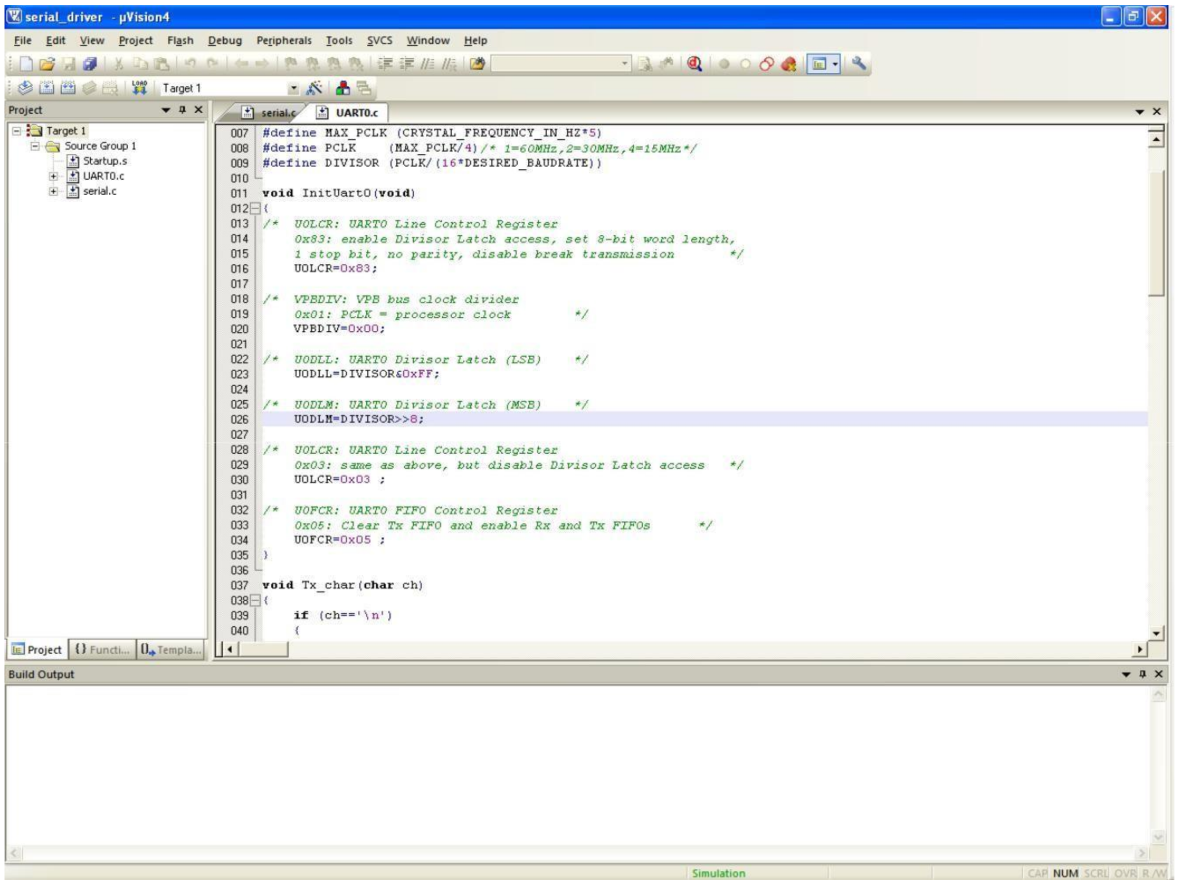
ii) Build the application.

iii) Download the hex file on to the ARM board and check the result .

Writing the program:

Create a new folder as **serial_driver.** In the **serial_driver** create a folder called **src.**

To write source code, you have to open keil software and can write program in new file. While saving the file save it with .c extension. The program code can be seen in figure below.



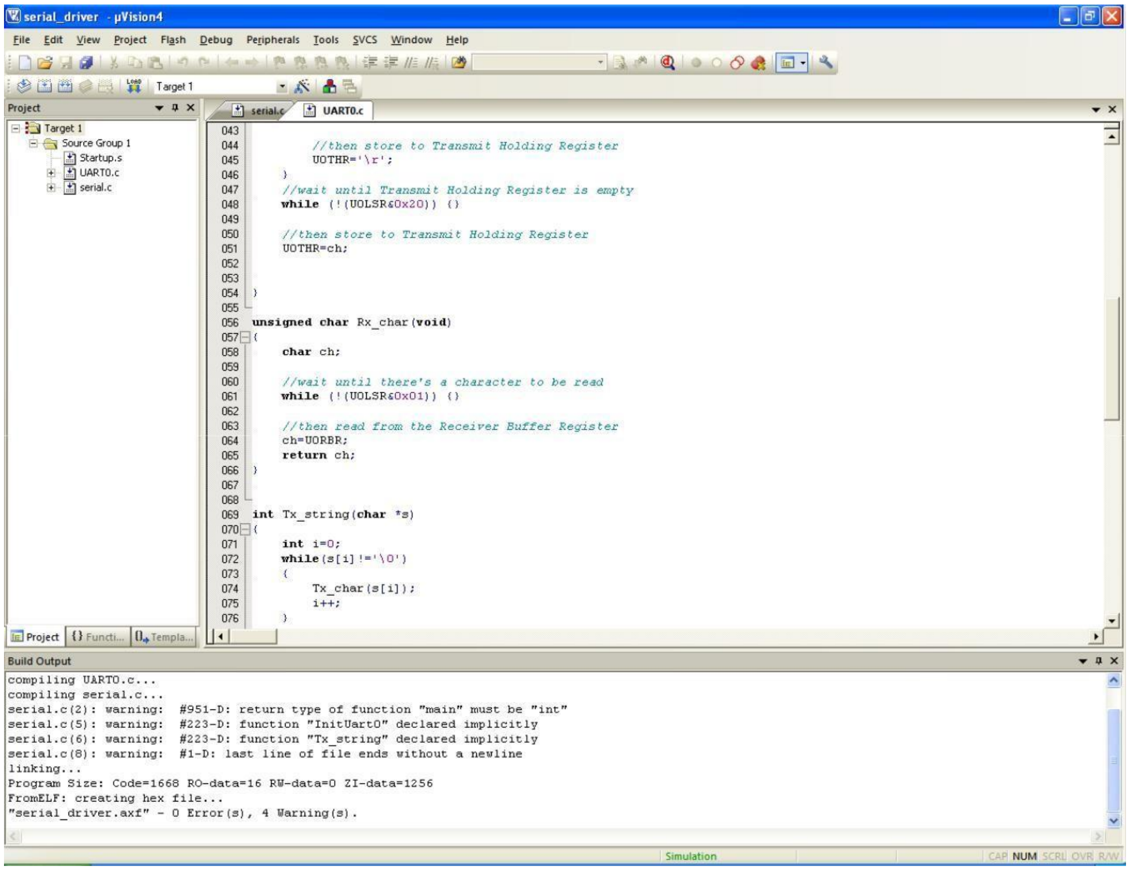
```
#define MAX_PCLK (CRYSTAL_FREQUENCY_IN_HZ*5)
#define PCLK     (MAX_PCLK/4)/* 1=60MHz,2=30MHz,4=15MHz*/
#define DIVISOR (PCLK/(16*DESIRED_BAUDRATE))

void InitUart0(void)
{
/*  U0LCR: UART0 Line Control Register
    0x83: enable Divisor Latch access, set 8-bit word length,
    1 stop bit, no parity, disable break transmission      */
    U0LCR=0x83;

/*  VPBDIV: VPB bus clock divider
    0x01: PCLK = processor clock        */
    VPBDIV=0x00;

/*  U0DLL: UART0 Divisor Latch (LSB)    */
    U0DLL=DIVISOR&0xFF;

/*  U0DLM: UART0 Divisor Latch (MSB)    */
    U0DLM=DIVISOR>>8;

/*  U0LCR: UART0 Line Control Register
    0x03: same as above, but disable Divisor Latch access   */
    U0LCR=0x03 ;

/*  U0FCR: UART0 FIFO Control Register
    0x05: Clear Tx FIFO and enable Rx and Tx FIFOs      */
    U0FCR=0x05 ;
}

void Tx_char(char ch)
{
    if (ch=='\n')
    {
```

Add UART0.c files in the project window.

After writing the code we to add all necessary files and save the code , after saving we have to compile and build the code. For building we can press F7 button or can click on

either of this icons.

After that we can see the results or can find out any errors or warnings in the Build output window as shown in figure below



```
		//then store to Transmit Holding Register
		U0THR='\r';
	}
	//wait until Transmit Holding Register is empty
	while (!(U0LSR&0x20)) {}

	//then store to Transmit Holding Register
	U0THR=ch;



}

unsigned char Rx_char(void)
{
	char ch;

	//wait until there's a character to be read
	while (!(U0LSR&0x01)) {}

	//then read from the Receiver Buffer Register
	ch=U0RBR;
	return ch;
}


int Tx_string(char *s)
{
	int i=0;
	while(s[i]!='\0')
	{
		Tx_char(s[i]);
		i++;
	}
```

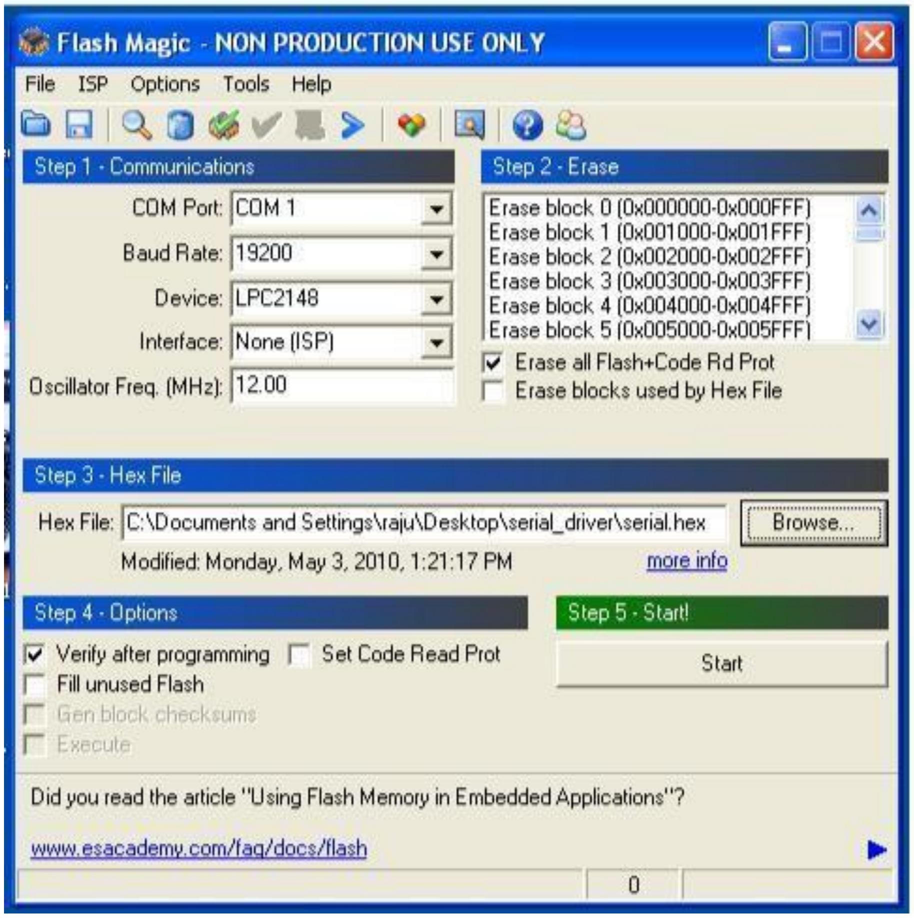
```
compiling UART0.c...
compiling serial.c...
serial.c(2): warning:  #951-D: return type of function "main" must be "int"
serial.c(5): warning:  #223-D: function "InitUart0" declared implicitly
serial.c(6): warning:  #223-D: function "Tx_string" declared implicitly
serial.c(8): warning:  #1-D: last line of file ends without a newline
linking...
Program Size: Code=1668 RO-data=16 RW-data=0 ZI-data=1256
FromELF: creating hex file...
"serial_driver.axf" - 0 Error(s), 4 Warning(s).
```

We can see that there are 0 errors and few warning , we can ignore warning as we can get correct output with warnings also.

When there are no errors then we can download this hex file in ARM using FlashMagic .

To download the **serial.hex** file on to the microcontroller

To download the hex file into the microcontroller board we use a software called **Flash magic tool.**



Step 1 - Communications

COM Port: COM 1
Baud Rate: 19200
Device: LPC2148
Interface: None (ISP)
Oscillator Freq. (MHz): 12.00

Step 2 - Erase

Erase block 0 (0x000000-0x000FFF)
Erase block 1 (0x001000-0x001FFF)
Erase block 2 (0x002000-0x002FFF)
Erase block 3 (0x003000-0x003FFF)
Erase block 4 (0x004000-0x004FFF)
Erase block 5 (0x005000-0x005FFF)

- [x] Erase all Flash+Code Rd Prot
- [ ] Erase blocks used by Hex File

Step 3 - Hex File

Hex File: C:\Documents and Settings\raju\Desktop\serial_driver\serial.hex

Modified: Monday, May 3, 2010, 1:21:17 PM

Step 4 - Options

- [x] Verify after programming
- [ ] Set Code Read Prot
- [ ] Fill unused Flash
- [ ] Gen block checksums
- [ ] Execute

Step 5 - Start!

Did you read the article "Using Flash Memory in Embedded Applications"?

www.esacademy.com/faq/docs/flash

0

Step 1-Communications

Set COM Port	:COM1
Baud Rate	19200
Device	: LPC2148
Interface	:None(ISP)
Oscillator Freq(MHz):12	

Step 2-Erase

Select the box Erase all Flash + Code Rd Prot

Step 3-Hex File

Click on browse to load the serial.hex file from the folder serial_driver.

Step 4-Options

Select the box Verify after programming.

Power up the microcontroller board using USB cable, make serial cable connection between PC and microcontroller's UART0 db9 connector. To make the board enter programming mode

Hold down ISP and Press Reset button SW2, then Keep the SW23 (Blue colour switch) in Release position.

Step 5-Start

CONCLUSION:

Ex.No:2 **INTERFACING ADC AND DAC**

AIM:

Perform Interfacing of ADC and DAC with ARM2148 and evaluate the response of internal and external variations.

THEORY:

Successive approximation ADC is the advanced version of Digital ramp type ADC which is designed to reduce the conversion and to increase speed of operation. The major draw of digital ramp ADC is the counter used to produce the digital output will be reset after every sampling interval. The normal counter starts counting from 0 and increments by one LSB in each count, this result in 2N clock pulses to reach its maximum value.

In successive approximation ADC the normal counter is replaced with successive approximation register as shown in below figure.

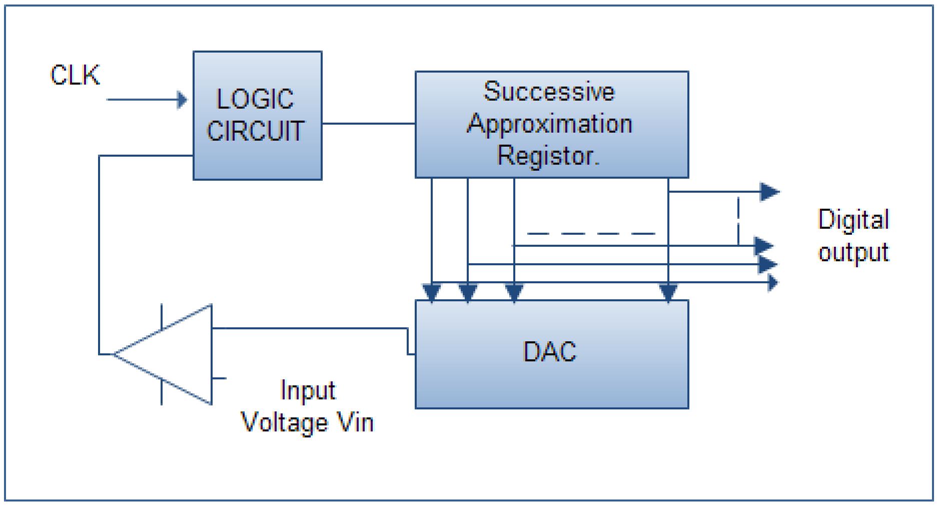


Figure 1. Successive approximation ADC

Figure 2 shows an example of a 4-bit conversion. The y-axis (and the bold line in the figure) represents the DAC output voltage. In the example, the first comparison shows that VIN < VDAC. Thus, bit 3 is set to 0. The DAC is then set to 01002 and the second comparison is performed. As VIN > VDAC, bit 2 remains at 1. The DAC is then set to 01102, and the third comparison is performed. Bit 1 is set to 0, and the DAC is then set to 01012 for the final comparison. Finally, bit 0 remains at 1 because VIN > VDAC.

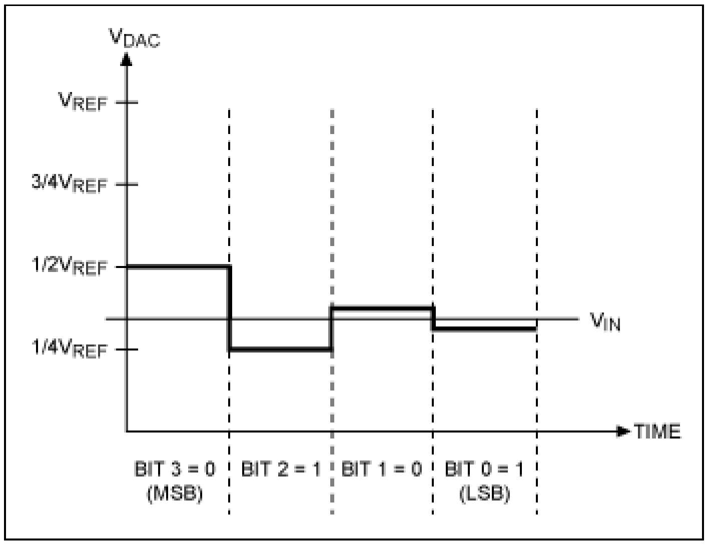


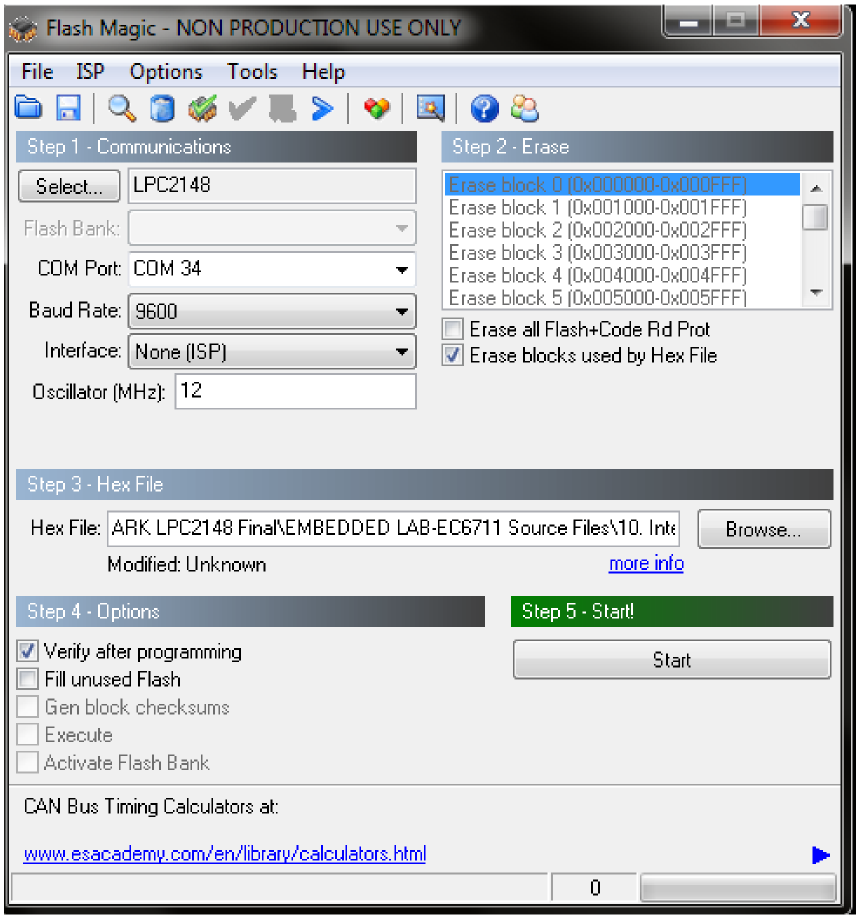
Figure 2. SAR operation (4-bit ADC example).

General setup:

1. Install KEIL SETUP Ver 5.18Trial from the CD.(Its used for develop the program)
2. Install the FT232 driver from the CD. (Driver setup)
3. Install the Flash Magic Setup from the CD. (It’s used for Execute the program)
4. For detailed information see KEIL µVision 4 Tool and Flash Magic.
5. Install Terminal setup\ teraterm-4.89, it’s only used for serial communication.

PROCEDURE:

1. Connect the 9V adaptor to the ARM2148 Board.
2. Connect ARM2148 Board’s Lower USB to PC’s USB by using given USB cable.
3. Keep the Switch SW21 in ON position.
4. Open Flash Magic and follow the steps1, 2, 3to choose Respective .hex program and proceed step 4, 5 to start.
5. The bottom of Flash Magic page indicate that completed, then ensure the Jumper positions of ADC channel I and II,
6. For internal variation, the POT is to be varied from min – max position.
7. The ADC value is shown in the LCD display varies from 0000-1023.


Flash Magic - NON PRODUCTION USE ONLY

File ISP Options Tools Help

Step 1 - Communications

Select... LPC2148

Flash Bank:

COM Port: COM 34

Baud Rate: 9600

Interface: None (ISP)

Oscillator (MHz): 12

Step 2 - Erase

Erase block 0 (0x000000-0x000FFF)
Erase block 1 (0x001000-0x001FFF)
Erase block 2 (0x002000-0x002FFF)
Erase block 3 (0x003000-0x003FFF)
Erase block 4 (0x004000-0x004FFF)
Erase block 5 (0x005000-0x005FFF)

☐ Erase all Flash+Code Rd Prot
☑ Erase blocks used by Hex File

Step 3 - Hex File

Hex File: ARK LPC2148 Final\EMBEDDED LAB-EC6711 Source Files\10. Inte

Modified: Unknown more info

Step 4 - Options

☑ Verify after programming
☐ Fill unused Flash
☐ Gen block checksums
☐ Execute
☐ Activate Flash Bank

Step 5 - Start!

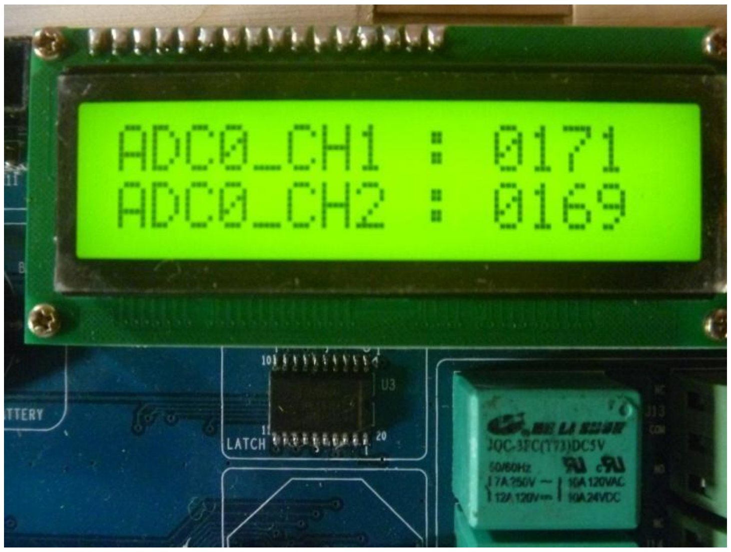
Start

CAN Bus Timing Calculators at:

www.esacademy.com/en/library/calculators.html

0

MODEL OUTPUT:


```
ADC0_CH1 : 0171
ADC0_CH2 : 0169
```

Similarly we have different programs related to ADC and DAC, Here we need to change, only 4th step of procedure to load different files for different Experiments.

For ADC with serial port experiment:

1. Follow the step up to 5 from the procedure.
2. Open the teraterm-4.89 software and close the SW23 (Blue colour).
3. The ADC value is to be displayed in PC, else once press the reset button (SW2).
4. After completion of this experiment open the SW23 (Blue colour)
5. If teraterm-4.89 software is in open condition means, the PC won’t communicate with the Kit, for that we need to close the teraterm-4.89.
6. The DAC value can be measured using multimeter.

CONCLUSION:

Ex.No:3 **INTERFACING LED**

AIM:

Perform Interfacing of LED with ARM2148 and evaluate the response of variations.

THEORY:

LEDs are the most efficient way to turn an electric current into illumination. When a current flows through a diode in the forward direction, it consists of surplus electrons moving in one direction in the lattice and “holes” (voids in the lattice) moving in the other. Occasionally, electrons can recombine with holes. When they do, the process releases energy in the form of photons.

This is true of all semiconductor junctions, but LEDs use materials that maximize the effect. The color of the light emitted (corresponding to the energy of the photon) is determined by the semiconductor materials that form the diode junction.

The latest high-brightness (HB) white LEDs are made possible by the discovery of semiconductor materials that produce blue or ultraviolet photons. In addition to the diode, an HB package contains “yellow” phosphors on the inside of its lens. Some “blue” photons escape, but others excite the phosphors, which then give off “yellow” photons. The result can be tuned in manufacturing to produce “white” light.

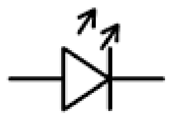


Figure. LED symbol

A great deal of LED engineering relates to controlling the quality of this light. From a circuit standpoint, there are a number of ways to interconnect multiple LEDs to increase and manage light output. The general approach is to drive series strings with a constant current, but there are

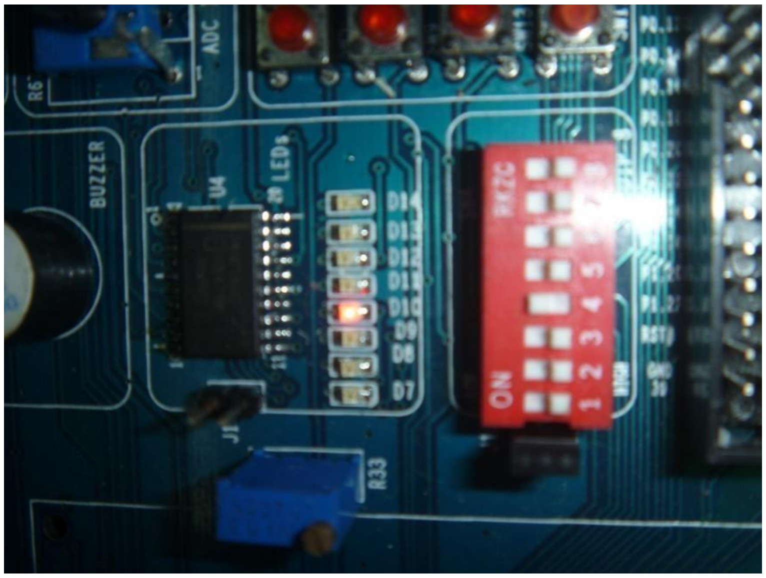
General setup:

1. Install KEIL SETUP Ver 5.18Trial from the CD.(Its used for develop the program)
2. Install the FT232 driver from the CD. (Driver setup)
3. Install the Flash Magic Setup from the CD. (It's used for Execute the program)
4. For detailed information see KEIL µVision 4 Tool and Flash Magic.
5. Install Terminal setup\ teraterm-4.89, it's only used for serial communication.

PROCEDURE:

1. Connect the 9V adaptor to the ARM2148 Board.
2. Keep the Switch SW21 in ON position.
3. Connect ARM2148 Board's Lower USB to PC's USB by using given USB cable.
4. Open Flash Magic and follow the steps1, 2, 3to choose program and step 4, 5 to start.
5. The bottom of Flash Magic page indicate that completed.
6. The LED glow according to corresponding input.

MODEL OUTPUT:



CONCLUSION:

Ex.No:4 **INTERFACING REAL TIME CLOCK**

AIM:

Perform Interfacing of real time clock with ARM2148 and evaluate the response of variations.

THEORY:

Real Time Clock (RTC) is used to store Time and Date in the system even when the system is not in operation. This is the system used in many devices including Laptop. Mobile phones, Tablet, Digital Cameras etc. RTC is inbuilt part of any electronics device. DS1307 is widely used RTC which we can interface with any controller. Many controllers now has inbuilt RTC to store timing information. In this article, we will learn how to configure inbuilt RTC of controller LPC2148.

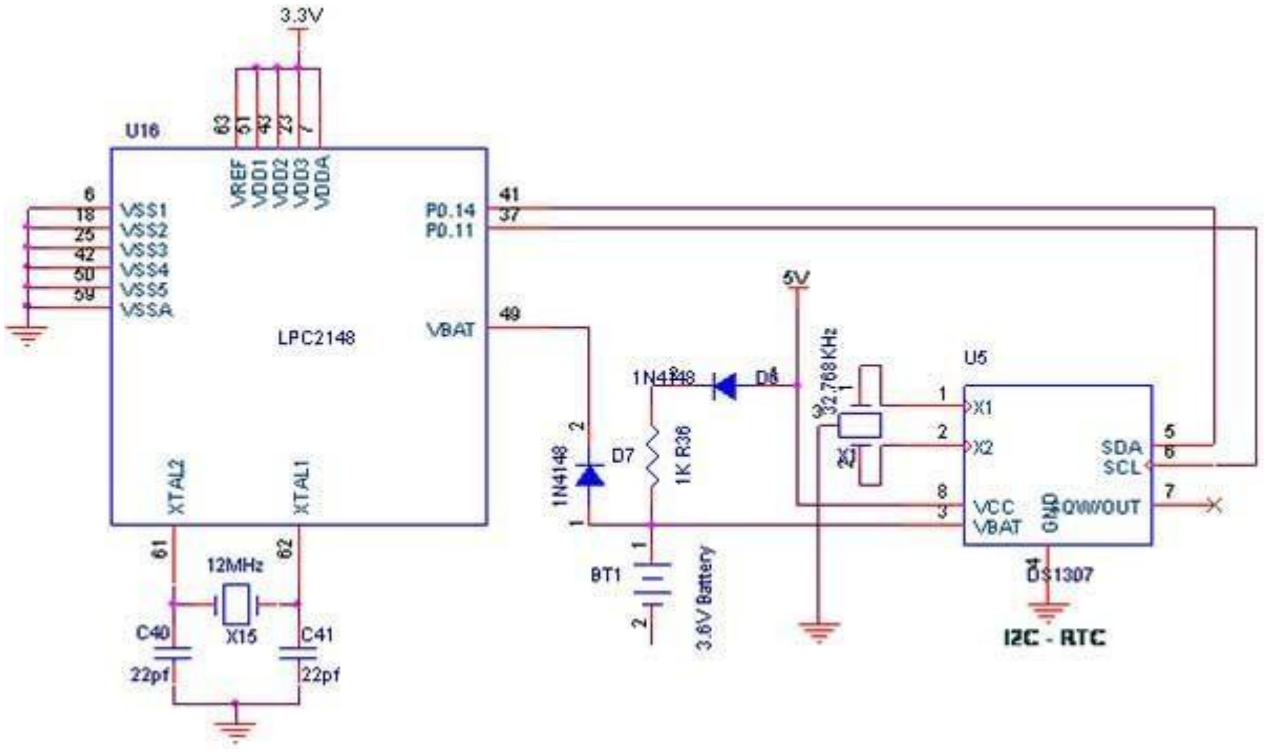


Figure. RTC

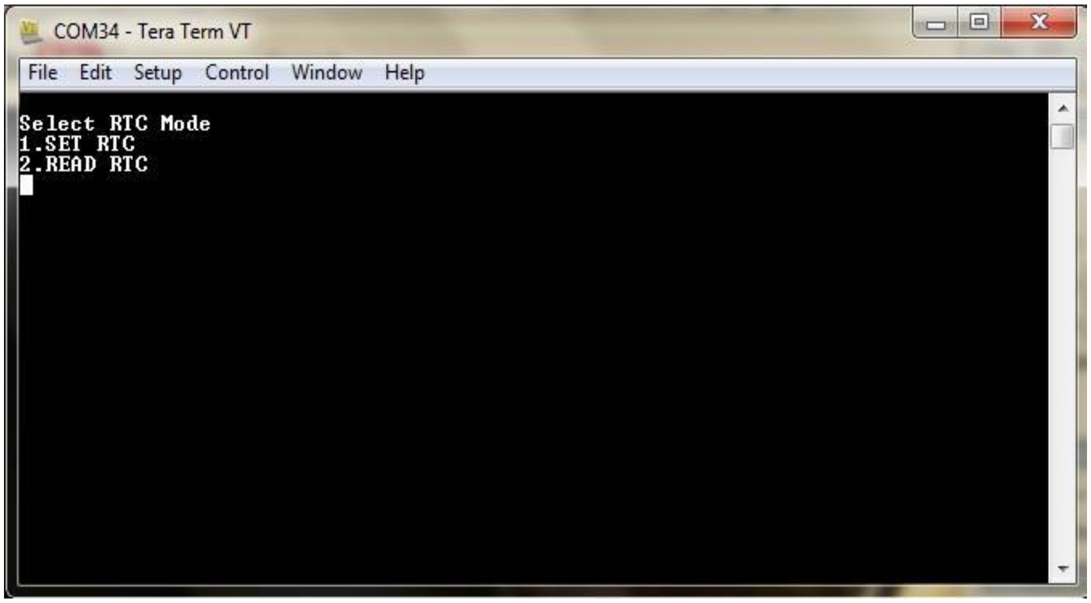
General setup:

1. Install KEIL SETUP Ver 5.18Trial from the CD.(Its used for develop the program)
2. Install the FT232 driver from the CD. (Driver setup)
3. Install the Flash Magic Setup from the CD. (It's used for Execute the program)
4. For detailed information see KEIL µVision 4 Tool and Flash Magic.
5. Install Terminal setup\ teraterm-4.89, it's only used for serial communication.

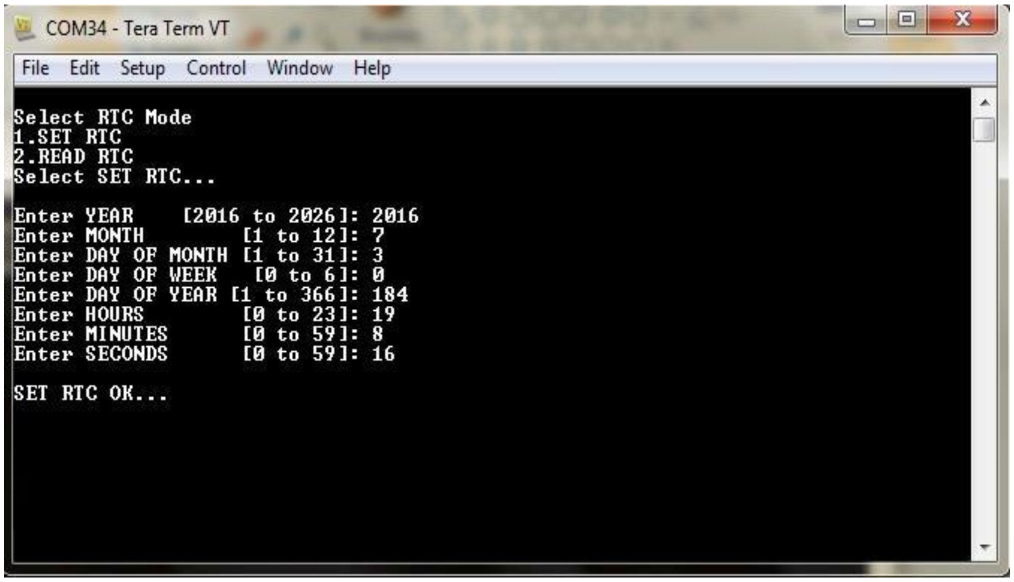
PROCEDURE:

1. Connect the 9V adaptor to the ARM2148 Board.
2. Keep the Switch SW21 in ON position.
3. Connect ARM2148 Board's Lower USB to PC's USB by using given USB cable.
4. Open Flash Magic and follow the steps1, 2, 3to choose Respective .hex program and step 4, 5 to start.
5. The bottom of Flash Magic page indicate that completed,
6. The RTC value is shown in the LCD display.

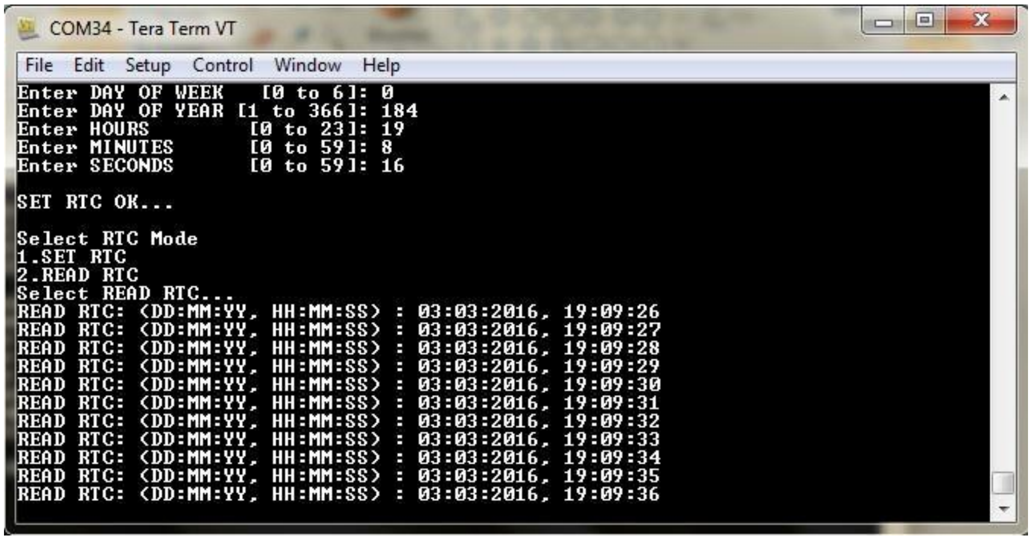
MODEL OUTPUT:



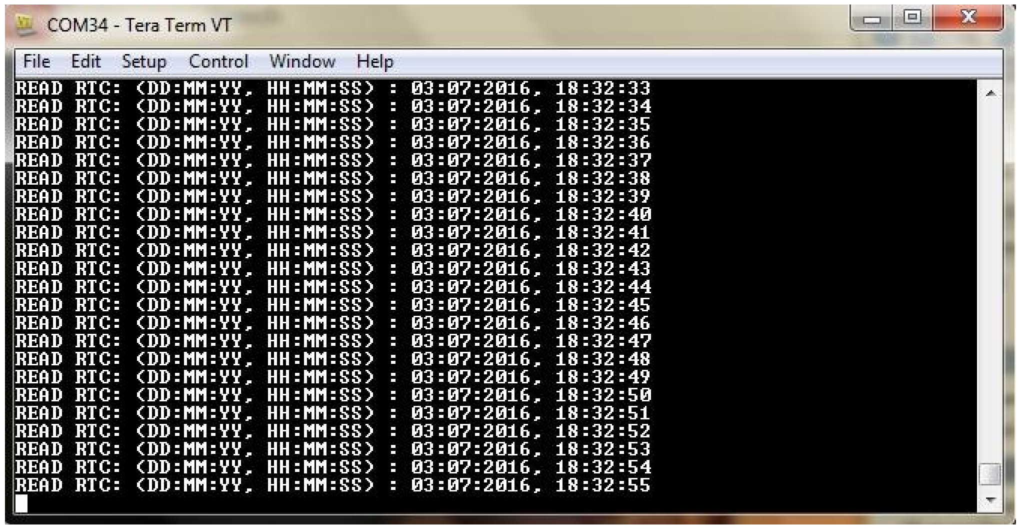
```
Select RTC Mode
1.SET RTC
2.READ RTC
```



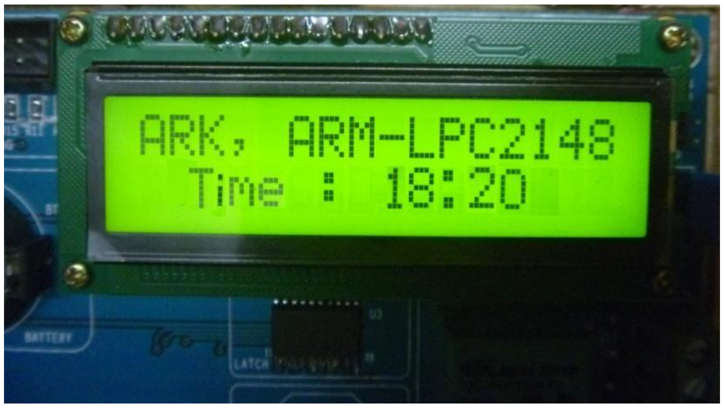
```
Select RTC Mode
1.SET RTC
2.READ RTC
Select SET RTC...

Enter YEAR    [2016 to 2026]: 2016
Enter MONTH         [1 to 12]: 7
Enter DAY OF MONTH [1 to 31]: 3
Enter DAY OF WEEK    [0 to 6]: 0
Enter DAY OF YEAR [1 to 366]: 184
Enter HOURS         [0 to 23]: 19
Enter MINUTES       [0 to 59]: 8
Enter SECONDS       [0 to 59]: 16

SET RTC OK...
```



```
Enter DAY OF WEEK    [0 to 6]: 0
Enter DAY OF YEAR [1 to 366]: 184
Enter HOURS         [0 to 23]: 19
Enter MINUTES       [0 to 59]: 8
Enter SECONDS       [0 to 59]: 16

SET RTC OK...

Select RTC Mode
1.SET RTC
2.READ RTC
Select READ RTC...
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:03:2016, 19:09:26
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:03:2016, 19:09:27
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:03:2016, 19:09:28
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:03:2016, 19:09:29
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:03:2016, 19:09:30
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:03:2016, 19:09:31
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:03:2016, 19:09:32
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:03:2016, 19:09:33
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:03:2016, 19:09:34
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:03:2016, 19:09:35
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:03:2016, 19:09:36
```



```
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:33
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:34
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:35
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:36
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:37
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:38
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:39
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:40
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:41
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:42
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:43
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:44
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:45
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:46
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:47
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:48
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:49
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:50
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:51
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:52
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:53
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:54
READ RTC: (DD:MM:YY, HH:MM:SS) : 03:07:2016, 18:32:55
```



Similarly we have different programs related to RTC, Here we need to change, only 4th step of procedure to load different files for different Experiments.

For RTC with serial port experiment:

1. Follow the step up to 5 from the procedure.
2. Open the teraterm-4.89 software and close the SW23 (Blue colour).
3. The RTC value is to be displayed in PC, else once press the reset button (SW2).
4. After completion of this experiment open the SW23 (Blue colour)
5. If teraterm-4.89 software is in open condition means, the PC won’t communicate with the Kit, for that we need to close the teraterm-4.89.

CONCLUSION:

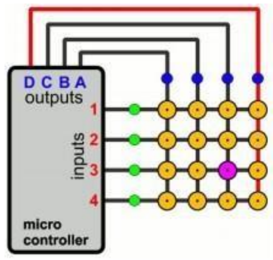
Ex.No:5 **INTERFACING KEYBOARD AND LCD**

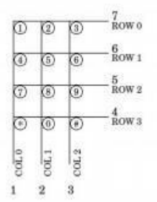
AIM:

Perform Interfacing of interfacing keyboard and LCD with ARM2148 and evaluate the response of variations.

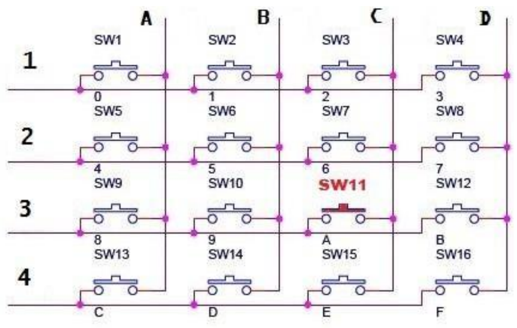
THEORY:

What is keypad(keyboard) ? – What you probably have in front of you, is a keyboard with more than 100 keys on it… If you are not familiar with the key matrices, then you may think that inside this keyboard, there is a chip that has at least the same number of inputs to read each key separately. Well, this is not true…





What are the key matrices?– The matrices are actually an interface technique. According to this technique, the I/O are divided into two sections: the columns and the rows. You can imagine a matrix as an excel sheet. Here is a 4 x 4 matrix. The A,B,C and D are the columns and the 1,2,3 and 4 are the rows. There are 16 knots that the rows and columns intersect. To make a keypad matrix, we will have to connect a switch to each knot. The switch will have a push-to-make contact. When the operator pushes this switch, it will connect the column and the row that it corresponds to.



Keypad-schematic

How does keypad matrix works? – To understand the operation principle, i will re-draw the above matrix without colors as you can see on above pic. I will also put connection switch to each row and column wire and show practical hardware pic. For the above 16-button 4×4 matrix, 8 pins of the microcontroller will be used. The first 4 pins will be OUTPUTS (gives logic '1') and will be connected to the COLUMN wires, while the other 4 pins will be INPUTS (logic '0') and will be connected to the ROW wires. The matrix is controlled by a microcontroller. The OUTPUTS of the microcontroller will NOT all have power at the same time. The outputs will go high one by one in cycle. This happens many times per second. When operator press switch 11 (highlighted in red color), column C and row 3's connection established, current pass from column C to row 3, and in actual sense row 3 read logic '1' from column D and microcontroller manipulate this connection and do work as we decided in programming.

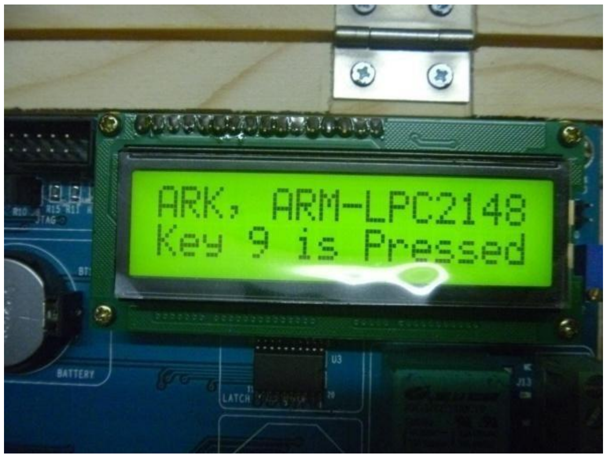
General setup:

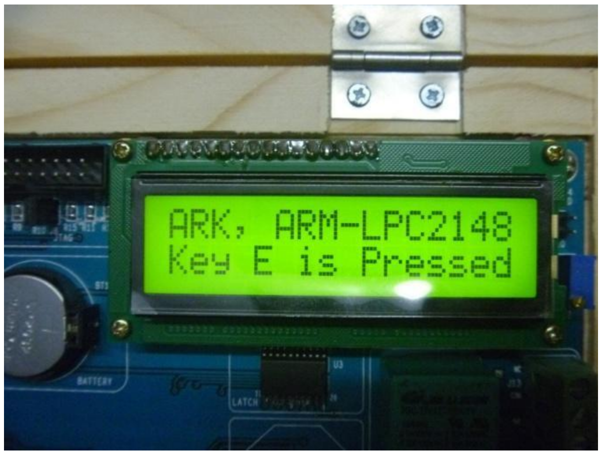
1. Install KEIL SETUP Ver 5.18Trial from the CD.(Its used for develop the program)
2. Install the FT232 driver from the CD. (Driver setup)
3. Install the Flash Magic Setup from the CD. (It's used for Execute the program)
4. For detailed information see KEIL µVision 4 Tool and Flash Magic.
5. Install Terminal setup\ teraterm-4.89, it's only used for serial communication.

PROCEDURE:

1. Connect the 9V adaptor to the ARM2148 Board.
2. Keep the Switch SW21 in ON position.
3. Connect ARM2148 Board's Lower USB to PC's USB by using given USB cable.
4. Open Flash Magic and follow the steps1, 2, 3to choose Respective .hex program and step 4, 5 to start.
5. The bottom of Flash Magic page indicates that completed.
6. By pressing of internal keypad, the position is to be displayed in LCD display varies from 1-F.

MODEL OUTPUT:





Similarly we have different programs related to Keypad, Here we need to change, only 4th step of procedure to load different files for different Experiments.

CONCLUSION:

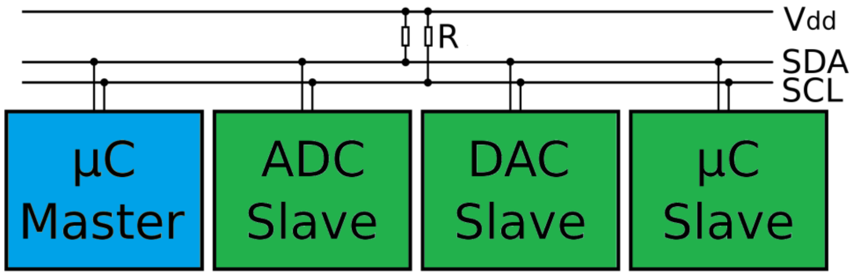
Ex.No:6 **INTERFACING EPROM AND INTERRUPT**

AIM:

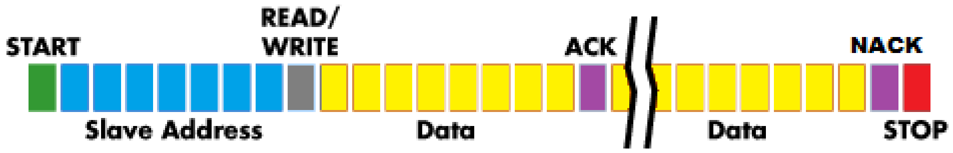
Perform Interfacing of EPROM and interrupt with ARM2148 and evaluate the response of variations.

THEORY:

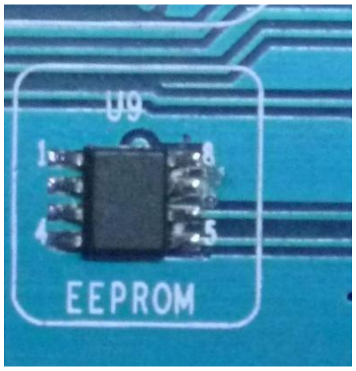
What is I2C? – I²C (Inter-Integrated Circuit)(alternately spelled I2C or IIC)(most commonly pronounced I-squared-C) is a multimaster serial single-ended computer bus invented by the Philips semiconductor division, today NXP Semiconductors, and used for attaching low-speed peripherals to main system.



How I2C protocol work – The I2C protocol requires only 2 signals: clock and data. Clock is known as SCL or SCK (for Serial Clock), while data is known as SDA (for Serial Data). What makes I2C unique is the use of special combinations of signal conditions and changes. Fundamentally, there are just two: Start and Stop. A 'START" condition is generated by the Master, followed by 7 bits of address, then a ReadWrite bit. If a slave device detects an address match, it will send an ACK by driving SDA low during the next clock cycle; if no slave recognizes the address then the SDA line will be left alone to be pulled up high. Following a successful ACK, data will be either sent to the slave device or read from the slave device (depending on what was indicated by the Read/Write bit).



Therefore, each byte is 9 bits: either 7 address plus one R/W plus one ACK/NAK, or 8 data plus one ACK/NAK. The last data byte of a transaction should generally be followed by a NAK, to indicate that it is intended to be the final byte. After this, either a STOP or a ReSTART should be issued by the Master. Bus errors are rarely introduced when using a dedicated I2C peripheral on the Master.



Feature of I2C protocol

- I2C protocol supports multiple data speeds: standard (100 kbps), fast (400 kbps) and high speed (3.4 Mbps) communications.
- Built in collision detection
- 10-bit Addressing
- Supports both Multi-master and Multi-master with Slave functions.
- Data broadcast (general call).

Since only two wires are required, I2C is well suited for boards with many devices connected on the bus. This helps reduce the cost and complexity of the circuit as additional devices are added to the system.

General setup:

1. Install KEIL SETUP Ver 5.18Trial from the CD.(Its used for develop the program)
2. Install the FT232 driver from the CD. (Driver setup)
3. Install the Flash Magic Setup from the CD. (It’s used for Execute the program)
4. For detailed information see KEIL µVision 4 Tool and Flash Magic.
5. Install Terminal setup\ teraterm-4.89, it’s only used for serial communication.

PROCEDURE:

1. Connect the 9V adaptor to the ARM2148 Board.
2. Keep the Switch SW21 in ON position.
3. Connect ARM2148 Board’s Lower USB to PC’s USB by using given USB cable.

4. Open Flash Magic and follow the steps1, 2, 3to choose Respective .hex program and step 4, 5 to start.
5. The bottom of Flash Magic page indicates that completed.
6. EEPROM Perform the following operations,

- EEPROM INIT …...
- EEPROM WRITE
- EEPROM READ
- I2C EEPROM PASS

MODEL OUTPUT:

CONCLUSION:

Ex.No:8 **INTERRUPT PERFORMANCE CHARACTERISTICS OF ARM**

AIM:

Perform Interrupt performance characteristics of ARM experiment with ARM2148 and evaluate the response of variations.

THEORY:

Interrupts for LPC214x MCUs and how to program them for those who are new to interrupts. To start with , first lets see : what **interrupts, IRQs and ISRs** are. As per wiki : "**An interrupt is a signal sent to the CPU which indicates that a system event has a occurred which needs immediate attention**". An '**Interrupt ReQuest**' i.e an '**IRQ**' can be thought of as a special request to the CPU to execute a ***function***(small piece of code) when an interrupt occurs. This ***function*** or 'small piece of code' is technically called an '**Interrupt Service Routine**' or '**ISR**'. So when an IRQ arrives to the CPU , it stops executing the code current code and start executing the ISR. After the ISR execution has finished the CPU gets back to where it had stopped.

Interrupts in LPC214x are handled by **Vectored Interrupt Controller (VIC)** and are classified into 3 types based on the priority levels.(Though Interrupts can classified in many different ways as well)

1. **Fast Interrupt Request i.e FIQ** : which has highest priority
2. **Vectored Interrupt Request i.e Vectored IRQ** : which has 'middle' or priority between FIQ and Non-Vectored IRQ.
3. **Non-Vectored IRQ** : which has the lowest priority.

In computing the term '**Vectored**' means that the CPU is aware of the address of the ISR when the interrupt occurs and **Non-Vectored** means that CPU doesnt know the address of the ISR nor the source of the IRQ when the interrupt occurs and it needs to be supplied by the ISR address. For the Vectored Stuff , the System internally maintains a table called **IVT or Interrupt Vector Table** which contains the information about Interrupts sources and their corresponding ISR address.

If you are still confused than here it is in a nutshell : The difference between Vectored IRQ(VIRQ) and Non-Vectored IRQ(NVIRQ) is that VIRQ has dedicated IRQ service routine for each interrupt source which while NVIRQ has the same IRQ service routine for all Non-Vectored Interrupts. You will get a clear picture when I cover some examples in examples section.

General setup:

6. Install KEIL SETUP Ver 5.18Trial from the CD.(Its used for develop the program)
7. Install the FT232 driver from the CD. (Driver setup)
8. Install the Flash Magic Setup from the CD. (It's used for Execute the program)
9. For detailed information see KEIL µVision 4 Tool and Flash Magic.
10. Install Terminal setup\ teraterm-4.89, it's only used for serial communication.

PROCEDURE:

7. Connect the 9V adaptor to the ARM2148 Board.
8. Keep the Switch SW21 in ON position.
9. Connect ARM2148 Board's Lower USB to PC's USB by using given USB cable.
10. Open Flash Magic and follow the steps1, 2, 3to choose Respective .hex program and step 4, 5 to start.
11. The bottom of Flash Magic page indicates that completed.
12. Press the Interrupt button, (either SW1 or SW3) and CPU operation will be switching for few milli-seconds. Its produce switching sound and BUZZER sound.

CONCLUSION

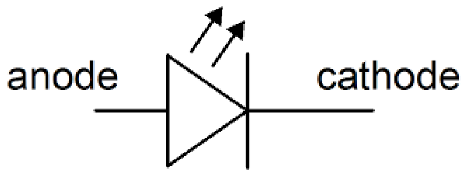
Ex.No:9 **FLASHING OF LEDS**

AIM:

Perform Interfacing of Flashing LED'S experiment with ARM2148 and evaluate the response of variations.

THEORY:

Light Emitting Diodes (LED) is the most commonly used components, usually for displaying pins digital states. Typical uses of LEDs include alarm devices, timers and confirmation of user input such as a mouse click or keystroke. Interfacing LED Fig. 1 shows how to interface the LED to microcontroller. As you can see the Anode is connected through a resistor to GND & the Cathode is connected to the Microcontroller pin. So when the Port Pin is HIGH the LED is OFF & when the Port Pin is LOW the LED is turned ON.



A light emitting diode (LED) is a device which converts electrical energy to light energy. LEDs are preferred light sources for short distance (local area) optical fiber network because they: are inexpensive, robust and have long life (the long life of an LED is primarily due to its being a cold device, i.e. its operating temperature being much lower than that of, say, an incandescent lamp), can be modulated (i.e. switched on and off) at high speeds (this property of an LED is also due to its being a cold device as it does not have to overcome thermal inertia),

I/O Ports

LPC 2148 has two I/O Ports each of 32 bit wide giving us total 64 I/O Pins. Ports are named as P0 and P1. Pins of each port are labeled as PX.Y where X stands for port number, 0 or 1 where else Y stands for pin number 0 to 31. Each pin can perform alternate functions also. For eg. P0.8 serves as GPIO as well as transmitter pin of UART1, PWM4 and AD1.1. On RESET, each pin is configured as GPIO. For any of the other use, programmer must configure it properly.

DEVELOP THE SKILL WITH PROGRAMMING

The first step towards programming is HOW TO CONFIGURE GPIO Pins. Let’s start with the associated concepts and registers.

```
 for(i=0;i<=8;i++)
{
            Turn_On_Led(0x01<<i);
            delay_mSec(500);
}
for(i=0;i<=8;i++)
{
            Turn_On_Led(0x80>>i);
            delay_mSec(500);
}
for(i=0;i<3;i++)
{
            Led_On_All();
            delay_mSec(750);
            Led_Off_All();
            delay_mSec(750);
}
```

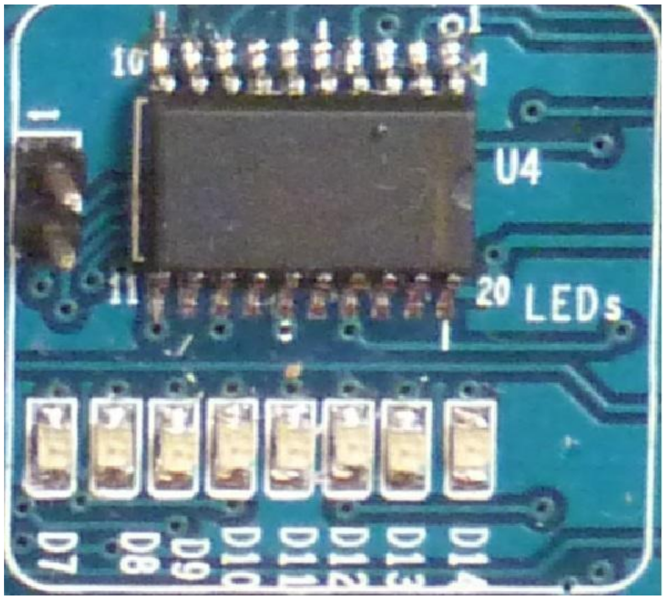
General setup:

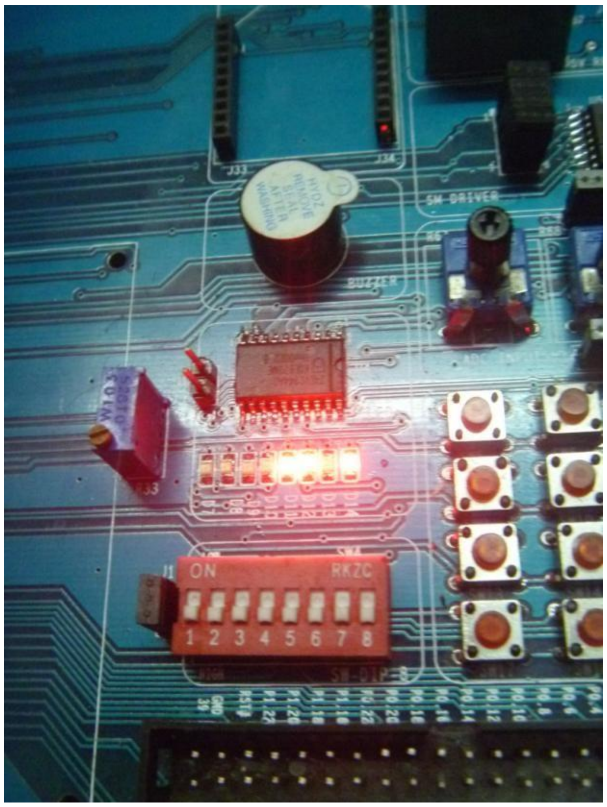
1. Install KEIL SETUP Ver 5.18Trial from the CD.(Its used for develop the program)
2. Install the FT232 driver from the CD. (Driver setup)
3. Install the Flash Magic Setup from the CD. (It’s used for Execute the program)
4. For detailed information see KEIL µVision 4 Tool and Flash Magic.
5. Install Terminal setup\ teraterm-4.89, it’s only used for serial communication.

PROCEDURE:

1. Connect the 9V adaptor to the ARM2148 Board.
2. Keep the Switch SW21 in ON position.
3. Connect ARM2148 Board’s Lower USB to PC’s USB by using given USB cable.
4. Open Flash Magic and follow the steps1, 2, 3to choose Respective .hex program and step 4, 5 to start.
5. The bottom of Flash Magic page indicates that completed.
6. The LED’s starts to flashing in a dancing manner according to the program.

MODEL OUTPUT:





CONCLUSION

Ex.No:10 **INTERFACING STEPPER MOTOR**

AIM:

Perform Interfacing of stepper motor experiment with ARM2148 and evaluate the response of variations.

THEORY:

Stepper motors are electromagnetic incremental devices that convert electric pulses to shaft motion (rotation). These motors rotate a specific number of degrees as a respond to each input electric pulse. Typical types of stepper motors can rotate 2°, 2.5°, 5°, 7.5°, and 15° per input electrical pulse. Rotor position sensors or sensorless feedback based techniques can be used to regulate the output response according to the input reference command. Stepper motors offers many attractive features such as:

• Available resolutions ranging from several steps up to 400 steps (or higher) per / rev.

• Several horsepower ratings.

• Ability to track signals as fast as 1200 pulses per second.

Stepper motors have many industrial applications such as [1]:

• Printers.

• Disk Drives.

• Machine Tools.

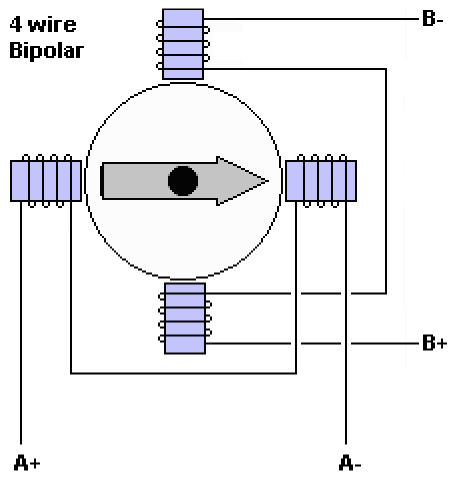
• Robotics.

• Tape Drives.

Types of Stepper Motors

Stepper motors are usually classified into three main categories, namely, Variable reluctance (single stack and multi stack), Permanent Magnet, and Hybrid motors.

2.1 Single Stack Variable Reluctance Stepper Motors



Above Fig. presents the basic circuit configuration of a typical 4-phase, 2-pole, single-stack, variable reluctance stepper motor. The stator is made of a single stack of steel laminations with the phase windings wound around the stator poles. The rotor is made of stack of steel laminations without any windings. The main principle of operation depends on aliging one set only of stator and rotor poles

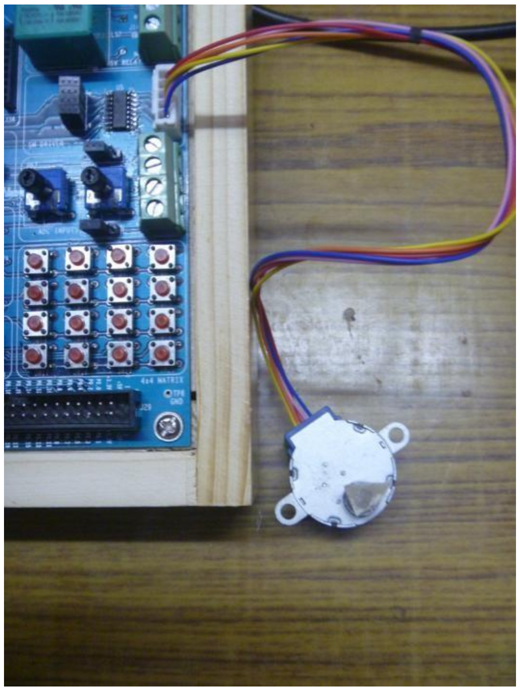
General setup:

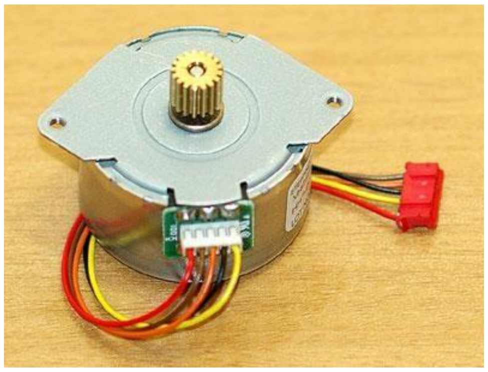
1. Install KEIL SETUP Ver 5.18Trial from the CD.(Its used for develop the program)
2. Install the FT232 driver from the CD. (Driver setup)
3. Install the Flash Magic Setup from the CD. (It's used for Execute the program)
4. For detailed information see KEIL µVision 4 Tool and Flash Magic.
5. Install Terminal setup\ teraterm-4.89, it's only used for serial communication.

PROCEDURE:

1. Connect the 9V adaptor to the ARM2148 Board.
2. Keep the Switch SW21 in ON position.
3. Connect ARM2148 Board's Lower USB to PC's USB by using given USB cable.
4. Open Flash Magic and follow the steps1, 2, 3to choose Respective .hex program and step 4, 5 to start.
5. The bottom of Flash Magic page indicates that completed.
6. Open teraterm-4.89, and keep the switch SW23 in close position to communicate with PC.

MODEL OUTPUT:





CONCLUSION

Ex.No:11 **INTERFACING OF TEMPERATURE SENSOR**

AIM:

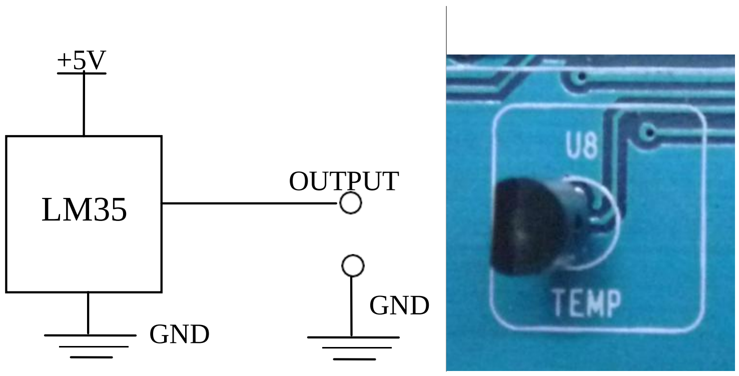
Perform Interfacing of temperature sensor experiment with ARM2148 and evaluate the response of variations.

THEORY:

Temperature is the most-measured process variable in industrial automation. Most commonly, a temperature sensor is used to convert temperature value to an electrical value. Temperature Sensors are the key to read temperatures correctly and to control temperature in industrials applications.

A large distinction can be made between temperature sensor types. Sensors differ a lot in properties such as contact-way, temperature range, calibrating method and sensing element. The temperature sensors contain a sensing element enclosed in housings of plastic or metal. With the help of conditioning circuits, the sensor will reflect the change of environmental temperature.

In the temperature functional module we developed, we use the LM34 series of temperature sensors. The LM34 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Fahrenheit temperature. The LM34 thus has an advantage over linear temperature sensors calibrated in degrees Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Fahrenheit scaling. The LM34 does not require any external calibration or trimming to provide typical accuracies of ±1.2°F at room temperature and ±11.2°F over a full -50 to +300°F temperature range. The LM34 is rated to operate over a -50° to +300°F temperature range.

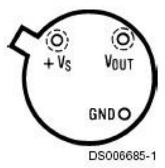


Circuit diagram for the LM35 temperature sensor functional module

It is easy to include the LM35 series in a temperature measuring application. The output voltage of LM35 is linearly proportional to the Fahrenheit temperature, it has a Linear +10.0 mV/°F scale factor which means that you will get n*10.0 mV output voltage if the environment temperature is n°F.

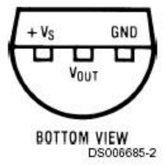
The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM34C, LM34CA and LM34D are also available in the plastic TO-92 transistor package. The LM34D is also available in an 8-lead surface mount small outline package. In our functional module, LM34H in metal can package (TO-46) is used in the functional module, it is very important to know that the wiring of sensor should be based on the positions of the leading pins in different packages.

TO-46
Metal Can Package
(Note 1)



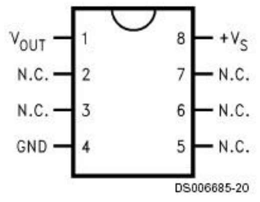
Order Numbers LM34H, LM34AH, LM34CH, LM34CAH or LM34DH
See NS Package Number H03H

TO-92
Plastic Package



Order Number LM34CZ, LM34CAZ or LM34DZ
See NS Package Number Z03A

SO-8
Small Outline
Molded Package



N.C. = No Connection

Top View
Order Number LM34DM
See NS Package Number M08A

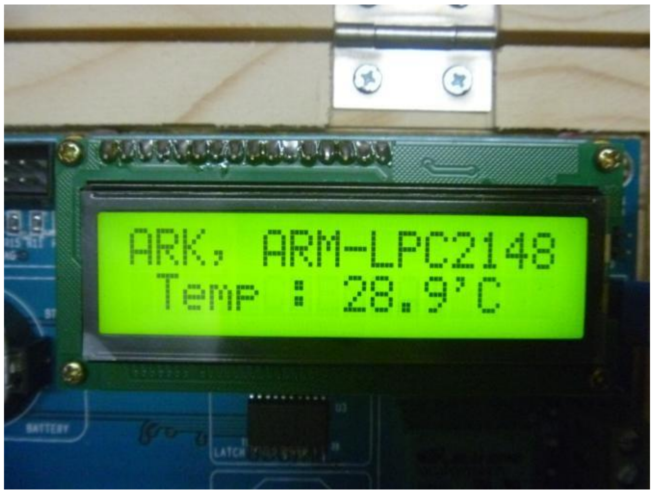
General setup:

1. Install KEIL SETUP Ver 5.18Trial from the CD.(Its used for develop the program)
2. Install the FT232 driver from the CD. (Driver setup)
3. Install the Flash Magic Setup from the CD. (It's used for Execute the program)
4. For detailed information see KEIL µVision 4 Tool and Flash Magic.
5. Install Terminal setup\ teraterm-4.89, it's only used for serial communication.

PROCEDURE:

1. Connect the 9V adaptor to the ARM2148 Board.
2. Keep the Switch SW21 in ON position.
3. Connect ARM2148 Board's Lower USB to PC's USB by using given USB cable.
4. Open Flash Magic and follow the steps1, 2, 3to choose Respective .hex program and step 4, 5 to start.
5. The bottom of Flash Magic page indicates that completed.
6. Vary the external temperature in LM35 and observe the variation in LCD.

MODEL OUTPUT:



CONCLUSION

Ex.No:12 **TEMPERATURE ON/OFF CONTROLLER**

AIM:

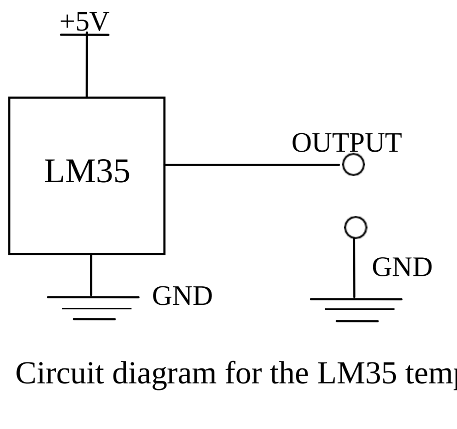
Perform temperature on/off controller experiment with ARM2148 and evaluate the response of variations.

THEORY:

Temperature is the most-measured process variable in industrial automation. Most commonly, a temperature sensor is used to convert temperature value to an electrical value. Temperature Sensors are the key to read temperatures correctly and to control temperature in industrials applications.

A large distinction can be made between temperature sensor types. Sensors differ a lot in properties such as contact-way, temperature range, calibrating method and sensing element. The temperature sensors contain a sensing element enclosed in housings of plastic or metal. With the help of conditioning circuits, the sensor will reflect the change of environmental temperature.

In the temperature functional module we developed, we use the LM34 series of temperature sensors. The LM34 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Fahrenheit temperature. The LM34 thus has an advantage over linear temperature sensors calibrated in degrees Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Fahrenheit scaling. The LM34 does not require any external calibration or trimming to provide typical accuracies of ±1.2°F at room temperature and ±11.2°F over a full -50 to +300°F temperature range. The LM34 is rated to operate over a -50° to +300°F temperature range.

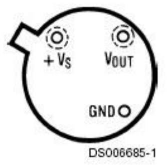


Circuit diagram for the LM35 temperature sensor functional module

It is easy to include the LM35 series in a temperature measuring application. The output voltage of LM35 is linearly proportional to the Fahrenheit temperature, it has a Linear +10.0 mV/°F scale factor which means that you will get n*10.0 mV output voltage if the environment temperature is n°F.

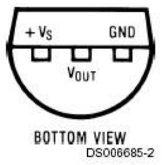
The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM34C, LM34CA and LM34D are also available in the plastic TO-92 transistor package. The LM34D is also available in an 8-lead surface mount small outline package. In our functional module, LM34H in metal can package (TO-46) is used in the functional module, it is very important to know that the wiring of sensor should be based on the positions of the leading pins in different packages.

TO-46
Metal Can Package
(Note 1)



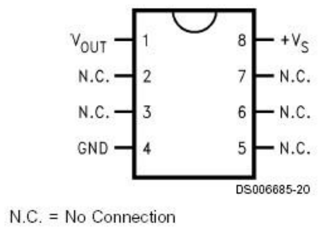
Order Numbers LM34H, LM34AH, LM34CH, LM34CAH or LM34DH
See NS Package Number H03H

TO-92
Plastic Package



Order Number LM34CZ, LM34CAZ or LM34DZ
See NS Package Number Z03A

SO-8
Small Outline
Molded Package



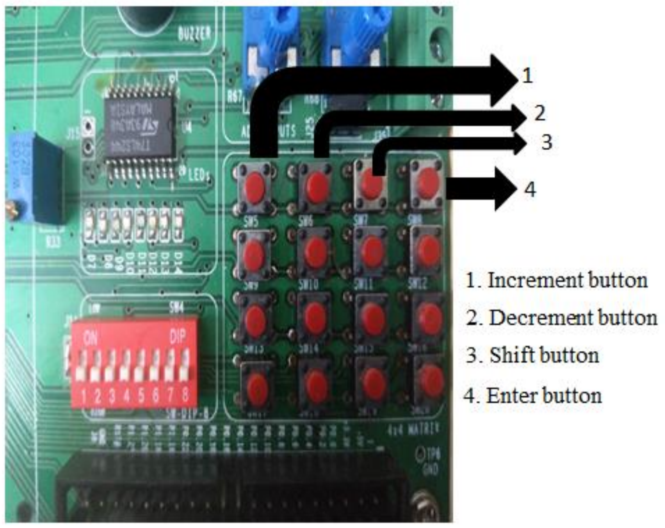
Top View
Order Number LM34DM
See NS Package Number M08A

General setup:

1. Install KEIL SETUP Ver 5.18Trial from the CD.(Its used for develop the program)
2.Install the FT232 driver from the CD. (Driver setup)
3.Install the Flash Magic Setup from the CD. (It’s used for Execute the program)
4.For detailed information see KEIL µVision 4 Tool and Flash Magic.
5.Install Terminal setup\ teraterm-4.89, it’s only used for serial communication.

PROCEDURE:

6. Connect the 9V adaptor to the ARM2148 Board.
7. Keep the Switch SW21 in ON position.
8. Connect ARM2148 Board’s Lower USB to PC’s USB by using given USB cable.
9. Open Flash Magic and follow the steps1, 2, 3to choose Respective .hex program and step 4, 5 to start.
10. The bottom of Flash Magic page indicates that completed.
11. Set values are adjusted by the following instruction 1as increment button, 2 as decrement button, 3 as shift button (from right to left), 4 as enter button.



12. Vary the set degree value in the lpc2148 board and observe the actual degree value in LCD.

CONCLUSION

Ex.No:12 **DC MOTOR CONTROLLER**

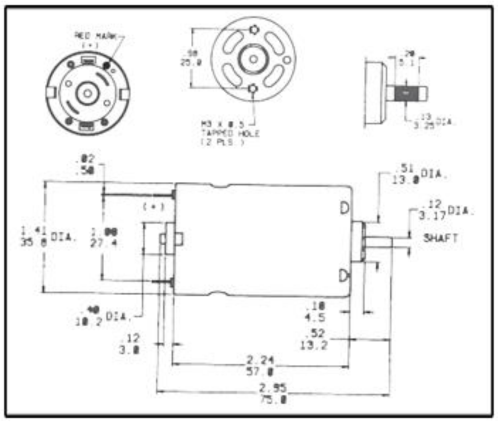
AIM:

Perform DC motor controller experiment with ARM2148 and evaluate the response of variations.

THEORY:

The reason to control the speed of the engine is to conquer the issue in the industry like to maintain a strategic distance from machines harms and to stay away from the moderate ascent time and high overshoot. This is on account of when the beginning voltage is high, it isn‟t reasonable for a machine as it can make machine harms. In this way, a controller like PID is created to beat this issue.

Three techniques have been used to control the speed of the DC motor, one involving a PID controller, other involving PWM hysteresis and hardware implementation using optocoupler sensor. We are going to discuss the theory related to PID controller, PWM hysteresis and optocoupler sensor. A proportional integral subsidiary controller (PID controller) is generally utilized as a part of mechanical control frameworks. It is a bland control circle criticism instrument and utilized as input controller. PID working standard is that it figures blunder esteem from the handled estimated esteem and the coveted reference point. Crafted by the controller is to limit the blunder by changing in the contributions of the framework. In the event that the framework isn‟t plainly known at that point applying PID controller give the best comes about in the event that it is tuned legitimately by keeping parameters of the framework as per the idea of the framework



General setup:

1.Install KEIL SETUP Ver 5.18Trial from the CD.(Its used for develop the program)

2.Install the FT232 driver from the CD. (Driver setup)

3.Install the Flash Magic Setup from the CD. (It's used for Execute the program)

4.For detailed information see KEIL µVision 4 Tool and Flash Magic.

5.Install Terminal setup\ teraterm-4.89, it's only used for serial communication.

PROCEDURE:

6.Connect the 9V adaptor to the ARM2148 Board.

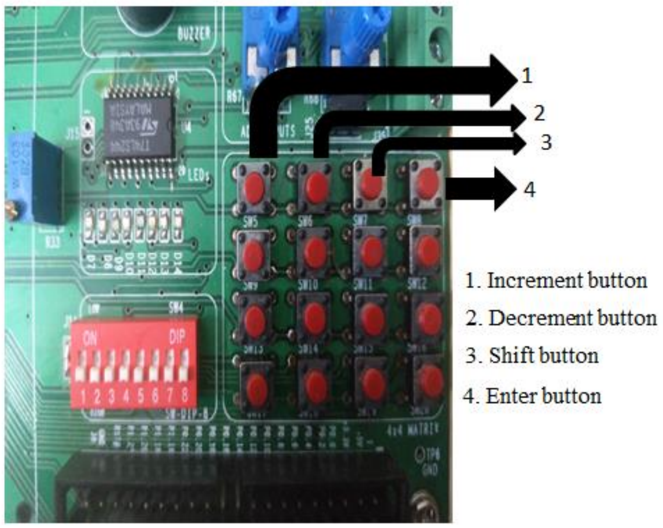
7.Keep the Switch SW21 in ON position.

8.Connect ARM2148 Board's Lower USB to PC's USB by using given USB cable.

9.Open Flash Magic and follow the steps1, 2, 3to choose Respective .hex program and step 4, 5 to start.

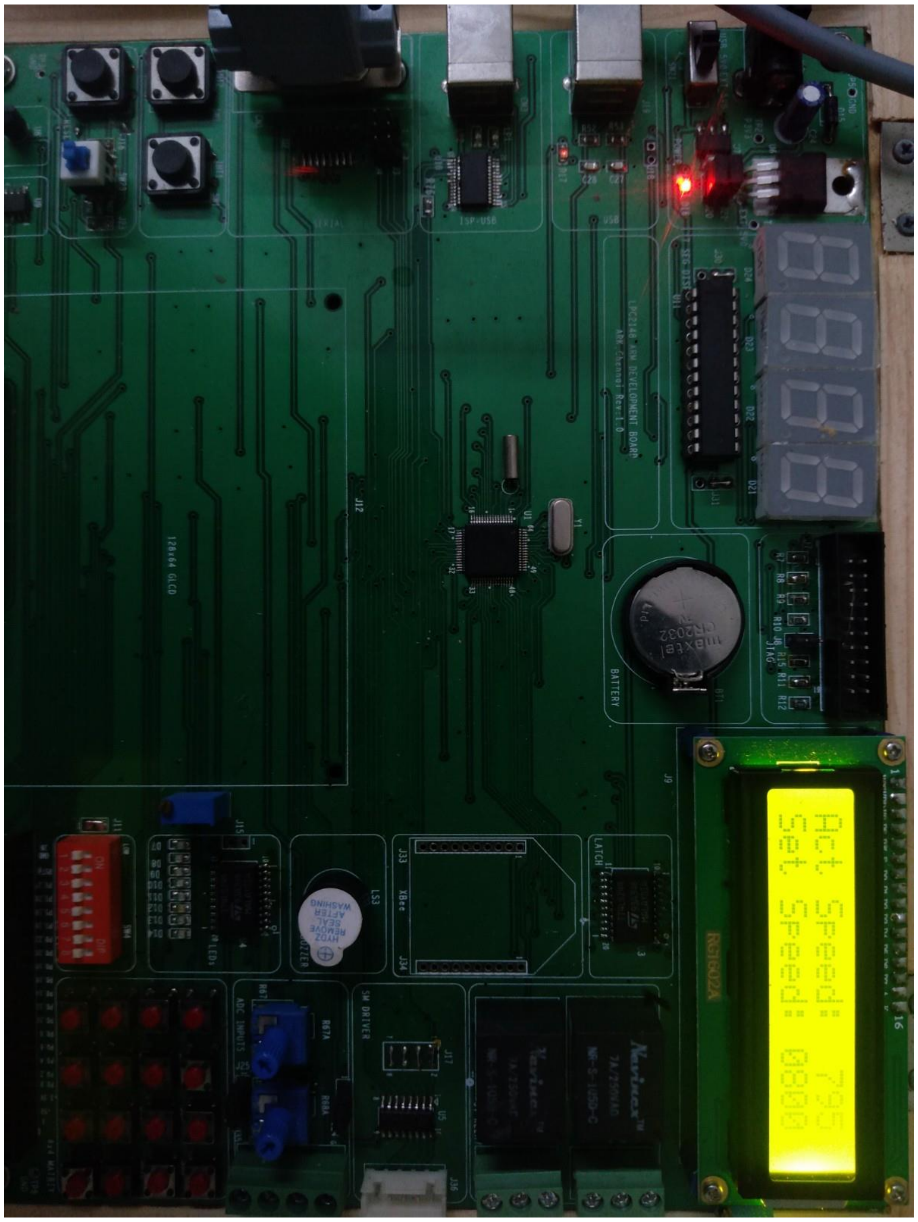
10. The bottom of Flash Magic page indicates that completed.

11. Set values are adjusted by the following instruction 1as increment button, 2 as decrement button, 3 as shift button (from right to left), 4 as enter button.



12. Vary the set value in the lpc2148 board and observe the actual value in LCD.

MODEL OUTPUT:



CONCLUSION