Lecture notes for ARM Architecture

Subject Code : BEEP185T20

Subject : Embedded Systems

Module : 1

Prepared by : S.RAJA, Assistant Professor, Department of EEE

Table of Contents **Page No.**

ARM architecture

1. Aim and Objective:

To study about the ARM architecture.

2. Prerequisites:

Digital Electronics

Microprocessor and Micro controller

3.Pre Test- MCQ type

1. Which is the component of the processor

   A . Arithmetic Logic Unit

   B. Set of the Register

   C. Control Unit

   D. All of the above mentioned options

2.Which one is not the control signal

   A.Read

   B.Write

   C.Reset

   D. None of the above mentioned options

3.Which is used the stack in processor

   A.FIFO

   B.LIFO

   C.LILO

   D.FILO

4. What is the order of the processor to execute the instruction?

   A.Decode,Fetch,Execute

   B.Fetch,Decode,Execute

   C.Execute,Decode, Fetch

   D.Decode,Execute,Fetch

5. EEPROM stands for

A. Erasable Edit Programmable Read Only memory

B. Erasable Edge Programmable Read Only memory

C. Electrically Erasable Programmable Read Only memory

D. Electrically Edit Programmable Read Only memory

4. The ARM7TDMI

The ARM7TDMI is a member of the Advanced RISC Machines (ARM) family of general purpose 32-bit microprocessors, which offer high performance for very low power consumption and price.

Advanced RISC machine (ARM) is the first reduced instruction set computer (RISC) processor for commercial use, which is currently being developed by ARM Holdings

- **Advanced RISC machine (ARM) is the reduced instruction set computer (RISC) processor, which is created by ARM Holdings**
- **90% of the mobile markets ruled by the ARM architecture technology**

Applications

Embedded Solutions

• Smart Cards • Automotive • Dashboard Controls

Enterprise Solutions

• Flash Cards • Hard Disks • Printers

Home Solutions

• Still Cameras • Digital TV • Set Top Box

Mobile Solutions

• Bluetooth • PDA • Smart Phone

Emerging Applications

• Gaming • Robot • And much more….

ARM 7 ARCHITECTURE Features

- 32-bit RISC-processor core (32-bit instructions)
- 37 pieces of 32-bit integer registers (16 available)
- Thumb instruction set
- Pipelined (ARM7: 3 stages)
- Cached (depending on the implementation)
- Von Neuman-type bus structure (ARM7), Harvard (ARM9)
- Debug Interface
- Embedded ICE macrocell
- Jazelle DBX(Direct Bytecode eXecution)
- 8 / 16 / 32 -bit data types
- 7 modes of operation (usr, fiq, irq, svc, abt, sys, und)

32-bit RISC-processor core

RISC Means Reduced Instruction Set Computer

THUMB set

- The THUMB set's 16-bit instruction length allows it to approach twice the density of standard ARM code while retaining most of the ARM's performance advantage over traditional 16-bit processor using 16-bit registers.

This is possible because THUMB code operates on the same 32-bit register set as ARM code

- The key idea behind THUMB is that of a super-reduced instruction set.
- EssentiallyARM7TDMI processor has two instruction sets:
    - the standard 32-bit ARM set
    - a 16-bit THUMB set

pipelining

- It is used speed up the no. of execution per unit time for processor by instruction parallelism
- The process of fetching the next instruction when the present instruction is being executed is called as **pipelining**
- **Pipelining** is a technique that implements a form of parallelism called instruction-level parallelism within a single **processor**.
- It therefore allows faster CPU throughput (the number of instructions that can be executed in a unit of time) than would otherwise be possible at a given clock rate.

Instruction execution without pipeline

The execution of instruction performed one by one, after complete the execution of an instruction the next instruction is fetched from the memory

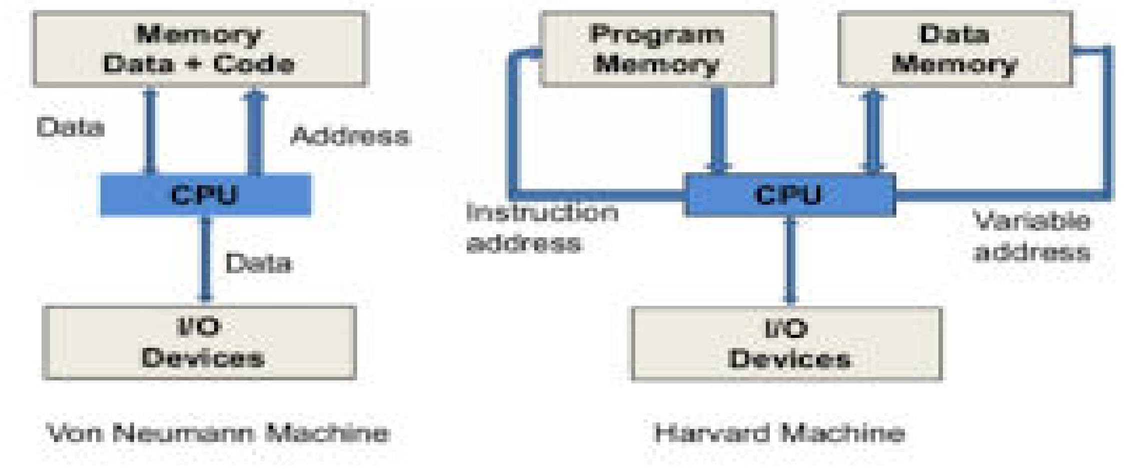
Suppose, these instructions are executed sequentially, and it takes the 1-unit clock cycle for each step to run. So, it would take 12-clock cycles(3 X 4) to execute these instructions.

Cache

- **Cache** is a small amount of memory which is a part of the CPU - closer to the CPU than RAM . It is **used** to temporarily hold instructions and data that the CPU is likely to reuse.

Harvard architecture vs Von-Neumann architecture

- In **Harvard architecture**, the CPU is connected with both the data memory (RAM) **and** program memory (ROM), separately.
- In **Von-Neumann architecture**, there is no separate data **and** program memory. Instead, a single memory connection is given to the CPU.



Debug Interface

- On chip unit for testing called as JTAG interface
- JTAG stands for Joint Test Action Group and defines the set of standards for testing the functionality of hardware.
- There is a set of scan cells located at the boundaries

Embedded ICE macrocell

- Macro cell ICE(In Circuit Emulator) is used to enable the testing .
- This unit is powered by breakpoint and watch point register and control & status register
- All the register work together to halt the ARM core to read status and thus do active debugging

Jazelle DBX(Direct Bytecode eXecution)

- ARM processors to execute Java byte code in hardware as a third execution state along with existing ARM and Thumb mode
- Useful to increase the execution speed of Java ME games and applications

Operating Modes

- ARM has seven operating modes:
  - User : Unprivileged mode under which most tasks run
  - FIQ(Fast Interrupt Request): Entered on a high priority interrupt request
  - IRQ(Interrupt Request): Entered on a low priority request
  - Supervisor: Entered on reset and when a software interrupt instruction is executed.
  - Abort: used to handle memory access violations
  - Undef: used to handle undefined instructions
  - System: privileged mode using the same register as user mode

4.3.Memory Interface

- The ARM processor has Von Neumann Architecture, with 32 bit data bus carrying both instruction and data
- Only the load, store, swap instructions can access from the memory

ARM Processor access the memory by four different types

1. Non sequential Bus Cycle

2. Sequential Bus Cycle

3. Internal Cycle

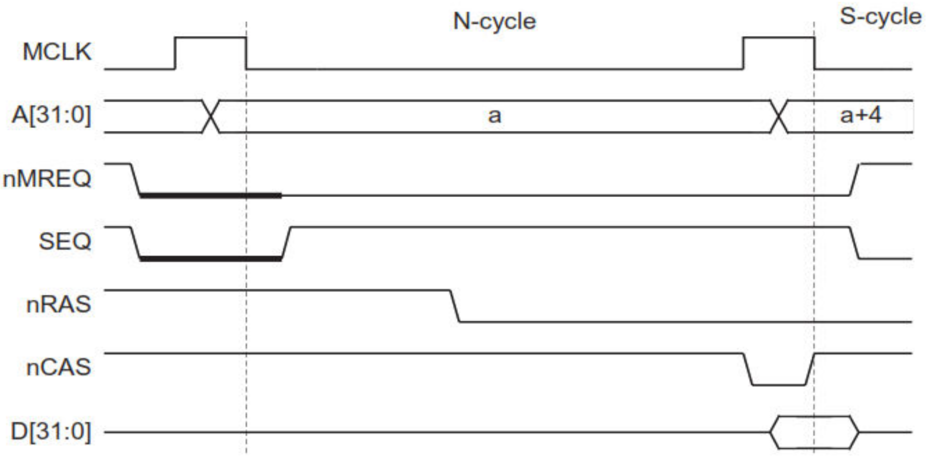
4.Co-processor Cycle

Processor determines the type transfer cycle takes place based on the two signals nMREQ and SEQ

nMREQ	SEQ	BUS Cycle	Description
0	0	N Cycle	Non sequential Bus Cycle
0	1	S- Cycle	Sequential Bus Cycle
1	0	I-Cycle	Internal Cycle
1	1	C-Cycle	Co-processor Cycle

1. Non sequential Cycle

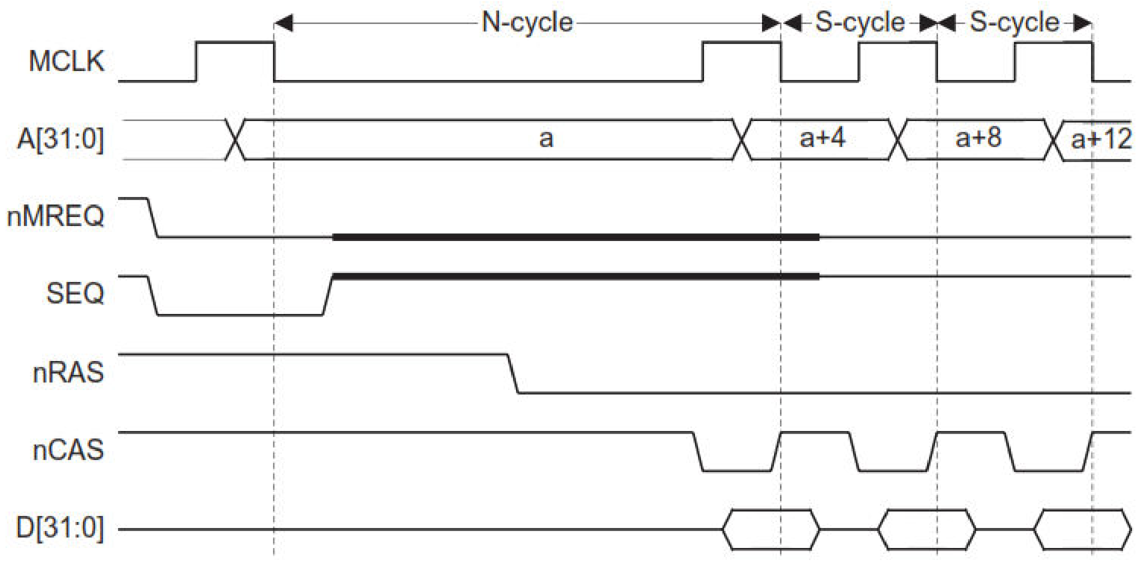
It is a type of memory access cycle in which access the memory through memory location is unrelated to the preceding cycle transfer.



Sequential Cycle

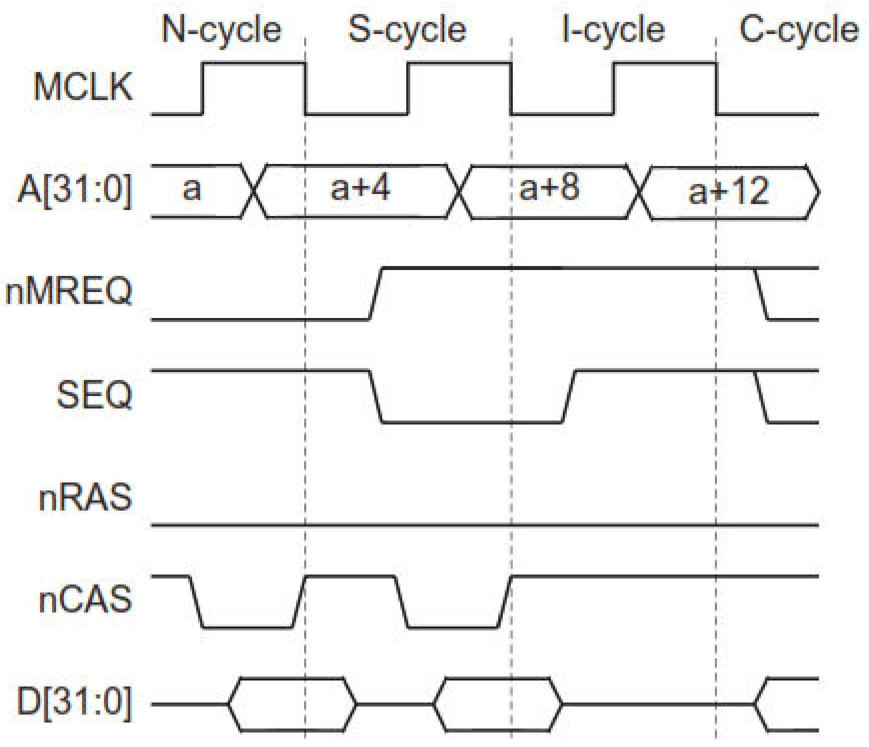
It is a type of memory access cycle in which access the memory through memory location is related to the preceding cycle transfer.

It is used generally program access from the memory and used for burst data transfer



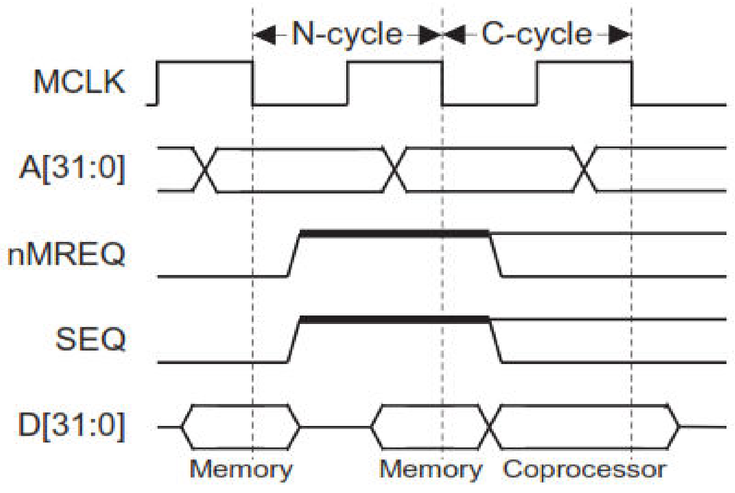
Internal Cycle

In this bus cycle no memory access performed only an internal function to be executed



Co-Processor register transfer Cycle

During this cycle. ARM7TDMI processor access the data from or to the Co-processor .memory access does not initiate the during the cycle.

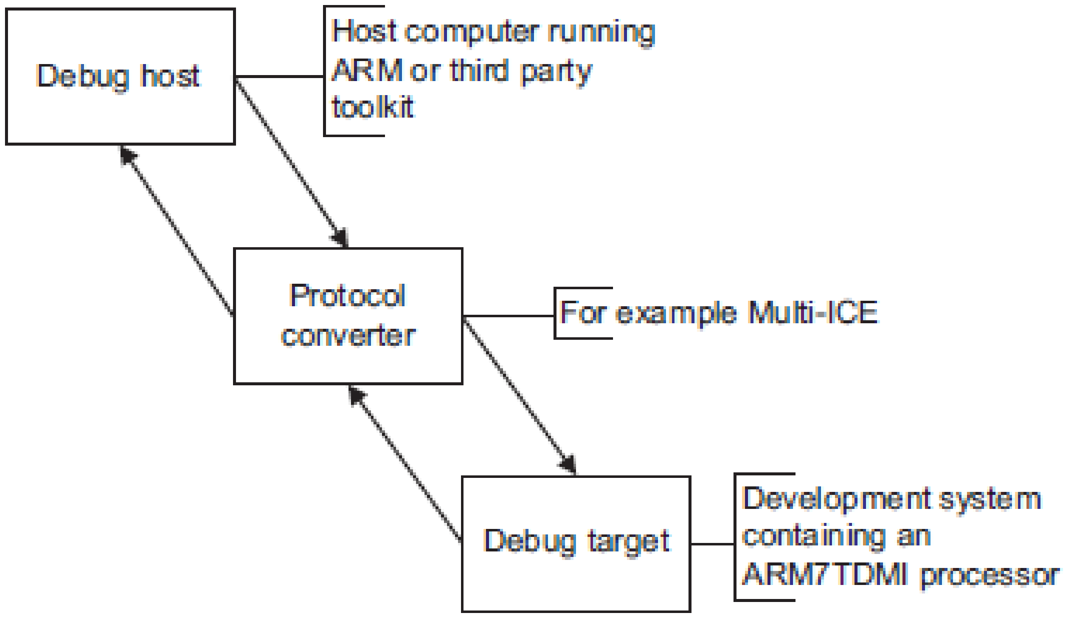


4.4.Debug Interface

- The debug interface is based on IEEE Std. 1149.1-1990, Standard Test Access Port and Boundary Scan Architecture.
- The ARM7TDMI processor contains hardware extensions for advanced debugging features
- These make is easier to develop application software, operating systems and hardware itself.

Stages of the Debug

- A request on one of the external debug signal, or on an internal functional unit known as EmbeddedICE Logic forces into debug state
- A break point, an instruction fetch
- A watch point, a data access
- An external debug request


Debug host
Host computer running ARM or third party toolkit
Protocol converter
For example Multi-ICE
Debug target
Development system containing an ARM7TDMI processor

Debug Host

- The Debug host is computer that is running a software debugger such as the Arm Debugger for Windows.

- The debug host allows to issue high level commands such as setting breakpoints of examining the contents of memory

Protocol Converter

- The protocol converter communicates with high commands issued by the debug host and the low level commands of JTAG interface.
- It interfaces to the host through an interface such as an enhanced parallel port.

Debug Target

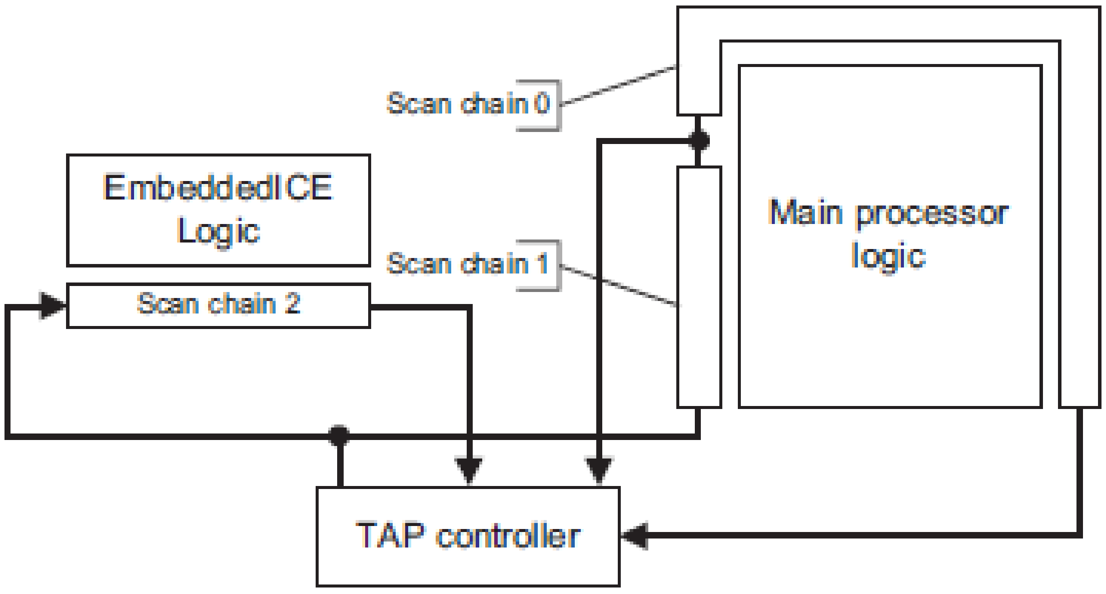
Debug target consists of EmbeddedICE Logic and TAP Controller

- The EmbeddedICE Logic

  This is a set of registers and comparators used to generate debug exceptions such as breakpoints

- The TAP Controller

  This controls the action of the scan chains using a JTAG serial interface



5.Post test-Multiple Choice Questions

The ARM core uses ____ Architecture.

a) **RISC** b) CISC c) Both D) none

2. ARM Processor specifically designed for to reduce ___

a) Size b) Power Consumption **C) both** d) none.

3.ARM Processor core is a key component of ___ bit embedded system.

a)8 b) 16 **c)32** d)64

4.ARM means ___

a) **Advance Risc Machine** b) Advance Review machine c) Advance Risc mechanism d) All

5..RISC means ____

a)**Reduced Instruction set computer** b) Reduced Instruct set computer C) both d) None

6.Conclusion

Hence the in this module we studied about the ARM Architecture in detail

7.References

1. ARM System Developer"s Guide – Andrew N.Sloss.

2. ARM Architecture Reference Manual - David Seal.

3. ARM System-on-Chip Architecture (2nd Edition) by Steve Furbe.

8. Assignment

1. Describe about the features of ARM7TDMI architecture

2. Write short notes about the Memory interface bus signal

3. Mention importance of the Debug interface bus signal

Lecture notes for LPC 2148 ARM Microcontroller and its programming

Subject Code	**: BEEP185T20**
Subject	**: Embedded Systems**
Module	**:2**
Prepared by	**: S.RAJA, Assistant Professor, Department of EEE**

Table of Contents **Page No.**

LPC 2148 Microcontroller Block diagram and its programming

1. Aim and Objective:

To study about the LPC 2148 Micro controller chip.

2. Prerequisites:

Digital Electronics

Microprocessor and Micro controller

3. Pre Test- MCQ type

1. Which is the component of the processor

   A . Arithmetic Logic Unit

   B. Set of the Register

   C. Control Unit

   D. All of the above mentioned options

2.Which one is not the control signal

   A.Read

   B.Write

   C.Reset

   D. None of the above mentioned options

3.Which is used the stack in processor

   A.FIFO

   B.LIFO

   C.LILO

   D.FI LO

4. What is the order of the processor to execute the instruction

   A.Decode,Fetch,Execute

   B.Fetch,Decode,Execute

   C.Execute,Decode, Fetch

   D.Decode,Execute,Fetch

5. EEPROM stands for

A. Erasable Edit Programmable Read Only memory

B. Erasable Edge Programmable Read Only memory

C. Electrically Erasable Programmable Read Only memory

D. Electrically Edit Programmable Read Only memory

4 .LPC2148 microcontrollers

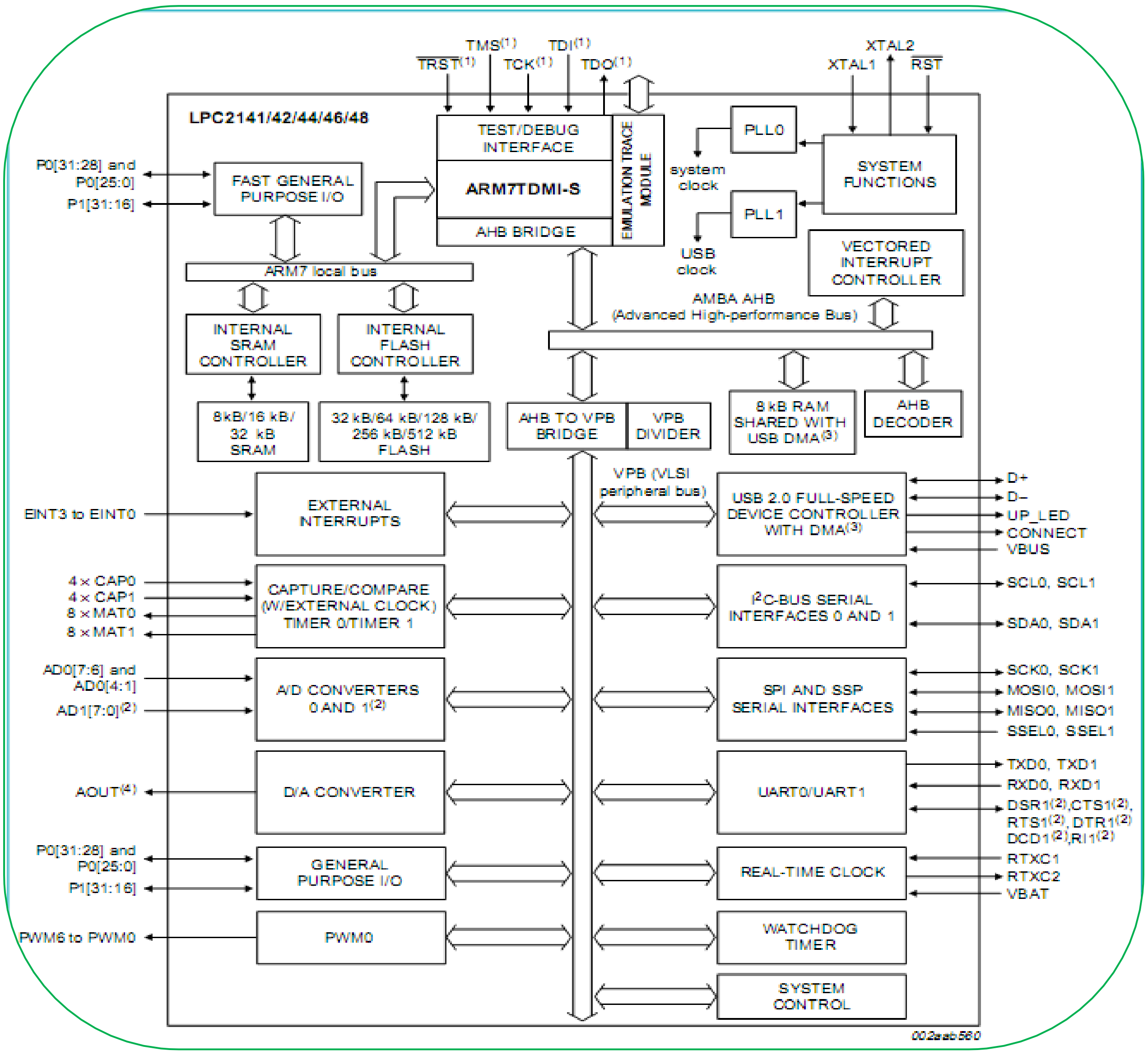
The LPC2148 microcontrollers are based on a 32 bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combines the microcontroller with embedded high speed flash memory of 512 KB

4.1. LPC 2148 Microcontroller Features

- 32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package.
- 32 kB of on-chip static RAM and 512 kB of on-chip flash program memory.
- 128 bit wide interface/accelerator enables high speed 60 MHz operation.
- In-System/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.
- EmbeddedICE RT and Embedded Trace interfaces offer real-time debugging with the on-chip RealMonitor software and high speed tracing of instruction execution.
- USB 2.0 Full Speed compliant Device Controller with 2 kB of endpoint RAM.
- In addition, the LPC2146/8 provide 8 kB of on-chip RAM accessible to USB by DMA.
- One or two (LPC2141/2 vs. LPC2144/6/8) 10-bit A/D converters provide a total of 6/14analog inputs, with conversion times as low as 2.44 ms per channel.
- Single 10-bit D/A converter provides variable analog output.
- Two 32-bit timers/external event counters (with four capture and four comparechannels each), PWM unit (six outputs) and watchdog.
- Low power real-time clock with independent power and dedicated 32 kHz clock input.
- Multiple serial interfaces including two UARTs (16C550), two Fast I2C-bus (400 kbit/s), SPI and SSP with buffering and variable data length capabilities.
- Vectored interrupt controller 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

4.2.LPC 2148 Micro Controller Block diagram

- The ARM7TDMI-S is a general purpose 32-bit microprocessor, which offers high performance and very low power consumption



4.3.System control Functions

- **Crystal Oscillator**
- **External Interrupt Inputs**
- **Miscellaneous System Controls and Status**
- **Memory Mapping Control**
- **PLL**
- **Power Control**

- **Reset**
- **APB Divider**
- **Wakeup Timer**

4.3.1Crystal Oscillator

LPC 2148 Micro controller operated at frequency 12-60MHZ. The clock frequency is generated using crystal oscillator which is connected to pin of XTAL1 and XTAL2

4.3.2.External Interrupt Inputs

It has 5 external interrupt. Which is

EINT0 – general purpose interrupt input is used wake up the processor from idle mode

EINT1 -general purpose interrupt input is used start the ISP command handler

EINT2 - general purpose interrupt input

EINT3- general purpose interrupt input

RESET- external reset input when this input is low chip resets

4.3.4.Memory Map

LPC 2148 microcontroller has 4GB of total memory space which is 32 bit internal bus address (2^{32}= 4GB)

It is a memory mapped Input/Output system in which RAM , ROM memory and peripherals share the same memory space

4.3.5.Phase Locked Loop(PLL)

- There are two PLL modules in the LPC2148 microcontroller.
- The PLL0 is used to generate the CCLK clock (system clock)
- while the PLL1 has to supply the clock for the USB at the fixed rate of 48 MHz.
- The PLL0 and PLL1 accept an input clock frequency in the range of 10 MHz to 25 MHzonly.
- The input frequency is multiplied up the range of 10 MHz to 60 MHz for the CCLK and 48 MHz for the USB clock using a Current Controlled Oscillators (CCO).
- The multiplier can be an integer value from 1 to 32 .

- CCO operates in the range of 156 MHz to 320 MHz, so there is an additional divider in the
- loop to keep the CCO within its frequency range while the PLL is providing the desired output frequency.
- The output divider may be set to divide by 2, 4, 8, or 16 to produce the output clock. Since the minimum output divider value is 2, it is insured that the PLL output has a 50% duty cycle.

4.3.6.Power Control

One of the important features of ARM micro controller is low power consumption in order to achieve the power optimization two modes are available

1. **Power down mode**
2. **Idle mode**

Power down mode

In this mode the crystal oscillator shutdown and chip does not receives any clocks. During this mode no operation takes place which can resume back only by either certain specific interrupt or reset that triggered.

Due to above all, the dynamic power consumption of chip is negligible hence power consumption almost zero

Idle mode

During this mode, instruction execution is suspected. But peripherals can continue operation and may be generated the interrupt to resume the chip to execute the instructions

Power consumption of the processor, bus, control logic and memory is eliminated.

4.3.7.APB divider

- The APB Divider determines the relationship between the processor clock (CCLK) and the clock used by peripheral devices (PCLK).
- The APB Divider serves two purposes.
- The first is to provides peripherals with desired PCLK via APB bus so that they can operate at the speed chosen for the ARM processor.

4.4. Timer

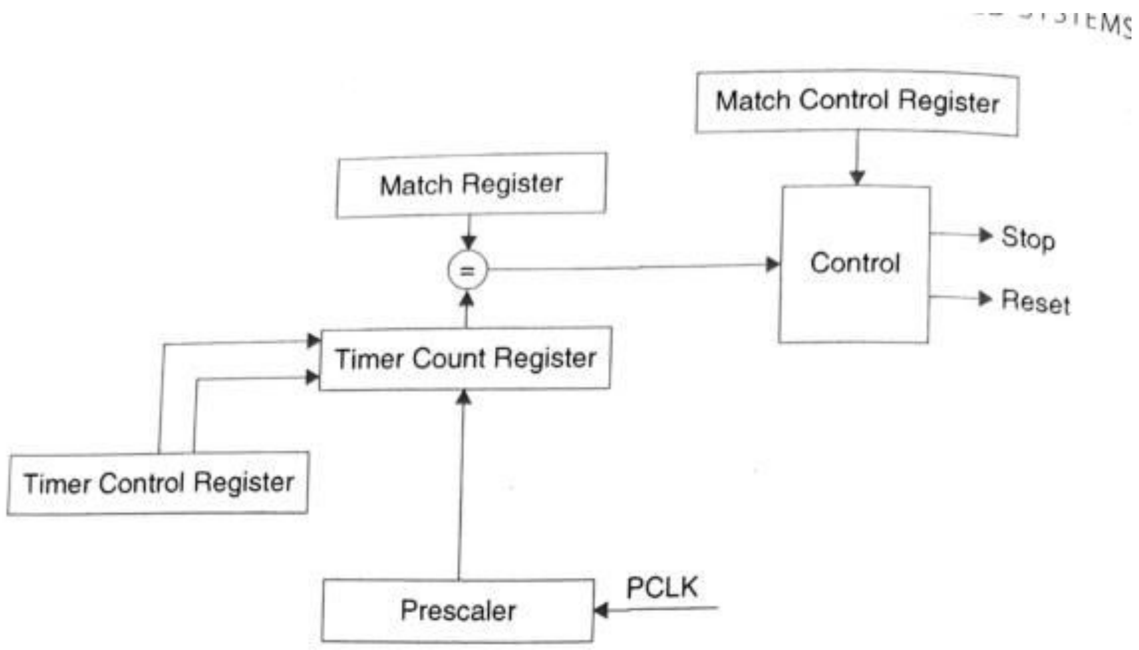
Timer and counter are same function only difference is timer uses the PCLK for timing whereas counter uses external source

Timer is used create time delay whereas counter used to counts number of events occurs

LPC 2148 Micro controller has 2 32 bit timer and associated registers

Timer operation

- Load a number in a match register
- Start the timer by enabling the E bit in TOTCR.
- The TOTC starts incrementing for every tick of the PCLK
- When the content of the TOTC equals the value in the match register, timing is said to have occurred.
- One of many possibilities can be made to occur when this happens
- The possibilities are to reset the timer count register, stop the timer, or generate an interrupt



Timer Registers

- Timer Count(T0TC) -32 bit Register which gives it a range of counting 0 from to 0xFFFF FFFF
- Timer Control(T0TCR)- 8 bit register bit0- enable bit enable bit is 1 count is enabled and starts bit 1 –reset bit

- Match Register(MR0 to MR3) is 32 bit register
- T0MCR is 16 bit Register which is used to specify the event to occur when the match occurs

4.5. PWM

- **Pulse Width Modulation (PWM)** is a technique by which width of a pulse is varied while keeping the frequency constant.
- PWM Pulse is control the period and duty cycle of the square wave
- LPC 2148 microcontroller can be generated up to 6 single edge controlled or 3 double edge controlled PWM outputs, or a mix of both types with help of Seven match registers.
- Double edge controlled PWM outputs can be programmed to be either positive going or negative going pulses.
- A period of a pulse consists of an **ON TIME** cycle (TON) and an **OFF** TIME cycle (TOFF). The fraction for which the signal is TON over a period is known as **duty cycle**.
- Duty Cycle (In %) = (TON/(TON+TOFF) x 100
- E.g. Consider a pulse with a period of 10ms which remains ON (high) for 2ms.The duty cycle of this pulse will be
- D = (2ms / 10ms) x 100 = 20%

Through PWM technique, we can control the power delivered to the load by using ON-OFF signal.

PWM Module Registers

- 32-bit Timer Counter, i.e. PWMTC (PWM Timer Counter). This Timer Counter counts the cycles of peripheral clock (PCLK).
- 32-bit PWM Prescale Register (PWMPR). scale this timer clock counts (PWMPR).
- LPC2148 has 7 PWM match registers (PWMMR0 – PWMMR06).
- . The PWM Latch Enable Register is used to control the update of the PWM Match registers when they are used for PWM generation.

PWM Pulse generation steps

- Whenever starts the PWM Timer. PWM Timer Counter (PWMTC) increments its value step by step and any value matches with these Match Registers then,

- PWM Timer Counter resets, or stops, or generates match interrupt, depending upon settings in PWM Match Control Register(PWMMCR).
- One matchMR0 Registers is used to dedicate for setting the PWM frequency. And other match registers are used to setting the duty cycle of PWM

C Program for generate the Single edge controlled PWM using LPC2148

```
    #include <LPC214x.H>                    /* LPC21xx definitions  */
    #include <stdio.h>
    void init_PWM (void)
    {
        PINSEL0 |= 0x00000008;    /* Enable P0.7 and P0.1 as PWM output */
        PWMPR    = 0x00000000;                 /* Load prescaler  */
        PWMPCR = 0x00000800;/* PWM channel 2 & 3 double edge control, output enabled */
        PWMMR0 = 0xfff;                       /* set cycle rate to sixteen ticks       */
        PWMTCR= 0x00000009;
        PWMMCR=0x00000002; ;                   /* On match with timer reset the counter */
    }
int main(void)
{
        init_PWM();
while(1)                              /* Loop forever */
        {
        PWMMR3 = 0x2fe;                  /* set rising  edge of PWM3 to 100 ticks    */
        PWMLER = 0x8;                         /* enable shadow latch for match 1 - 3   */
        }
}
```

4.6. Real Time Clock(RTC)

- The Real Time Clock (RTC) is a set of counters for measuring time when system power is on, and optionally when it is off.
- It uses little power in Power-down mode.
- On theLPC214x, the RTC can be clocked by a separate 32.768 KHz oscillator or by a programmable prescale divider based on the APB clock
- Measures the passage of time to maintain a calendar and clock.
- Ultra Low Power design to support battery powered systems.
- Provides Seconds, Minutes, Hours, Day of Month, Month, Year, Day of Week, and Day of Year.
- Dedicated 32 kHz oscillator or programmable prescaler from APB clock.
- Dedicated power supply pin can be connected to a battery or to the main 3.3 V.

4.7. Vectored Interrupt Controller

- 32 interrupt request inputs
- **16 vectored IRQ interrupts**
- **16 priority levels dynamically assigned to interrupt requests**
- **Software interrupt generation**

VIC programmably assigns them into 3 categories, FIQ, vectored IRQ, and non-vectored IRQ.

- Fast Interrupt reQuest (FIQ) requests have the highest priority.
- Vectored IRQs have the middle priority, but only 16 of the 32 requests can be assigned to this category.
- Non-vectored IRQs have the lowest priority.

5. Laboratory experiments

Ex.No : **Date :**

AIM:

Waveform generation using 10 bit DAC
To write the embedded C program to generate a triangular and square wave form using internal 10 bit DAC using LPC2148 ARM Micro controller.

APPARATUS REQUIRED :

1. LPC 2148 ARM Microcontroller Development board.
2 Keil µVision version 5
3. Flash Magic .

Input:

Output:
DAC pin (P0.25)

ALGORITHM:

- First, configure P0.25/AOUT pin as DAC output using PINSEL Register.
- Then set settling time using BIAS bit in DACR Register.
- Now write 10-bit value (which we want to convert into analog form) in VALUE field of DACR Register.

[A]. **Square waveform generation using internal DAC of LPC2148**

```
#include <lpc214x.h>
#include <stdint.h>


void delay_ms(uint16_t j)
  {
        uint16_t x,i;
        for(i=0;i<j;i++)
        {
            for(x=0; x<6000; x++);  /* loop to generate 1 milisecond delay with  Cclk = 60MHz */
         }
  }

int main (void)
  {
   uint16_t value;
   uint8_t i;
    i = 0;
    PINSEL1 = 0x00080000; /* P0.25 as DAC output */
while(1)
  {
   value = 1023;
   DACR = ( (1<<16) | (value<<6) );
   delay_ms(100);
   value = 0;
   DACR = ( (1<<16) | (value<<6) );
   delay_ms(100);
   }
 }
```

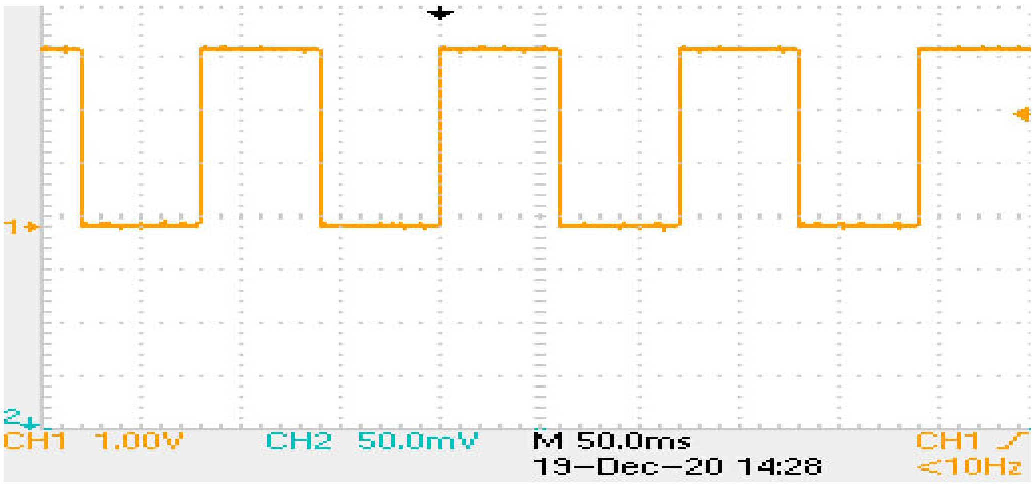
[B]Triangular wave generation using internal DAC of LPC2148

```
#include <lpc214x.h>
#include <stdint.h>
void delay_ms(uint16_t j)
{
  uint16_t x,i;
  for(i=0;i<j;i++)
       {
        for(x=0; x<6000; x++);  /* loop to generate 1 milisecond delay with Cclk = 60MHz */
        }
}

int main (void)
{
   uint16_t value;
```

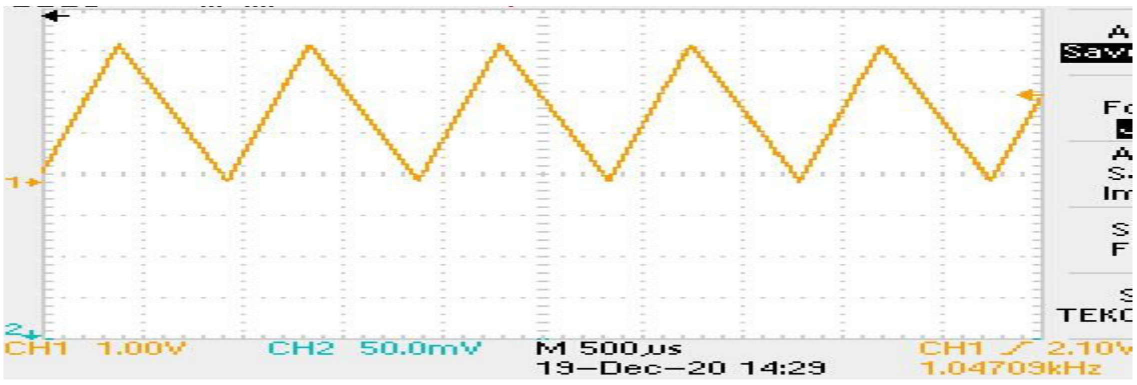
```
    uint8_t i;
    i = 0;
     PINSEL1 = 0x00080000; /* P0.25 as DAC output */
while(1)
  {
     value = 0;
     while ( value != 1023 )
       {
         DACR = ( (1<<16) | (value<<6) );
         value++;
        }
   while ( value != 0 )
       {
         DACR = ( (1<<16) | (value<<6) );
         value--;
        }
     }

  }
```

OUTPUT WAVEFORMS

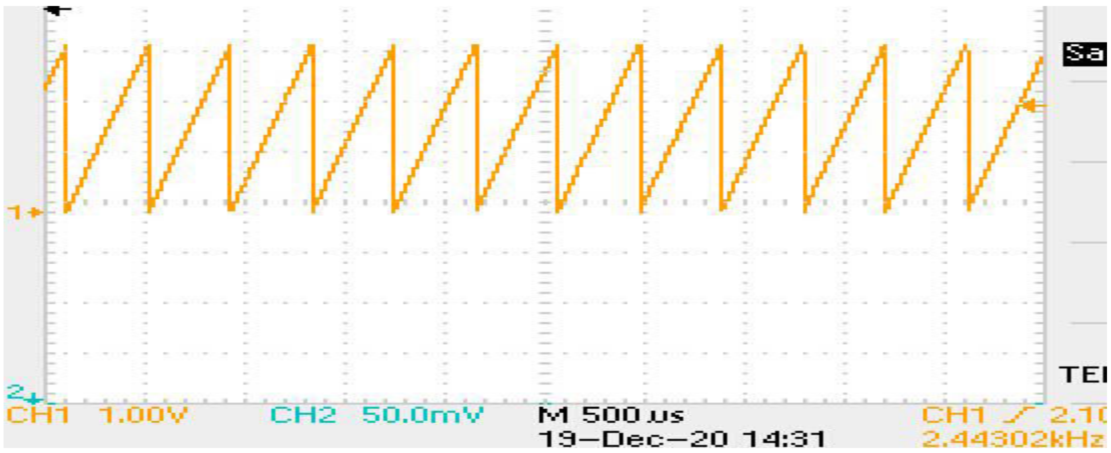
[A] SQUARE WAVE FORM



[B]TRIANGULAR WAVEFORM



[C] SAWTOOTH WAVE FORM



TABULATION

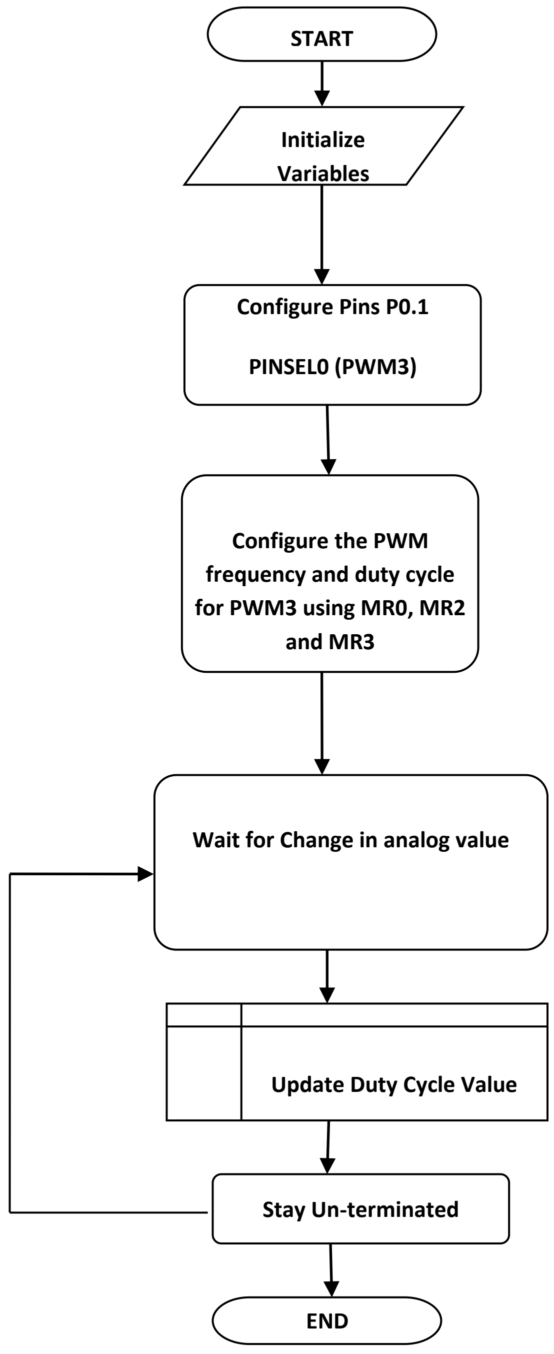
WAVE FORM	AMPLITUDE	TIME(ms)	FREQUENCY(HZ)
Square waveform			
Triangular Waveform			
Saw tooth Waveform			

Result :

Thus the embedded C program to generate the triangular and Square wave form with internal 10 bit DAC using LPC2148 ARM Micro controller was executed and output was verified with oscilloscope.

Pulse Width Modulation(PWM) Waveform generation

Flow Chart


START
Initialize Variables
Configure Pins P0.1
PINSEL0 (PWM3)
Configure the PWM frequency and duty cycle for PWM3 using MR0, MR2 and MR3
Wait for Change in analog value
Update Duty Cycle Value
Stay Un-terminated
END

Ex.No : **Date :**

AIM:

Pulse Width Modulation(PWM) Waveform generation
To write the embedded C program to generate a PWM waveform using LPC2148 ARM Micro controller.

APPARATUS REQUIRED :

1. LPC 2148 ARM Microcontroller Development board.
2 Keil µVision version 5
3. Flash Magic .

Input:

AD0.3 (P0.30) – Place jumper JP5 ('I' Label position)

Output:

PWM3 is used for Demo (**P0.1**)

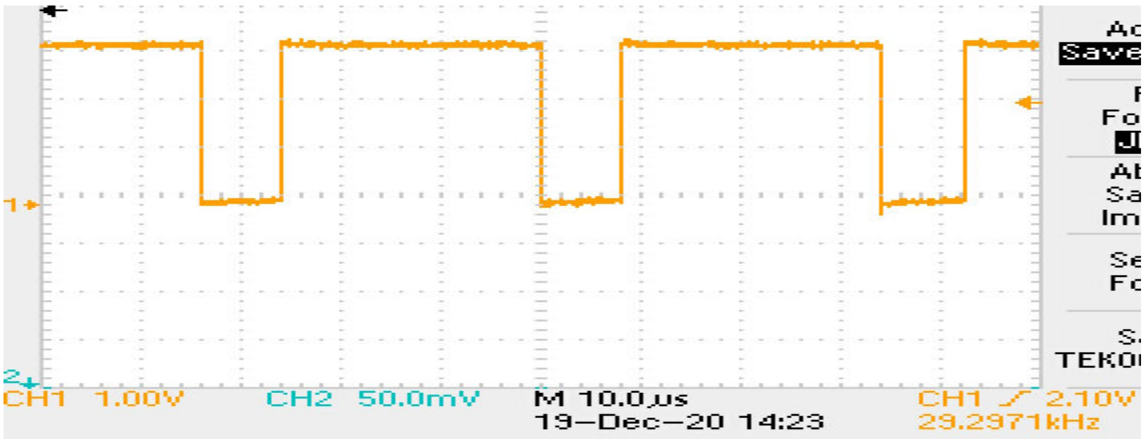
ALGORITHM:

- Reset and disable PWM counter using PWMTCR
- Load prescale value according to need of application in the PWMPR
- Load PWMMR0 with a value corresponding to the time period of your PWM wave
- Load any one of the remaining six match registers (two of the remaining six match registers for double edge controlled PWM) with the ON duration of the PWM cycle. (PWM will be generated on PWM pin corresponding to the match register you load the value with).
- Load PWMMCR with a value based on the action to be taken in the event of a match between match register and PWM timer counter.
- Enable PWM match latch for the match registers used with the help of PWMLER
- Select the type of PWM wave (single edge or double edge controlled) and which PWMs to be enabled using PWMPCR
- Enable PWM and PWM counter using PWMTCR+++

```
Program :
#include <LPC214x.H>                     /* LPC21xx definitions  */
#include <stdio.h>


void PWM0_isr(void)  __irq
{
        PWMIR       |= 0x00000001;               /* Clear match0 interrupt */
        VICVectAddr  = 0x00000000;
}


void init_PWM (void) {

        VICVectAddr8 = (unsigned)PWM0_isr;        /* Set the PWM ISR vector address */
        VICVectCntl8 = 0x00000028;                /* Set channel */
        VICIntEnable = 0x00000100;                /* Enable the interrupt */
        PINSEL0 |= 0x00008008;                    /* Enable P0.7 and P0.1 as PWM output */
        PWMPR    = 0x00000000;                    /* Load prescaler  */

        PWMPCR = 0x00000C0C;                      /* PWM channel 2 & 3 double edge control, output enabled
*/
        PWMMCR = 0x00000003;                      /* On match with timer reset the counter */
        PWMMR0 = 0x400;                         /* set cycle rate to sixteen ticks       */
        PWMMR1 = 0;                           /* set rising  edge of PWM2 to 100 ticks    */
        PWMMR2 = 0x200;                         /* set falling edge of PWM2 to 200 ticks   */
        PWMMR3 = 0x100;                         /* set rising  edge of PWM3 to 100 ticks    */
        PWMLER = 0xF;                         /* enable shadow latch for match 1 - 3   */
        PWMTCR = 0x00000002;                      /* Reset counter and prescaler           */
        PWMTCR = 0x00000009;                      /* enable counter and PWM, release counter from reset */
}


void Delay ()
{
        unsigned int i,j;
        for (i=0;i<250;i++)
                for (j=0;j<700;j++);
}
int main (void)
{
        unsigned long val;
        unsigned long oldval = 0;
        VPBDIV =        0x02;
        PINSEL0         |=      0x00050000;
        PINSEL1         =       0x15400000;
        init_PWM();
        U1LCR  =        0x83;
        U1DLL  =        0xC3;
        U1LCR  =        0x03;
        AD0CR   = 0x00230608;                   /* Setup A/D: 10-bit AIN0 @ 3MHz */
        AD0CR |= 0x01000000;                    /* Start A/D Conversion */
        while (1)
        {                                         /* Loop forever */
                do
                {
                        val = AD0GDR;
```

```
                                /* Read A/D Data Register */
                } while ((val & 0x80000000) == 0);       /* Wait for end of A/D Conversion */
                val = ((val >> 6) & 0x3FF);              /* Extract AIN0 Value */
                Delay (); Delay();

                if (val != oldval)
                {
                        PWMMR2          =       val;
                        PWMLER          =       0xF;
                        oldval    =     val;
                        printf ("Val : %4d \n\r", val);
                }
        }
}
```

MODEL GRAPH



TABULATION

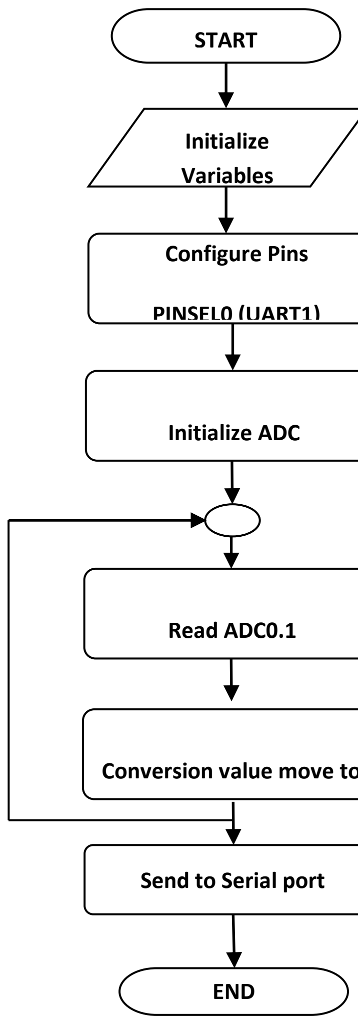
PWM	TON	TOFF	TOTAL TIME	DUTY CYCLE
1.				

Result :

Thus the embedded C program to generate the PWM wave form using LPC2148 ARM Micro controller was executed and output was verified with oscilloscope.

Temperature Sensor(LM 35) Interface

- **Flow Chart**



Ex.No : **Date :**

Temperature Sensor(LM 35) Interface

AIM:

To write the embedded C program to read on-chip ADC value of Temperature sensor LM35 and display in hyper terminal using UART1 using LPC2148 ARM Micro controller.

APPARATUS REQUIRED :

1. LPC 2148 ARM Microcontroller Development board.
2 Keil µVision version 5
3. Flash Magic .

Input:

AD0.1 (P0.28) - LM35 Temp Sensor

Output:

UART0 Pins : P0.8 - Txd | P0.9 – Rxd

Connect Serial Cable at **P2** (Board DB9 connector) to PC’s DB9 Connector.

PROGRAM

```
#include <LPC214x.h>

#include <stdio.h>


#define DONE   0x80000000

#define START  0x01000000

#define PRESET0x00230600

void Delay ()

{

        unsigned int i,j;

        for (i=0;i<50;i++)

                for (j=0;j<500;j++);

}

void Welcome ()

{

        printf ("-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.\n\r");

        printf ("------------------------------------------------------\n\r");

        printf ("*** Temperature Sensor Interfacing with Tyro Kit ***\n\r");

        printf ("------------------------------------------------------\n\r");

        printf (">> Put Jumper J in 'E' Mode to Enable Temp Sensor Block \n\r");

        printf (">> Connect UART1 to COM Port @ 9600 Baud Rate\n\n\r");

        printf ("*******************************************************\n\r");

        printf ("************************* Result ***********************\n\r");

        printf ("*******************************************************\n\n\r");


}
```

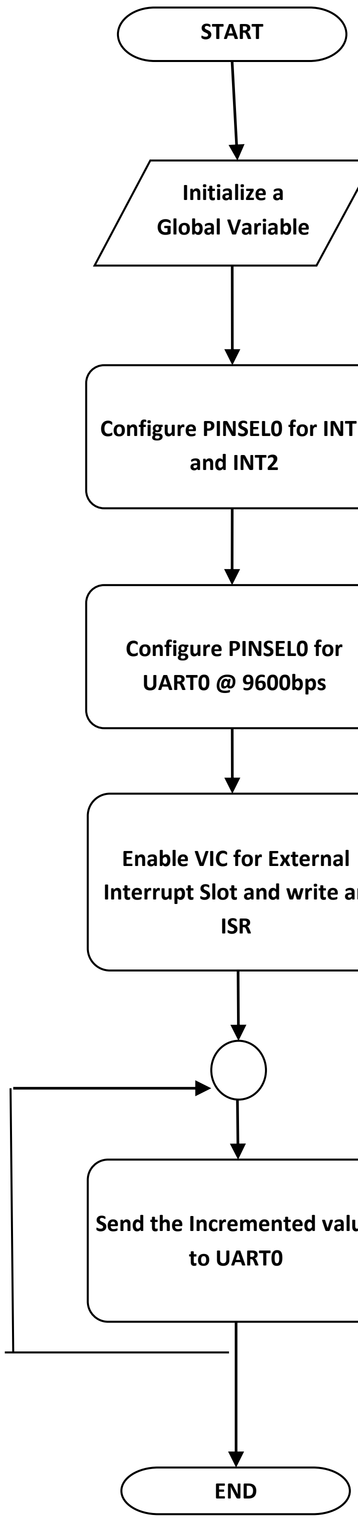
```
void Serial_Init ()

{

        PINSEL0         |= 0x00050000;                  //Configure TxD1 and RxD1 @ P0.8 & P0.9

        U1LCR  =        0x83;

        U1DLL  =        195;

        U1LCR  =        0x03;

}


void main ()

{

        unsigned long Val;

        VPBDIV =        0x02;                           //pclk @ 30MHz

        Serial_Init ();

        PINSEL1         =       0x01 << 24;                     //P0.28 configure as ADC0.1

        Welcome ();

        AD0CR  =        PRESET | 0x02;          //ADC Config: 10bit/9Clock | BURST = 1 | CLKDIV = 0x06

        AD0CR  |=       START;                          //Start Conversion NOW


        while (1)

        {

                do

                {

                        Val = AD0GDR;


                }while ((Val & DONE) == 0);             //Check if Conversion is DONE
                Val = ((AD0GDR >> 6) & 0x3FF);          //Extract Result from Bits AD0GDR.15 - AD0GDR.6


                printf (">> Current Temperature :  %4d ", Val);

                printf ("\xF8\F \r");

        }


}
```

Result

Thus the embedded C program to read on-chip ADC value of Temperature sensor LM35 and display in hyper terminal using UART1 using LPC2148 ARM Micro controller was executed and output was verified.
.

Study of the external Interrupts

Flow Chart


START
Initialize a Global Variable
Configure PINSEL0 for INT1 and INT2
Configure PINSEL0 for UART0 @ 9600bps
Enable VIC for External Interrupt Slot and write an ISR
Send the Incremented value to UART0
END

Ex.No : **Date :**

Study of external Interrupts

AIM:

To write the embedded C program to read the external interrupts INT1 and INT2 and display in hyper terminal using UART1 using LPC2148 ARM Micro controller.

APPARATUS REQUIRED :

1. LPC 2148 ARM Microcontroller Development board.
2 Keil µVision version 5
3. Flash Magic .

Input:

INT 1 (P0.14) –Press to Increment the data

INT 2 (P0.15) – Press to Increment the data

Output:

UART0 Pins : P0.8 - Txd | P0.9 – Rxd

Connect Serial Cable at **P1** (Board DB9 connector) to PC's DB9 Connector.

PROGRAM

```
#include <lpc214x.h>

#include <stdio.h>

int volatile EINT1          =         0;

int volatile EINT2          =         0;

void ExtInt_Serve1(void)__irq;

void ExtInt_Serve2(void)__irq;


/*---------------------------------<   INT2 Initialization  >-----------------------------*/

void ExtInt_Init2(void)

{

          EXTMODE  |= 4;          //Edge sensitive mode on EINT2

          EXTPOLAR = 0;           //Falling Edge Sensitive

          PINSEL0  |= 0x80000000;         //Enable EINT2 on P0.15

          VICVectCntl1 = 0x20 | 16;        //Use VIC1 for EINT2 ; 16 is index of EINT2

          VICVectAddr1 = (unsigned long) ExtInt_Serve2;           //Set Interrupt Vec Addr in VIC1

          VICIntEnable |= 1<<16;          //Enable EINT2

}


/*---------------------------------<   INT1 Initialization  >-----------------------------*/

void ExtInt_Init1(void)

{

          EXTMODE  |= 2;                                //Edge sensitive mode on EINT1

          EXTPOLAR = 0;                                 //Falling Edge Sensitive

          PINSEL0  |= 0x20000000;                       //Enable EINT2 on P0.14

          VICVectCntl0 = 0x20 | 15;                     //Use VIC0 for EINT2 ; 15 is index of EINT1

          VICVectAddr0 = (unsigned long) ExtInt_Serve1;           //Set Interrupt Vec Addr in VIC0
```

```
        VICIntEnable |= 1<<15;                      //Enable EINT1

}


/*---------------------------------<   Serial Initialization  >-----------------------------*/

void Serial_Init(void)

{

        PINSEL0 |=      0X00000005;             //Enable Txd0 and Rxd0

        U0LCR  =        0x00000083;             //8-bit data, no parity, 1-stop bit

        U0DLL  =        0x00000061;     //XTAL = 12MHz (pclk = 60 MHz, VPB = 15MHz(pclk=cclk/4)

        U0LCR  =        0x00000003;                                     //DLAB = 0;

}

/*-----------------------------------<   Delay Function  >---------------------------------*/

 void DelayMs(unsigned int count)        {


   unsigned int i,j;

   for(i=0;i<count;i++)

        {

                for(j=0;j<1000;j++);

        }

}


/*-----------------------------------<   Main Function  >---------------------------------*/


void main(void)

{

        Serial_Init();

        ExtInt_Init1();                                                 //Initialize Interrupt 1
        ExtInt_Init2();                                                     //Initialize Interrupt 2

        putchar(0x0C);
        printf ("***************** External Interrupt Demo ***************************\n\n\r");
        DelayMs(100);
        printf (">>> Press Interrupt Keys SW2(INT1) and SW3(INT2) for Output... \n\n\n\r");
        DelayMs(100);

        while(1)
        {
                DelayMs(500);
                printf("INT1 Generated : %d  |  INT2 Generated : %d \r", EINT1, EINT2);
                DelayMs(500);
        }

}

/*----------------------------------<   Interrupt Sub-Routine  >-----------------------------*/

void ExtInt_Serve1(void)__irq
{
        ++EINT1;
        EXTINT |= 2;
        VICVectAddr = 1;
}

void ExtInt_Serve2(void)__irq
```

```
{
	++EINT2;
	EXTINT |= 4;
	VICVectAddr = 0;
}
```

Result

Thus the embedded C program to read the external interrupts INT1 and INT2 and display in hyper terminal using UART1 using LPC2148 ARM Micro controller was executed and output was verified.

6. Post test-Multiple Choice Questions

1.LPC 2148 microcontroller is based on ----------------------architecture

A. ARM5
B. ARM7
C. ARM 9
D. ARM11

2.LPC 2148 microcontroller operated voltage is ---------V

A. 5
B. 5.5
C. 3.3
D. 3

3.LPC2148 has

A. 32kB on chip RAM and 128kB FLASH memory
B. 1MB RAM and 512kB on chip FLASH memory
C. 64kB RAM and 512kB on chip FLASH memory
D. 32kB RAM and 512kB on chip FLASH memory

4.ADC in LPC 2148 microcontroller is ----------------------- successive approximation register type

A. 16 bit
B. 12 bit
C. 10 bit
D. 32 bit

5. ADC in LPC 2148 microcontroller is ----------------------- type

A. 10 bit successive approximation register
B. 10 bit Dual slope
C. 10 bit parallel
D. None of the mentioned above options

6.The -------------------------- block is to configure the Pins to the desired functions

A. Pinstatus
B. pinselect
C. pincontrol

D. None of the above mentioned options

7. Conclusion

Hence the in this module we studied about the LPC Micro controller block diagram and its programming in detail

8. References

1. Lyna B.Das ,Embedded Systems – An Integrated Approach, Pearson Education 2014.

2. ARM Architecture Reference Manual - David Seal.

3. https://www.nxp.com/docs/en/user-guide/UM10139.pdf

9. Assignment

1. Write the short notes on Real Time clock and its purpose in LPC 2148 microcontroller

2. Write a C Program for generate double edge controlled PWM pulse with help of LPC 2148 microcontroller

3. Write a C Program for generate the saw tooth wave form with help of DAC module in LPC 2148 microcontroller