Design And Analysis Of Microcontroller using Arm processor Involved various Interfacing

1.Abstract;

In this work Microcontroller is a test using various methods. Pulse Width Modulation (PWM) is a very useful approach for controlling analog circuits with the processor's virtual outputs. PWM is used in a wide variety of applications starting from size and communications to energy management. PWM is a well-known method to generate a signal of various responsibility cycle. Here we can use it to govern the brightness of LED the usage of LPC2148. Interrupt due to an outside source which includes an external switch, sensor, or tracking device. These interrupt are special events that require instant attention. An interrupt can be configured to be prompt via a variety of triggers. We've got a programmed timer in LPC2148 to generate a particular delay. Here we'll generate interrupt to carry out an identical task. Throughout this educational series, we'll use interrupt with numerous peripherals of LPC2148 ARM7. In well-known, timer approach precisely how it sounds. Timer and counter may be a very essential feature which lets in us to offer time variable to our microcontroller based totally task. Most microcontrollers come with a built-in timer peripheral. Here in this feature, we'll use Timer0 to generate a particular put off of 1 sec. We have to test the fee in Timer Counter Register (T0TCR) and wait till T0TC reaches one thousand counts. Which will then generate one thousand ms i.E. 1 sec. Time put off. Microcontroller to layout sophisticated embedded utility. We have now not blanketed exclusive styles of stage triggering configurations of the outside interrupt. I agree with our mind is still warm with it. To hold simplicity in our example undertaking we'll blink LED whilst interrupt is generated after every 0.5 Sec.

Keywords: Microcontroller,pwm,interfacing,timer,counter,embedded

Acronyms:

[1] RETI -Return from Interrupt
[2] PWM- Pulse Width Modulation
[3] ISR -Interrupt Service Routine
[4] ADC- Analog-to-Digital Converter
[5] UART- "Universal Asynchronous Receiver/Transmitter
[6] LCD- Liquid Crystal Display
[7] TTL- Transistor-Transistor Logic.
[8] ADHD- Attention Deficit Hyperactivity Disorder
[9] RTC- Real-Time-Clock
[10] LSB- least significant bit

2.Introduction:

Microcontrollers do the whole thing with ones and zeros. That approach microcontroller works with 3.3V and 0V as virtual 1 & zero[1]. It can't produce for example 1V or 2.5V or another price specific than 0V and 3.3V. Here PWM function allows us to generate any voltage level between 0V and 3.3V. Now we can see the way it's been done using PWM so that we will control the brightness of an LED[2]. This might be the first-rate manner to see the impact of PWM. Before we proceed any further permit's talk a little bit approximately responsibility cycle. The emergence of strong green merchandise and systems are acknowledged to satisfy the call for the energy consumption of seven billion populations of the world. Light Emitting Diode (LED) is a low power intake, efficient, environment pleasant small bulb and produces no heat during operation[3] In comparison to different lighting fixtures technology, LED light tends to be directional. This is a disadvantage for most popular lighting programs but can be an advantage for a spot or flood lighting [4]. Globally, electric powered lighting fixtures money owed for a few one-fifth of all-electric power ate up [5]. Therefore any green electric lights initiative is a subject of the hobby of all mankind Dimming led saves power at a kind of 1:1 ratio. This way that if you dim LEDs right down to 50% of their output you will shop nearly 50% of your energy utilization [6]. Dimming LEDs run cooler than the existing tube lights and bulbs, which have to enlarge the life of the digital additives of the driving force, as well as prosper on the LEDs. Dimming of the light output of the White LED down panel is finished with pulse width modulation so that you can keep a regular coloration or color temperature of light [7]. They use the ARM7 based 32-bit microcontroller i.E. LPC2148, designed through the NXP as the controller of the White LED down panel.[8] White LED down panel is hooked up to the GPIO pins of the LPC2148 microcontroller with one of a kind intermediate modules to supply the right contemporary to force the LED down panel. This paper examines the implementation of Pulse Width Modulation (PWM) to take a look at and evaluate the exceptional results of PWM the usage of the internal module of LPC2148 microcontroller and GPIO pins (without referring to the built-in module of the microcontroller) and examine with the various approach.[9] The Wi-Fi module continuously transmits or receives serial records (RS232 protocol) through the internet without wires[10].It can provide the received statistics and receives the statistics to be transmitted to and from a bunch system through a host controller interface (HCI). The most famous host controller interface these days is both a UART or a USB. Here, I will most effective consciousness on the UART interface; it could be without problems show how a Bluetooth module can be integrated directly to a number system through a UART connection[11]. The interrupt is a sign from a tool connected to a computer or from a program inside the controller that reasons principal software to forestall and parent out what to do next. ISR (Interrupt Service Routine) is done when an interrupt takes place. A segment of an application that takes control whilst an interrupt is received and carry out the operations required to provide the interrupt[12]**.** There are several approaches we will clock ARM Microcontroller. One manner is to apply External Clock with duty cycle 50-50 and in a frequency variety 1 MHz to 50 MHz linked to XTAL1 Pin. The second way is through connecting External Crystal Oscillator but its range is decreased between 1 MHz to 30 MHz. We also can use an on-chip PLL Oscillator but right here outside clock frequency should no longer exceed variety from 10 MHz to 25 MHz[13]. A Microcontroller can't make any choices on controlling something in the actual international without sensing something about it. The best component for a microcontroller to test is the reputation of a switch[14].Is it open or closed? Switches can be used as operator controls. They can be used for many purposes, however, they may be in the handiest one in all states: ON (closed) or OFF (open). GPIO related registers and their capabilities. We want to replace 'x' with the port variety to get the check in call[15]. Say for example IOxPIN sign-in will become IO0PIN whilst used for PORT0 and IO1PIN when used for PORT1. These all check-in names described in the header record "lpc2148.H" Register names defined within the header record are not anything but a pointer which holds the cope with of actual sign-in in LPC2148.[16]

3.Research methodology

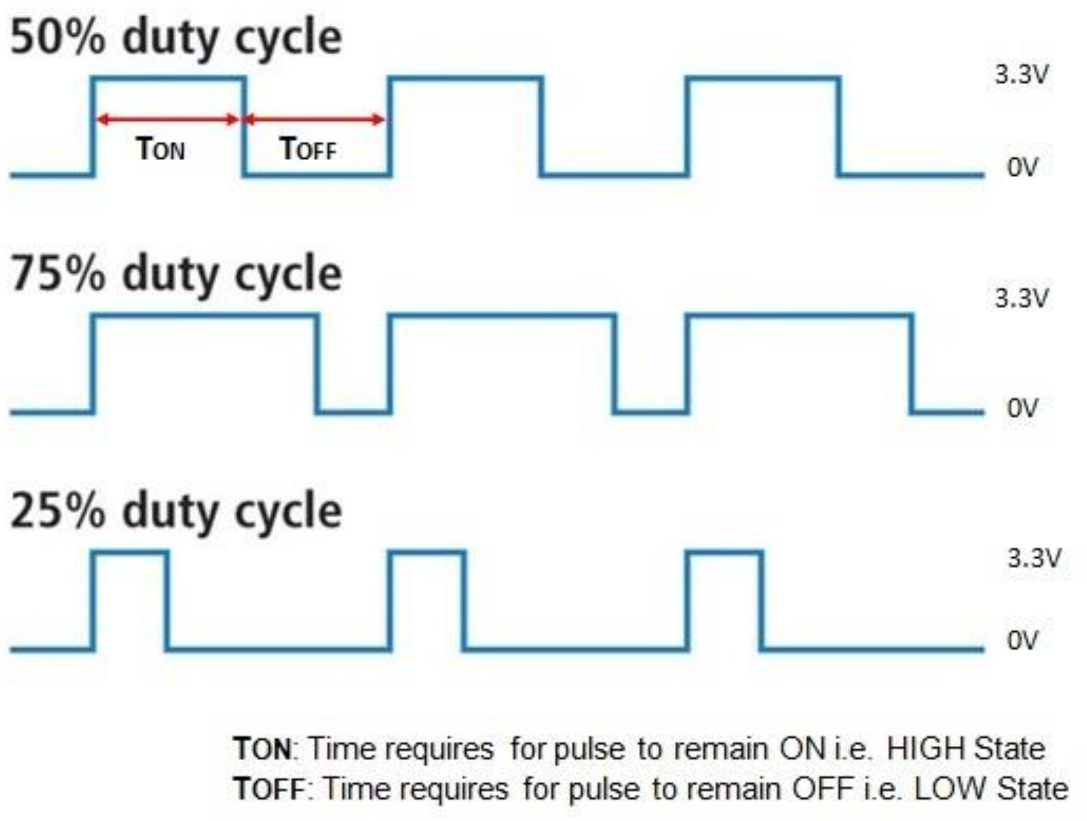
3.1 PWM using ARM7 microcontroller:

Pulse Width Modulation (PWM) is a very beneficial approach for controlling analog circuits with the processor's virtual outputs. PWM is utilized in a huge variety of packages ranging from dimension and communications to electricity control[17]. PWM is a well-known technique to generate a signal of the various duty cycle. Here we will use it to govern the brightness of LED the usage of LPC2148. Microcontrollers do the whole thing with ones and zeros. That way microcontroller works with three.3V and 0V as digital 1 & 0[18]. It can't produce as an instance 1V or 2.5V or any other price different than 0V and 3.3V. Here PWM characteristic allows us to generate any voltage stage among 0V and 3.3V. Now we will see how it's been completed using PWM so that we can manage the brightness of an LED[19]. This might be an exceptional way to look at the impact of PWM. Before we continue any also let's discuss a little bit approximately the responsibility cycle[20]. When the sign is HIGH, we call this "ON time". To describe the amount of "on-time"(TON), we use the idea of the responsibility cycle. The duty cycle is measured in percent. The percentage obligation cycle describes the overall performance of a virtual signal ON over a c program language period or time frame. This duration is the inverse of the frequency of the waveform[21].If the digital signal spends half of the time ON and remaining OFF, we'd say this virtual sign has a duty cycle of 50%. If the percent is better than 50% which means virtual signal spends more time in HIGH country than LOW state then we'd say responsibility cycle is higher than 50%[22]

Duty cycle = Ton/Ton+Toff

Effenicy of Duty cycle(%) =Ton/Ton+Toff*100

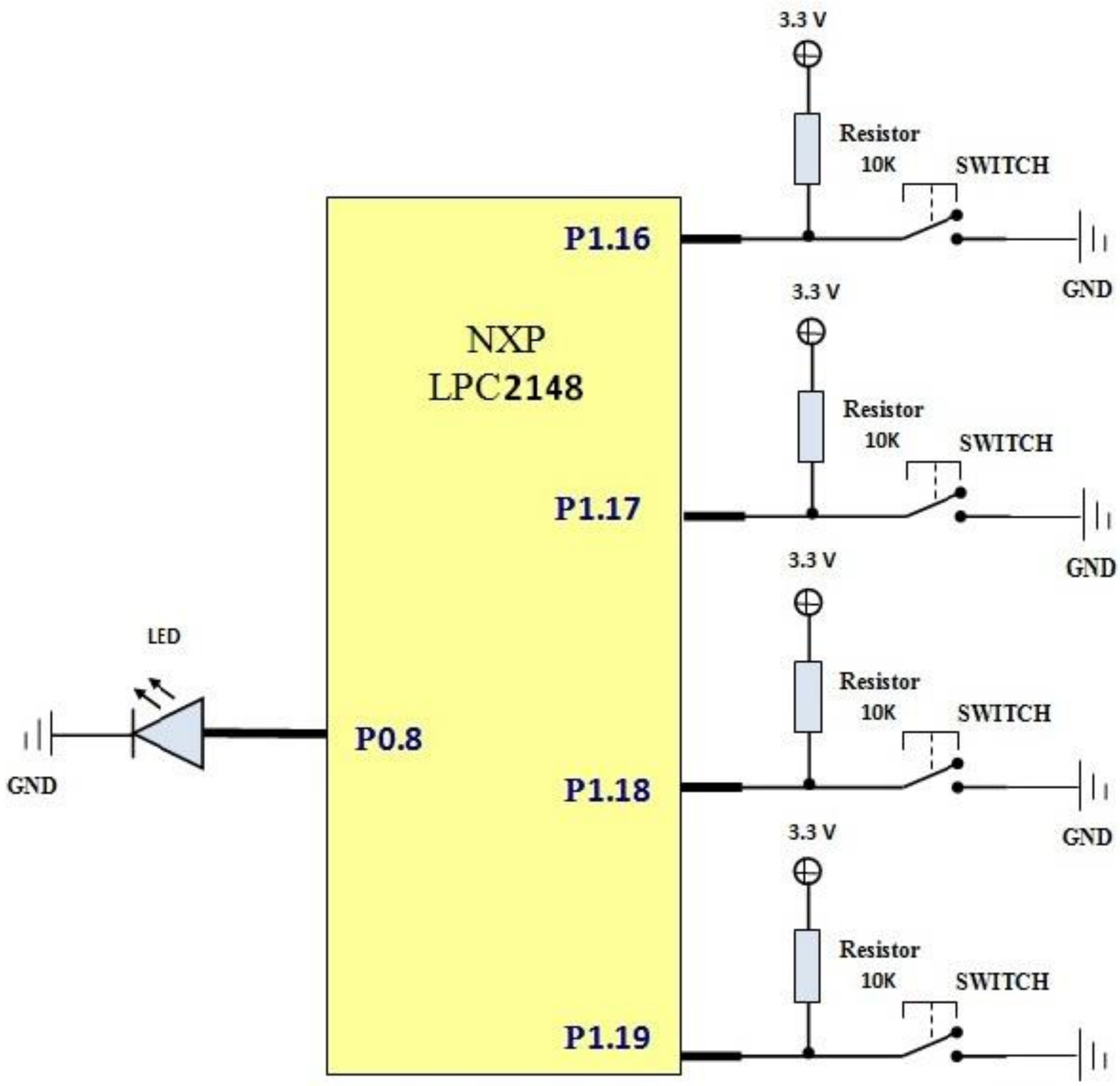
Setup the Pin Select Register to pick out PWM for the port we need Setup the Clock Divider Register for the PCLK (VPBDIV) Setup the PWM Timer Prescale Register Setup the IO direction for the PWM pin you intend to apply as output Enable the specific PWM channel you need to apply within the PWM Control Register(PWMCR) Set the maximum wide variety of counts in a single cycle[23]. This is executed in Match Register zero (PWMMR0) Set the value for duty cycle in the particular healthy sign in you need to apply (PWMMRx wherein "x" is from 1 to 6) Setup the PWM Match Control Register to cause a counter reset while the fit sign-in takes place (PWMMCR) Set the PWM latch Enable Register to enable using match cost (PWMLER0 Reset the timer counter the use of a bit in the PWM Timer Control Register (PWMTCR) Enable the timer counter and permit the PWM mode the usage of the PWM timer manipulate register (PWMTCR).[24]


50% duty cycle
TON
TOFF
3.3V
0V
75% duty cycle
3.3V
0V
25% duty cycle
3.3V
0V
TON: Time requires for pulse to remain ON i.e. HIGH State
TOFF: Time requires for pulse to remain OFF i.e. LOW State

3.1.2PWM interface with arm processor:

PWM in LPC2148 ARM7 looks complex than the overall motive timers. However, it is a further preferred motive timer with a few extra hardware. The PWM in LPC2148 can generate six channels of the unmarried side managed PWM or 3 channels of twin edge controlled PWM. There are 7 registers to house the PWM with sign in 0 getting used to set base frequency (f =1/T). Thus for the reason that there may be the simplest one base frequency sign up all 6 channels must have the base frequency. PWM in LPC2148 may be single side or double side. On the double side, the PWM ending point of pulse and starting point of the heartbeat is variable and can be set every cycle. In a single area PWM simplest the finishing fringe of the PWM is a variable and the beginning area is constantly set at the bottom frequency. Single aspect calls for one sign up (plus the bottom register) so you may have 6-unmarried edge channels of PWM on the LPC2148. For double edge PWM. You want a base sign in plus a beginning aspect check-in plus and finishing side check-in so that the most effective 3-channels of double part PWM is to be had at the LPC2148. You can blend unmarried and double facet channels so long as you don't run out of 6 to be had registers. Instead of spending extra time, I would prefer to begin straight to enforce example venture. But earlier than we get any besides let's have a brief take a look at PWM registers in LPC2148 Microcontroller. We should manipulate the brightness of LED related at Pin P0.8 of LPC2148. We might be the usage of the duration of 10ms. We additionally ought to connect Four Switches at Pin P1.16, P1.17, P1.18, and P1.19 which controls the heartbeat width and so whilst we press any switch among those. We will get equivalent brightness.

3.1.3 circuit diagram:



3.1.4Registers and function:

PWMTCR	PWM Timer Control Register– PWMTCR is used to manipulate the timer counter functions. The Timer Counter can be disabled or reset via the PWMTCR
PWMPR	PWM Prescale Register– The PWMTC (PWM Timer Counter) is incremented each PWMPR+1 cycles of PCLK.
PWMMR0-PWMMR6	PWM Match Register 0- PWM Match Register 6– PWMMR0-6 can be enabled through PWMMCR to reset the PWMTC, stop each the PWMTC and PWMPC, and/or generate an interrupt whilst it suits the PWMTC. Also, a suit among PWMMR0-PWMMR6 and the PWMTC units all PWM outputs which are unmarried side mode and units PWM1 if it's miles in double-edge mode.
PWMMCR	PWM Match Control Register– The PWMMCR is used to manipulate if an interrupt is generated and if the PWMTC is reset when fit occurs.
PWMIR	PWM Interrupt Register– The PWMIR may be written to clean interrupt. The PWMIR maybe examine to pick out which of the viable interrupt sources are pending.
PWMLER	PWM Latch Enable Register– Enables use of latest PWM Match values
PWMPCR	PWM Control Register– Enables PWM outputs and selects PWM channel kinds as both single sides or double area managed.

3.1.5 Program:

```
#encompass <lpc214x.H>

void initPWM(void);                    // Initialize PWM

void initClocks(void);          // Setup PLL and Clock Frequency

int important(void)

initClocks(); //Initialize CPU and Peripheral Clocks @ 60Mhz

initPWM(); //Initialize PWM

//IO1DIR = 0x1; This is not wanted!

  //Also via default all pins are configured as Inputs after MCU Reset.

   Even as(1)

    if( !((IO1PIN) & (1<<16)) ) // Check P1.Sixteen

    PWMMR4 = 2500; //T-ON=25% , Hence 25% Bright

    PWMLER = (1<<4); //Update Latch Enable bit for PWMMR4

    else if( !((IO1PIN) & (1<<17)) ) // Check P1.17

    PWMMR4 = 5000; //50% Bright

      PWMLER = (1<<4)

else if( !((IO1PIN) & (1<<18)) ) // Check P1.18

PWMMR4 = 7500; //seventy five% Bright

PWMLER = (1<<four);

else if( !((IO1PIN) & (1<<19)) ) // Check P1.19

PWMMR4 = ten thousand; //one hundred% Bright
```

```
PWMLER = (1<<four);

 //return 0; //generally this wont execute ever

void initPWM(void)

PINSEL0 = (PINSEL0 & ~(1 << 16 (1 << 17); // Select PWM4 output for Pin0.Eight

PWMPCR = 0x0; //Select Single Edge PWM - by using default its single Edged so this line can be eliminated

PWMPR = 60-1; // 1 micro-second decision

PWMMR0 = ten thousand; // 10ms length duration

PWMMR4 = 500; // zero.5ms - pulse period i.E width (Brigtness level)

PWMMCR = (1<<1); // Reset PWMTC on PWMMR0 suit

PWMLER = (1<<0)four); // update MR0 and MR4

PWMPCR = (1<<12); // permit PWM output

PWMTCR = (1<<1) ; //Reset PWM TC & PR

//Now , the final second - allow everything

PWMTCR = (1<<zero (1<<three); // enable counters and PWM Mode

//PWM Generation goes energetic now - LED must be 25% Bright after boot!!

 //Now you can get the PWM output at Pin P0.Eight!

Void initClocks(void)

PLL0CON = 0x01;   //Enable PLL

PLL0CFG = 0x24;   //Multiplier and divider setup

PLL0FEED = 0xAA;  //Feed collection

PLL0FEED = 0x55;

while(!(PLL0STAT & 0x00000400)); //is locked?

PLL0CON = 0x03;   //Connect PLL after PLL is locked

PLL0FEED = 0xAA;  //Feed series

PLL0FEED = 0x55;

VPBDIV = 0x01;    // PCLK is equal as CCLK i.E.60 MHz
```

3.2External interrupt involved ARM7 microcontroller

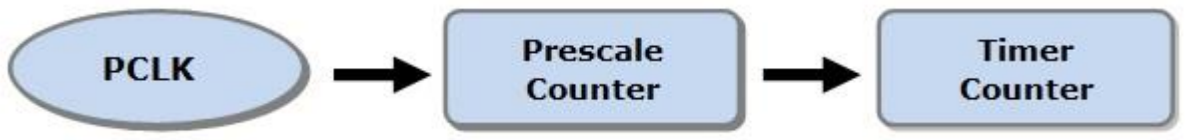
The interrupt is caused by an outside supply consisting of an outside switch, sensor, or tracking tool. These interrupt are unique events that require immediate attention. An interrupt can be configured to be activated by way of a selection of triggers[25]. Say as an instance: the chip awaits a signal alternate on a pin connected to a switch or sensor and timed interrupts. Seven Pins of PORT0 on LPC2148[26] Microcontroller may be configured as external interrupt input The external interrupt characteristic has four registers associated with the EXTINCT join up incorporates the interrupt flags, and the EXTWAKEUP sign up contains bits that allow Individual external interrupts to wake up the microcontroller from Power-down mode. The EXT MODE and EXTPOLAR registers specify the extent and part sensitivity parameters[27]

3.2.2 Register ,function and its working:

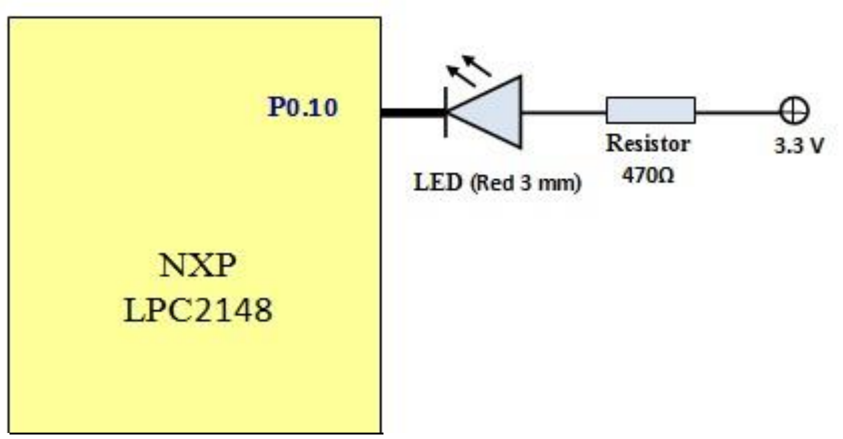
EINT0	External Interrupt Input zero	Pins P0.1 and P0.Sixteen may be selected to carry out the EINT0 function
EINT1	External Interrupt Input 1	Pins P0.Three and P0.14 can be decided on to perform the EINT1 feature
EINT2	External Interrupt Input 2	Pins P0.7 and P0.15 may be selected to perform EINT2 characteristic
EINT3	External Interrupt Input 3	Pins P0.Nine, P0.20 and P0.30 can be selected for EINT3

3.2.3Timer interfacing uing arm processor:

Timer and counter may be a very essential function which lets in us to offer time variable to our microcontroller based undertaking. Most microcontrollers come with a built-in timer peripheral. The LPC2148 has two functionally identical standard purpose timers: Timer0 and Timer1. Each timer is 32-bit in conjunction with a 32-bit Prescaler[28]. Timer permits us to generate a unique time postpone. For Example: In our blink LED instance undertaking, we've generated a random postpone of approximate 1 Sec. However, the usage of Timers will generate an accurate time delay. We'll get into that at the same time as discussing the instance task of Timer[29]. Apart from this, we can use timers as pulse-width modulators and additionally as unfastened going for walks timer. Timers in LPC2148 ARM7 Microcontroller allows us to do genuinely cool stuff. Also, the timer enhances using a microcontroller in lots of specific approaches. We may additionally want to dedicate one greater put up to understand Match and Capture registers and their uses in the actual international application[30]. Let's first recognize the unfastened running counter and related basics. The heart of timers of the LPC2148 Microcontroller is a 32-bit unfastened jogging counter, that's designed to matter cycles of the Peripheral Clock (PCLK) or an external clock, this counter is programmable with a 32-bit Prescaler[32]. The tick fee of the Timer Counter (TC) is controlled by the 32-bit variety written in the Prescaler Register (PR) in the following way. There is a Prescale Counter (PC) which increments on every tick of the PCLK. When it reaches the price in the Prescaler sign in, the timer count number is incremented and the Prescaler Counter (PC) is reset, on the following PCLK. This motivates the timer counters to increment on every PCLK while PR=0, every 2 PCLKs whilst PR=1, and many others.[31]



In LPC2148 ARM7 Microcontroller, The VIC is a component from the ARM top cell range of modules that is an extraordinarily optimized interrupt controller. The VIC is used to address all the on-chip interrupt resources from peripherals. Each interrupt source is connected to the VIC on a set channel. Our application software can connect those channels to the CPU interrupt lines (FIQ, IRQ) Interrupt is a signal from a device connected to a pc or from software inside the controller that reasons the main application to stop and parent out what to do subsequent. ISR (Interrupt Service Routine) is performed when an interrupt happens. A section of software that takes manipulates while an interrupt is acquired and performs the operations required to provide the interrupt. Whenever any tool desires carrier of microcontroller, the tool notifies the microcontroller using sending interrupt signal. Upon receiving an interrupt signal, the microcontroller stops or interrupts predominant program float and saves the address of the following practice (PC) at the stack pointer (SP). It jumps to a hard and fast location in memory, known as interrupt vector desk that maintains the address of the ISR (Interrupt Service Routine). Each interrupt has its very own ISR. The microcontroller receives the address of the ISR from the interrupt vector table and soars to it. It starts to execute the Interrupt Service Routine until it reaches the remaining education of the subroutine which is RETI (Return from Interrupt). RETI no longer utilized in C Coding. (Fig: Interrupt and ISR Relation)Upon executing the last education in Interrupt Service Routine the microcontroller returns to the place in which it left off or interrupted previously. And first, it gets the program counter (PC) address from the stack pointer by using popping the pinnacle two bytes of the stack into the PC. Then it begins to execute from that cope with and continue executing the predominant program. Whenever any tool wishes service of microcontroller, the tool notifies the microcontroller by way of sending an interrupt sign. Upon receiving an interrupt signal, the microcontroller stops or interrupt primary software glide It jumps to a set place in memory, referred to as interrupt vector table that keeps the cope with of the ISR (Interrupt Service Routine). Each interrupt has its very own ISR. The microcontroller gets the address of the ISR from the interrupt vector table and soars to it. It starts to execute the Interrupt Service Routine till it reaches the remaining guidance of the subroutine that is RETI (Return from Interrupt). RETI is now not utilized in C Coding. Interrupt and ISR Relation) Upon executing the last coaching in Interrupt Service Routine the microcontroller returns to the region in which it left off or interrupted formerly. And first, it receives the program counter (PC) address from the stack pointer via popping the top bytes of the stack into the PC.



Then it starts off evolved to execute from that deal with and hold executing important program

3.2.4 Interpet and function:

PC	Prescale Counter: The 32-bit PC is a counter that is incremented to the cost stored in PR (Prescale Register) whilst cost in PR is reached, The TC (Timer Counter) is incremented, and PC is cleared. The PC is observable and controllable thru the bus interface
PR	Prescale Register: The 32-bit sign-in which keep the most price of prescale counter and then it reset
TC	Timer Counter: This is 32-bit Timer Counter which receives incremented whenever PC Prescale Counter value reaches its maximum cost as laid out in PR
TCR	Timer Control Register: The timer Control register is used to control the timer manipulate features. We'll permit, disable and reset Timer Counter (TC) via this check-in CTCR
CTCR	Count Control Register: This register selects Timer Counter Mode. In our instance,we have used Timer Mode. This can be achieved by way of setting CTCR to 0x0. [In Timer Mode every rising PCLK edge can increment Timer's Prescale Counter (PC) or clear PC and increment Timer Counter (TC)]

3.2.5 program:

```
Void delay_ms(unsigned int counts) //Using Timer0

 T0TCR = 0x02;        //Reset Timer

T0TCR = 0x01;        //Enable timer

 even as(T0TC < counts); //wait until TC reaches the favored postpone

 T0TCR = 0x00;        //Disable timer
```

Setting Clock Configurations of Clock and PLL in LPC2148 ARM7 Microcontroller. There are several approaches we can clock ARM Microcontroller. One manner is to use an External Clock with a duty cycle of 50-50 and in a frequency range of 1 MHz to 50 MHz linked to XTAL1 Pin. The 2nd way is via connecting External Crystal Oscillator however its variety is lower among 1 MHz to 30 MHz. We also can use an on-chip PLL Oscillator however here external clock frequency ought to not exceed the range from 10 MHz to 25 MHz. In this educational, we can deal with External Crystal with PLL. External Clock source and External Crystal most effective can be discussed in the future because it's no longer required at this moment.

4.Result:

We will use PWM in LPC2148 ARM7 Microcontroller to control the brightness of the LED. We can do much stuff by way of the era of the varying duty cycle. Outside interrupt in LPC2148 ARM7 Microcontroller to design sophisticated embedded applications is an Interrupt Service Routine (ISR). As we've got used Timer0 as a supply of interrupt. An MR0 match event increases an IRQ (Interrupt Request). Because we understand Timer zero is a source of interruption so read present-day cost in Timer0's Interrupt Register i.E. T0IR. When MR0 in shape occasion occurs we'll toggle LED linked to Pin P0.10. We additionally ought to make certain to clean interrupt flag 'T0IR = readVal' and speak to for cease of interrupt via putting VICVectAddr= 0x0). I agree with our mind is still warm with it. To hold simplicity in our example undertaking we'll blink LED whilst interrupt is generated after every 0.5 Sec. We have to test the price in Timer Counter Register (T0TCR) and wait till T0TC reaches a thousand counts. Which will then generate one thousand ms i.E. 1 sec. Time put off.

5.Discussion:

This paper we discuss the design and analysis of arm processor microcontroller Interrupt is a signal from a device connected to a pc or from software inside the controller that reasons the main application to stop and parent out what to do subsequent. ISR (Interrupt Service Routine) is performed when an interrupt happens). Because we understand Timer zero is a source of interruption so read present-day cost in Timer0's Interrupt Register i.E. T0IR. When MR0 in shape occasion occurs we'll toggle LED linked to Pin P0.10. We additionally ought to make certain to clean interrupt flag 'T0IR = readVal' and speak to for cease of interrupt via putting VICVectAddr= 0x0). . We have to test the price in Timer Counter Register (T0TCR) and wait till T0TC reaches a thousand counts. Which will then generate one thousand ms i.E. 1 sec. Time put off. I agree with our mind is still warm with it. To hold simplicity in our example undertaking we'll blink LED whilst interrupt is generated after every 0.5 Sec. We will use PWM in LPC2148 ARM7 Microcontroller to control the brightness of the LED. We can do much stuff by way of the era of the varying duty cycle. Outside interrupt in LPC2148 ARM7 Microcontroller to design sophisticated embedded applications is an Interrupt Service Routine (ISR). As we've got used Timer0 as a supply of interrupt

6.conclusion:

We have to choose Timer Mode then setup the prescale fee in Prescale Register (T0PR) to 59999. The prescale value defines the resolution of Timer0. In this situation, at 6000 clock cycles at 60 MHz. We'll get 1 ms resolution. Then really reset Timer0. An MR0 match event increases an IRQ (Interrupt Request). Because we understand Timer zero is a source of interruption so read present-day cost in Timer0's Interrupt Register i.E. T0IR. We will use PWM in LPC2148 ARM7 Microcontroller to control the brightness of the LED. We can do much stuff by way of the era of the varying duty cycle. Outside interrupt in LPC2148 ARM7 Microcontroller to design sophisticated embedded application. We have to test the price in Timer Counter Register (T0TCR) and wait till T0TC reaches a thousand counts. Which will then generate one thousand ms i.E. 1 sec. Time put off. applications is an Interrupt Service Routine (ISR). As we've got used Timer0 as a supply of interrupt. . To hold simplicity in our example undertaking we'll blink LED whilst interrupt is generated after every 0.5 Sec

7.Reference:

[1] Gallo, Daniele, et al. "An advanced energy/power meter based on ARM microcontroller **fo**r smart grid applications." 17th Symp. IMEKO TC4, 3rd Symp. IMEKO TC19 and 15th IWADC, Kosice, Slovakia. 2010.

[2] Drosos, Christos, M. Zayadine, and Dimitris Metafas. "Real-time communication protocol development using SDL for an embedded system on chip based on ARM microcontroller." Proceedings 13th Euromicro Conference on Real-Time Systems. IEEE, 2001

[3]Chaber, Patryk, and Maciej Ławryńczuk. "Fast analytical model predictive controllers and their implementation for STM32 ARM microcontroller." IEEE Transactions on Industrial Informatics 15.8 (2019): 4580-4590.Z

[4] Bhuyan, Ariful Islam, and Tuton Chandra Mallick. "Gyro-accelerometer based control of a robotic arm using AVR microcontroller." 2014 9th International Forum on Strategic Technology (IFOST). IEEE, 2014.

[5] Olawale, Jegede, et al. "Development of a microcontroller based robotic arm." Proceedings of the 2007 Computer Science and IT Education Conference. 2007.

[6] Cheng, Qian-Qian, et al. "Gamma measurement based on CMOS sensor and ARM microcontroller." Nuclear Science and Techniques 28.9 (2017): 122.

[7]Ismail, Kristian, Aam Muharam, and Mulia Pratama. "Design of CAN bus for research applications purpose hybrid electric vehicle using ARM microcontroller." Energy Procedia 68 (2015): 288-296.

[8] Mateski, Stojan, and Zoran Anastasovski. "Digital sound recorder with ARM microcontroller and SD card." 2012 20th Telecommunications Forum (TELFOR). IEEE, 2012.

[9] Atmadja, Wiedjaja, et al. "Hydroponic system design with real time OS based on ARM Cortex-M microcontroller." IOP Conference Series: Earth and Environmental Science. Vol. 109. No. 1. IOP Publishing, 2017.

[10] Zhao, Xia, et al. "Design of intelligent security system based on arm microcontroller." 2010 International Conference on E-Product E-Service and E-Entertainment. IEEE, 2010.

[11] Toresano, L. O. H. Z., et al. "Data acquisition system of 16-channel EEG based on ATSAM3X8E ARM Cortex-M3 32-bit microcontroller and ADS1299." AIP Conference Proceedings. Vol. 1862. No. 1. AIP Publishing LLC, 2017.

[12] Bejo, Agus, Wanchalerm Pora, and Hiroaki Kunieda. "Development of a 6-axis robotic arm controller implemented on a low-cost microcontroller." 2009 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology. Vol. 1. IEEE, 2009.

[13] Krishna, R., et al. "Design and implementation of a robotic arm based on haptic technology." Int. J. of Eng. Research and Applications 2.34 (2012).

[14] Faravar, Arian. Design, implementation and control of a robotic arm using PIC 16F877A microcontroller. Diss. Eastern Mediterranean University (EMU)-Doğu Akdeniz Üniversitesi (DAÜ), 2014

[15] Gao, Fu-You, and Jun-Yong Zhou. "Design and implementation of fingerprint lock with RF wireless control based on ARM microcontroller." Computer Engineering and Design 31.11 (2010): 2482-2486.

[16] Tresanchez, M. T., et al. "An inexpensive wireless smart camera system for IoT applications based on an ARM CORTEX-M7 microcontroller." Journal of Ubiquitous Systems and Pervasive Networks 11.2 (2019): 1-8

[17] Ismail, B., et al. "Development of a single phase SPWM microcontroller-based inverter." 2006 IEEE International Power and Energy Conference. IEEE, 2006

[18] Xu, R. Z., W. Guo, and C. X. Li. "A new type of embedded autolevelling control system based on arm microcontroller for carding machine." 2006 1ST IEEE Conference on Industrial Electronics and Applications. IEEE, 2006

[19] Ruimei, Zhao, and Wang Mei. "Design of ARM-based embedded Ethernet interface." 2010 2nd International Conference on Computer Engineering and Technology. Vol. 4. IEEE, 2010.

[20] Wenkai, Chen, Zhang Genbao, and Zhang Zhenqiang. "An Intelligent Detecting System of Material Location of Heavy Hammer Type Based on the Microcontroller with ARM Cortex-M3 Kernel [J]." Computer Measurement & Control 11 (2008)

[21] Dakhole, Ashwini Y., and Mrunalini P. Moon. "Design of intelligent traffic control system based on ARM." International journal of advance research in computer science and management studies 1.6 (2013).

[22] Jha, Hare Ram, Akash Priyadarshi, and Anamika Kumari. "Electronic module of hydraulic damper test bench using ARM microcontroller interfacing in labview." International Journal of Scientific & Engineering Research 4.1 (2013).

[23] Shaikh, Samrin, and Shashank Pujari. "Migration from microcontroller to FPGA based SoPC design: Case study: LMS adaptive filter design on Xilinx Zynq FPGA with embedded ARM controller." 2016

International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT). IEEE, 2016.

[24] Hiren, Patel, and Patel Dipak. "Gui based data acquisition system using arm-cortex m3 microcontroller." IJCSIT) International Journal of Computer Scienceand Information Technologies 3.1 (2012): 3199-3204

[25] Khajone, Saurabh A., S. W. Mohod, and V. M. Harne. "Implementation of a wireless gesture controlled robotic arm." International Journal of innovative research in computer and communication engineering 3.1 (2015): 377-379.

[26] Roshanna, Lakshmi Narayana, et al. "Development of Ethernet based remote monitoring and controlling of MST radar transmitters using ARM cortex microcontroller." Sensors & Transducers 148.1 (2013): 40.

[27] Hsiung, Steve C., et al. "Collaborated Efforts in TI ARM M4/32Bits Microcontroller Curricula Developments and Assessments." (2018).

[28] Soares, Marshall, and Alireza R. Behbahani. "A 0.15 um SOI High Temperature ARM Microcontroller for Local Control Nodes: Status of the Next Step." 2018 Joint Propulsion Conference. 2018

[29] Dolinay, Jan, Petr Dostálek, and Vladimír Vašek. "ARM-based Microcontroller Platform for Teaching Microcontroller Programming." International Journal of Education and Information Technologies

[30] Ruimei, Zhao, and Wang Mei. "Design of ARM-based embedded Ethernet interface." *2010 2nd International Conference on Computer Engineering and Technology*. Vol. 4. IEEE, 2010.

[31] Mateski, Stojan, and Zoran Anastasovski. "Digital sound recorder with ARM microcontroller and SD card." 2012 20th Telecommunications Forum (TELFOR). IEEE, 2012.

[32] Atmadja, Wiedjaja, et al. "Hydroponic system design with real time OS based on ARM Cortex-M microcontroller." IOP Conference Series: Earth and Environmental Science. Vol. 109. No. 1. IOP Publishing, 2017.