Unit -3 8051 Real time controls

In this article, we focus on Timers/Counters of the 8051 micro controller. The 8051 has two counters/timers which can be used either as timer to generate a time delay or as counter to count even.

In Intel 8051, there are two 16-bit timer registers. These registers are known as Timer 0 and Timer 1. The timer registers can be used in two modes. These modes are Timer mode and the Counter mode. The only difference between these two modes is the source for incrementing the timer registers.

Timer Mode

In the timer mode, the internal machine cycles are counted. So this register is incremented in each machine cycle. So when the clock frequency is 12MHz, then the timer register is incremented in each millisecond. In this mode it ignores the external timer input pin.

Counter Mode

In the counter mode, the external events are counted. In this mode, the timer register is incremented for each 1 to 0 transition of the external input pin. This type of transitions is treated as events. The external input pins are sampled once in each machine cycle, and to determine the 1or 0 transitions, another machine cycle will be needed. So in this mode, at least two machine cycles are needed. When the frequency is12MHz, then the maximum count frequency will be 12MHz/24 = 500 KHz. So for event counting the time duration is 2 µs.

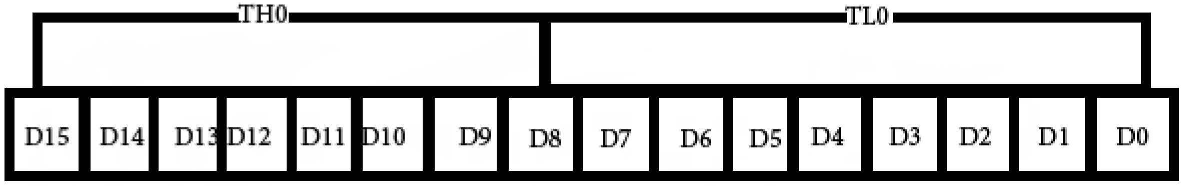
There are four different modes of the Timer or Counter. The Mode 0 to Mode 2 are for both of the Timer/Counter. Mode 3 has a different meaning for each timer register. There is a register called TMOD. This register can be programmed to configure these timers or counters.

The Serial port is used for serial communication in mode 1 and 3. Timer1 is used for generating the baud rate. So only Timer 0 is available for timer or counter operations.

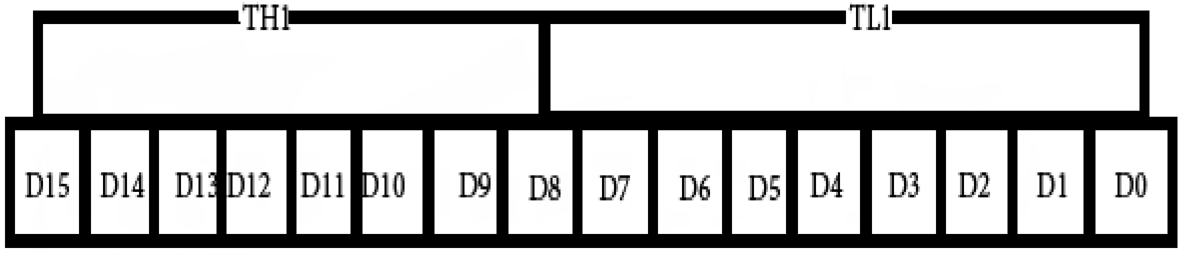
Counting /timing device as timer a device for timing when the inputs to counting are given by a clock. The clock pulses are internally given at the specific time intervals in case of functioning as timer.

The 8051 has two timers: timer0 and timer1. They can be used either as timers or as counters. Both timers are 16 bits wide. Since the 8051 has an 8-bit architecture, each 16-bit is accessed as two separate registers of low byte and high byte. First we shall discuss about Timer0 registers. it's happening outside the microcontroller.

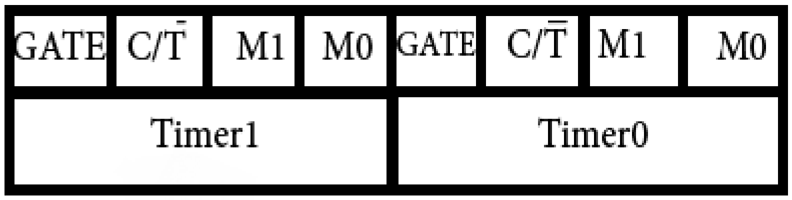
Timer 0 registers is a 16 bits register and accessed as low byte and high byte. The low byte is referred as a TL0 and the high byte is referred as TH0. These registers can be accessed like any other registers.



Timer 1 registers is also a 16 bits register and is split into two bytes, referred to as TL1 and TH1.



TMOD (timer mode) Register: This is an 8-bit register which is used by both timers 0 and 1 to set the various timer modes. In this TMOD register, lower 4 bits are set aside for timer0 and the upper 4 bits are set aside for timer1. In each case, the lower 2 bits are used to set the timer mode and upper 2 bits to specify the operation.



In upper or lower 4 bits, first bit is a GATE bit. Every timer has a means of starting and stopping. Some timers do this by software, some by hardware, and some have both software and hardware controls. The hardware way of starting and stopping the timer by an external source is achieved by making GATE=1 in the TMOD register. And if we change to GATE=0 then we do no need external hardware to start and stop the timers. The second bit is C/T bit and is used to decide whether a timer is used as a time delay generator or an event counter. If this bit is 0 then it is used as a timer and if it is 1 then it is used as a counter. In upper or lower 4 bits, the last bits third and fourth are known as M1 and M0 respectively. These are used to select the timer mode.

Interrupt structure of 8051-

Now in this section, we will see the interrupt structure of Intel 8051 microcontroller.

Interrupts are basically the events that temporarily suspend the main program, pass the control to the external sources and execute their task. It then passes the control to the main program where it had left off.

Interrupts in 8051 microcontroller are more desirable to reduce the regular status checking of the interfaced devices or inbuilt devices. Interrupt is an event that temporarily suspends the main program, passes the control to a special code section, executes the event-related function and resumes the main program flow where it had left off. Interrupts are of different types like software and hardware, maskable and non-maskable, fixed and vector interrupts, and so on. Interrupt Service Routine (ISR) comes into the picture when interrupt occurs, and then tells the processor to take appropriate action for the interrupt, and after ISR execution, the controller jumps into the main program.

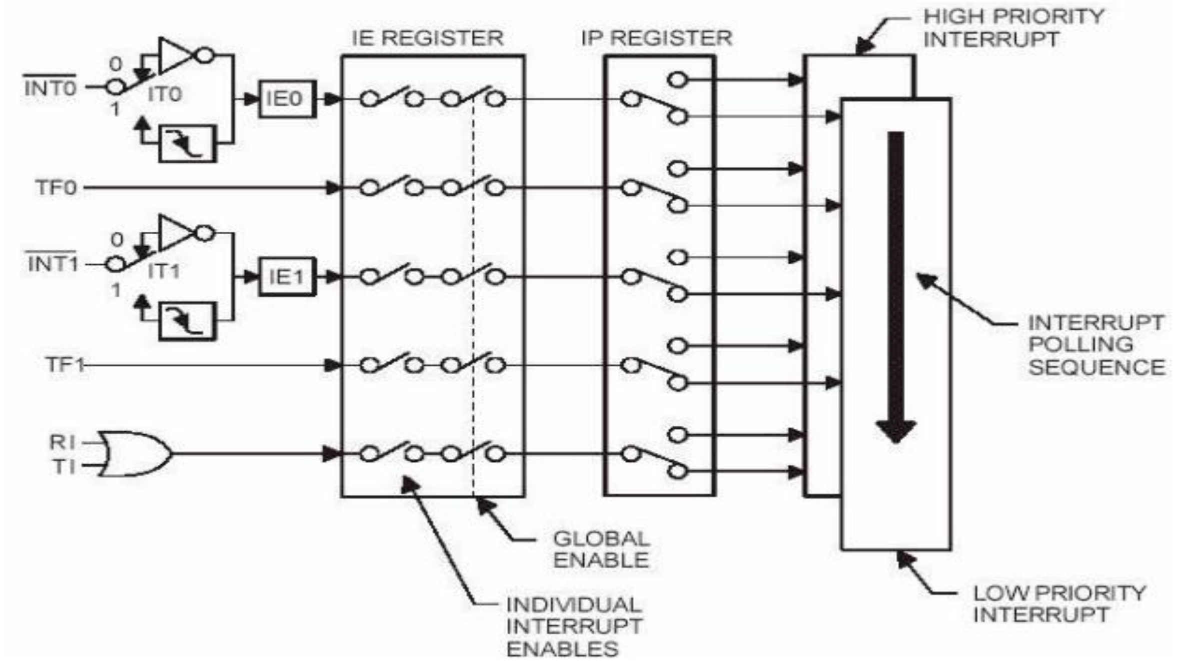
8051 has five interrupts. These interrupts are INT0, INT1, TO, T1, TI/RI. All of the interrupts can be enabled or disabled by using the IE (interrupt enable) register.

The interrupt addresses of these interrupts are like below −

Interrupt	Address
INT0	0003H
INT1	000BH
T0	0013H
T1	001BH
TI/RI	0023H

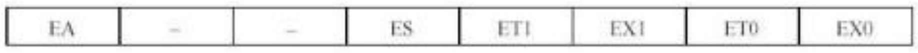
Interrupt Structure of 8051 Micro controller

Upon 'RESET' all the interrupts get disabled, and therefore, all these interrupts must be enabled by software. In all these five interrupts, if anyone or all are activated, this sets the corresponding interrupt flags as shown in the figure. All these interrupts can be set or cleared by bit in some special function register that is Interrupt Enabled (IE), and this in turn depends on the priority, which is executed by IP interrupt priority register.



Interrupt Enable (IE) Register:

This register is responsible for enabling and disabling the interrupt. It is a bit addressable register in which EA must be set to one for enabling interrupts. The corresponding bit in this register enables particular interrupt like timer, external and serial SNSCT 16EC008-MPMCA/UNIT-1 GAUTAMI.A,AP/ECE inputs. In the below IE register, bit corresponding to 1 activates the interrupt and 0 disables the interrupt.

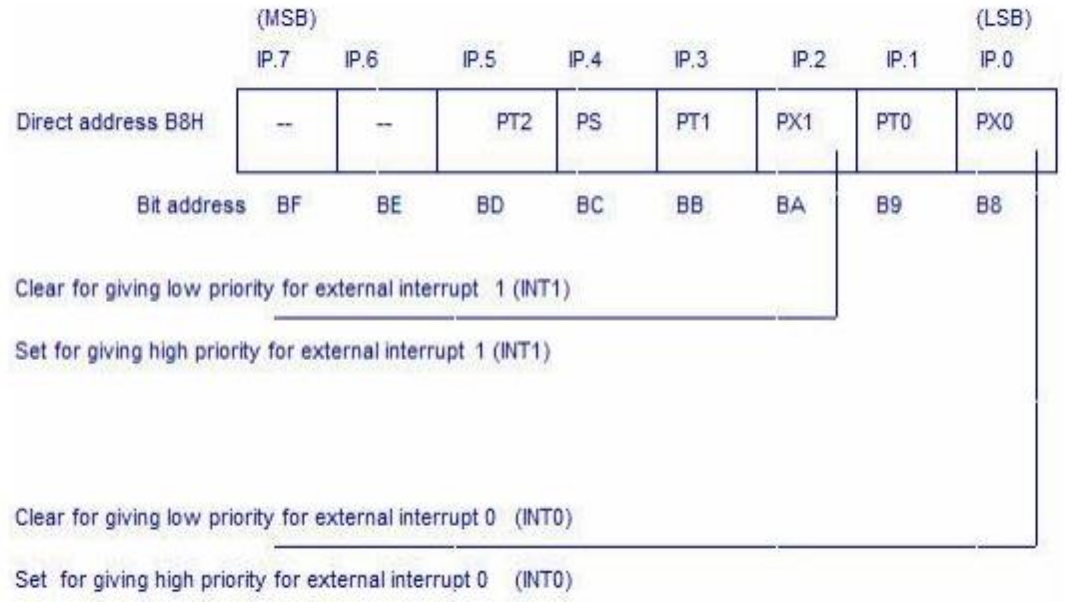


EA	-	-	ES	ET1	EX1	ET0	EX0

EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, interrupt source is individually enable or disabled by setting or clearing its enable bit.
-	IE.6	Not implemented, reserved for future use*.
-	IE.5	Not implemented, reserved for future use*.
ES	IE.4	Enable or disable the Serial port interrupt.
ET1	IE.3	Enable or disable the Timer 1 overflow interrupt.
EX1	IE.2	Enable or disable External interrupt 1.
ET0	IE.1	Enable or disable the Timer 0 overflow interrupt.
EX0	IE.0	Enable or disable External Interrupt 0.

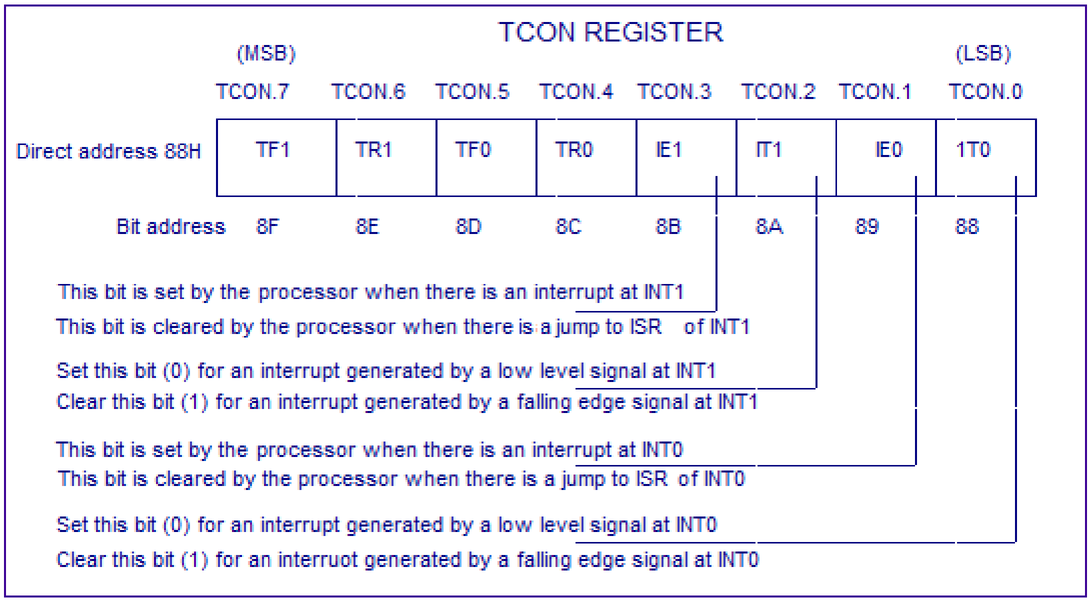
Interrupt Priority Register (IP):

It is also possible to change the priority levels of the interrupts by setting or clearing the corresponding bit in the Interrupt priority (IP) register as shown in the figure. This allows the low priority interrupt to interrupt the high-priority interrupt, but prohibits the interruption by another low-priority interrupt. Similarly, the high-priority interrupt cannot be interrupted. If these interrupt priorities are not programmed, the microcontroller executes in predefined manner and its order is INT0, TF0, INT1, TF1, and SI.


(MSB)
(LSB)
IP.7
IP.6
IP.5
IP.4
IP.3
IP.2
IP.1
IP.0
Direct address B8H
--
--
PT2
PS
PT1
PX1
PT0
PX0
Bit address
BF
BE
BD
BC
BB
BA
B9
B8
Clear for giving low priority for external interrupt 1 (INT1)
Set for giving high priority for external interrupt 1 (INT1)
Clear for giving low priority for external interrupt 0 (INT0)
Set for giving high priority for external interrupt 0 (INT0)

TCON Register:

In addition to the above two registers, the TCON register specifies the type of external interrupt to the 8051 microcontroller, as shown in the figure. The two external interrupts, whether edge or level triggered, specify by this register by a set, or cleared by appropriate bits in it. And, it is also a bit addressable register.



TCON REGISTER

	(MSB) TCON.7	TCON.6	TCON.5	TCON.4	TCON.3	TCON.2	TCON.1	(LSB) TCON.0
Direct address 88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	1T0
Bit address	8F	8E	8D	8C	8B	8A	89	88

This bit is set by the processor when there is an interrupt at INT1
This bit is cleared by the processor when there is a jump to ISR of INT1

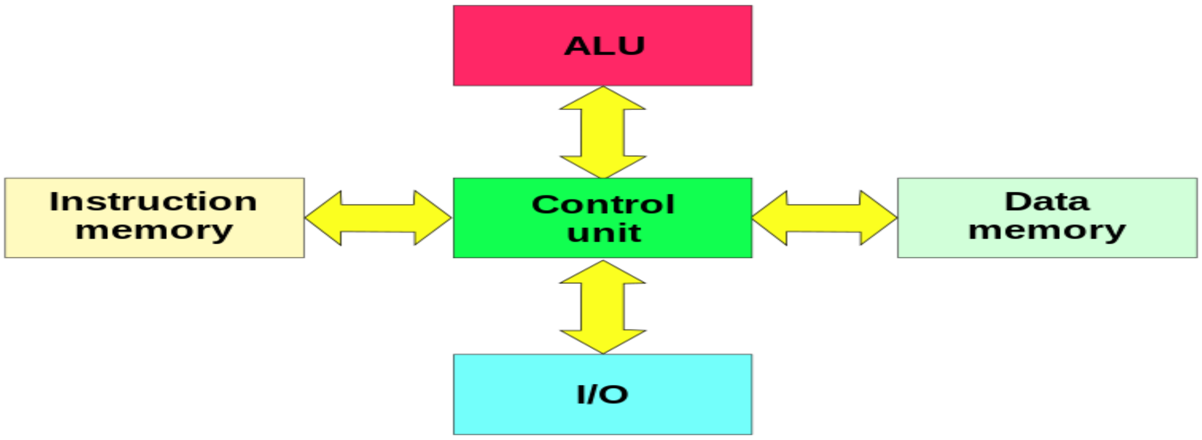
Set this bit (0) for an interrupt generated by a low level signal at INT1
Clear this bit (1) for an interrupt generated by a falling edge signal at INT1

This bit is set by the processor when there is an interrupt at INT0
This bit is cleared by the processor when there is a jump to ISR of INT0

Set this bit (0) for an interrupt generated by a low level signal at INT0
Clear this bit (1) for an interruot generated by a falling edge signal at INT0

PIC Microcontroller

The PIC microcontroller was introduced in 1993 by Microchip although the original chip design was created by General Instruments in 1985. PIC microcontrollers are meant to enable simple programming and interfacing in embedded system design. Most of the PIC microcontrollers that hit the market are 8-bits microcontrollers, although Microchip did introduce some 16-bits and 32-bits PIC microcontrollers.



PIC microcontrollers are based on the Harvard architecture where program and data busses are kept separate. Early versions of PIC microcontrollers use EPROM to store the program instruction but have adopted the flash memory since 2002 to allow better erasing and storing of the code.

Thanks to its simplicity in architecture and ease-of-use, PIC microcontrollers have proved to be a hit amongst hobbyists, students, and professionals. The PIC16F84 and PIC16F877 were some of the most popular PIC microcontrollers with basic functionalities. Applications that require richer peripherals, higher performance or memory can rely on the PIC18F family.

PIC Microcontroller Features

One of the reasons why I'm such a big fan of PIC microcontrollers is that they often have similar features. This makes working with various part numbers of PIC microcontrollers easy as the architecture, peripherals, and design cycle are almost the same.

Basic PIC microcontrollers may have limited peripherals while their more advanced counterparts are rich with communications, memory, I/Os, and special function peripherals. However, you can expect the PIC microcontrollers generally have these features.

Memory

Modern PIC microcontrollers are built with Flash memory to store program instructions. Flash memory has a larger capacity and is easily-erased compared to EPROM or OTP microcontrollers.

PIC microcontrollers also feature built-in RAM and EEPROM that are useful in storing application parameters and run-time variables. You'll also find that the special function registers and stack pointers of the PIC microcontrollers are quite standard across various part numbers.

Oscillator

Like other microcontrollers, PIC needs a clocking source to drive its system timing. This is often done with a crystal or in some PICs, an internal oscillator.

I/O

PIC microcontrollers labeled segregate I/O pins by ports (e.g. PORTA, PORTB). For an 8-bit PIC, a single port contains 8 individual pins. Each pin can be configured as input, output or alternative peripheral functions.

Peripherals

In most PIC microcontrollers, you'll also find the following peripherals.

- EUSART - provides serial UART communication.
- SSP - allows SPI and I2C interface.
- CCP - capture and compare module and PWM output.
- ADC - convents analog signal to digital values.
- Timers - Depending on the part number, there could be several times in the PIC.

Designing With A PIC Microcontroller

It is fair to suggest that Microchip's dominant market share is due to its comprehensive development support that reduces development cost. For example, the MPLAB IDE, which is used to create program

code for PIC microcontrollers, has always been downloadable for free. Programming tools, such as the PICKIT, are more affordable compared to other manufacturers.
PIC microcontrollers have proved to be popular amongst hobbyists with their affordability and ease-of-use. With that said, PIC microcontrollers are also viable solutions for commercial and industrial applications. It is easy to kickstart a PCB project with the PIC.

PCB designers could bank on its rich knowledgebase where datasheets, user guides, and various application notes are available. In recent years, various startup kits with popular PIC microcontrollers are made available. With the right PCB design software, you'll also have access to readily-available libraries of PIC microcontrollers. OrCAD can provide you with the collaborative layout and design tool that you need to get any device made easily and with the utmost care.

Arm processor

An Arm processor is one of a family of central processing units (CPUs) based on the reduced instruction set computer (RISC) architecture for computer processors. Arm Limited, the company behind the Arm processor, designs the core CPU components and licenses the intellectual property to partner organizations, which then build Arm-based chips according to their own requirements. Arm Limited does not manufacture or sell any chips directly.

Arm Limited offers designs for both 32-bit and 64-bit RISC multicore processors. The processors use a much simpler instruction set than their Intel counterparts, which are based on the complex instruction set computing (CISC) architecture. The two types of processors also employ different methods to optimize performance and increase efficiency. For example, Intel takes a hardware approach to maximizing performance, whereas Arm takes a software approach.

Arm processors can execute many more millions of instructions per second than Intel processors. By stripping out unneeded instructions and optimizing pathways, an Arm processor can deliver outstanding performance while using much less energy than a CISC-based processor. The reduction in power also means that Arm CPUs generate less heat. That's not to say Arm processors are inherently better than Intel processors, only that they're better suited to specific use cases.

Arm processor features

Because of their reduced instruction set, Arm processors require fewer transistors, resulting in a smaller die size for the integrated circuitry. Their smaller size, reduced complexity and lower power consumption make them suitable for increasingly miniaturized devices.

Arm processor features include the following:

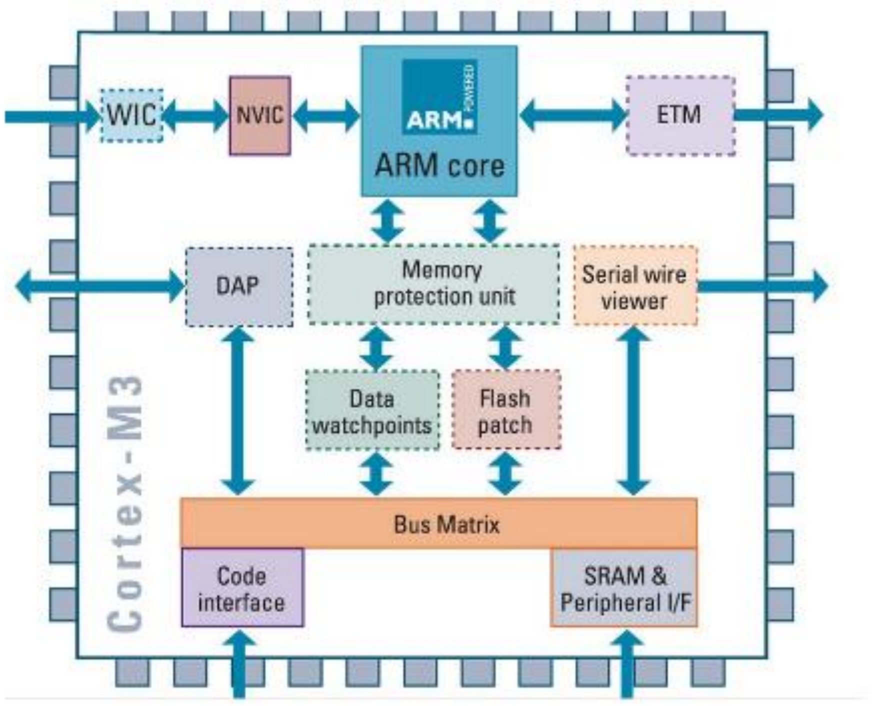
- load/store architecture
- integrated security
- orthogonal instruction set
- single-cycle execution

- energy efficiency
- 64- and 32-bit execution states
- hardware virtualization support

The simplified design of Arm processors offers more efficient multicore processing and easier coding for developers. While they don't offer the same raw compute throughput as Intel CPUs, Arm processors sometimes exceed the performance of Intel processors for applications that exist on both architectures.

ARM Architecture

The ARM architecture processor is an advanced reduced instruction set computing [RISC] machine and it's a 32bit reduced instruction set computer (RISC) microcontroller. It was introduced by the Acron computer organization in 1987. This ARM is a family of microcontroller developed by makers like ST Microelectronics, Motorola, and so on. The ARM architecture comes with totally different versions like ARMv1, ARMv2, etc., and, each one has its own advantage and disadvantages.



The ARM cortex is a complicated microcontroller within the ARM family that has ARMv7 design. There are 3 subfamilies within the ARM cortex family :

- ARM Cortex Ax-series
- ARM-Cortex Rx-series
- ARM-Cortex Mx-series

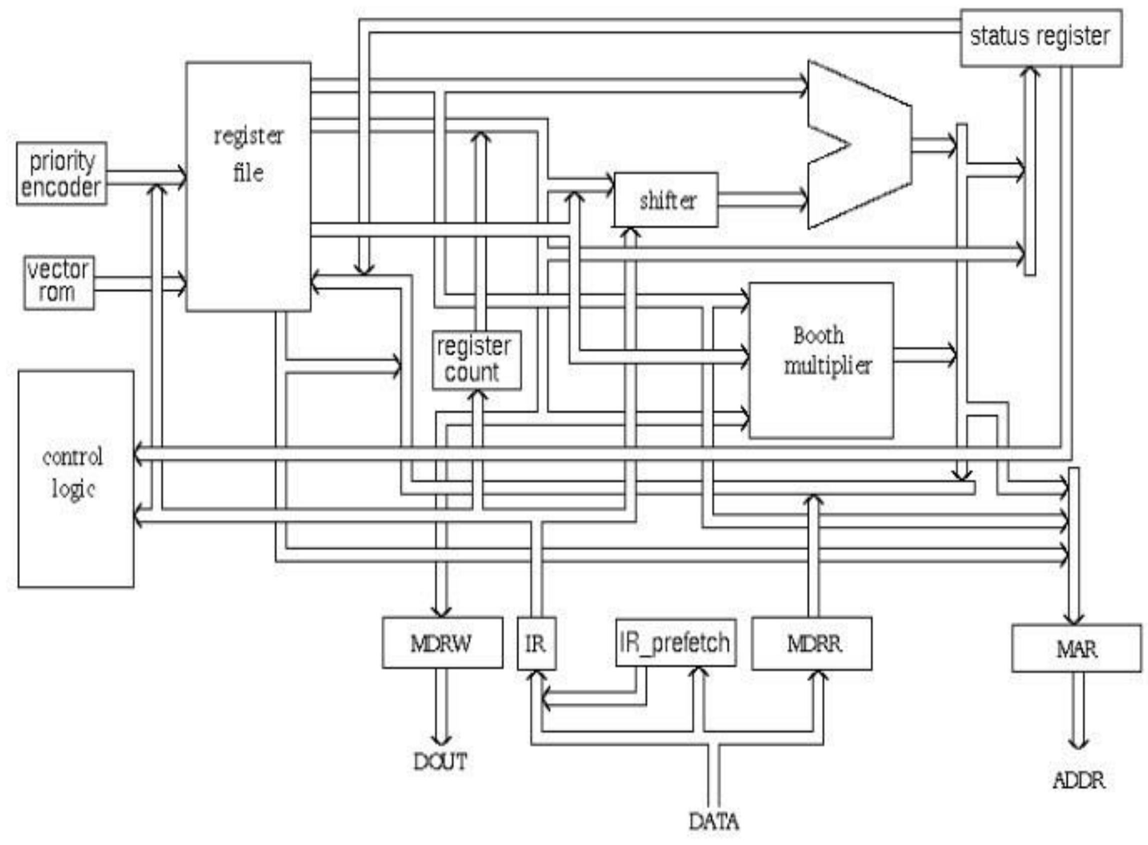
The ARM Architecture

- Arithmetic Logic Unit
- Booth multiplier
- Barrel shifter
- Control unit
- Register file

This article covers the below mentioned components.

The ARM processor conjointly has other components like the Program status register, which contains the processor flags (Z, S, V and C). The modes bits conjointly exist within the program standing register, in addition to the interrupt and quick interrupt disable bits; some special registers: Some registers are us Priority encoder: The encoder is used in the multiple load and store instruction to point which register within the register file to be loaded or kept .

Multiplexers: several multiplexers are accustomed to the management operation of the processor buses. Because of the restricted project time, we tend to implement these components in a very behavioral model. Each component is described with an entity. Every entity has its own architecture, which can be optimized for certain necessities depending on its application. This creates the design easier to construct and maintain.



Arithmetic Logic Unit (ALU)

The ALU has two 32-bits inputs. The primary comes from the register file, whereas the other comes from the shifter. Status registers flags modified by the ALU outputs. The V-bit output goes to the V flag as well as the Count goes to the C flag. Whereas the foremost significant bit really represents the S flag, the ALU output operation is done by NORed to get the Z flag. The ALU has a 4-bit function bus that permits up to 16 opcode to be implemented.

Booth Multiplier Factor

The multiplier factor has 3 32-bit inputs and the inputs return from the register file. The multiplier output is barely 32-Least Significant Bits of the merchandise. The entity representation of the multiplier factor is shown in the above block diagram. The multiplication starts whenever the beginning 04 input goes active. Fin of the output goes high when finishing.

Booth Algorithm

Booth algorithm is a noteworthy multiplication algorithmic rule for 2's complement numbers. This treats positive and negative numbers uniformly. Moreover, the runs of 0's or 1's within the multiplier factor are skipped over without any addition or subtraction being performed, thereby creating possible quicker multiplication. The figure shows the simulation results for the multiplier test bench. It's clear that the multiplication finishes only in16 clock cycle.

Barrel Shifter

The barrel shifter features a 32-bit input to be shifted. This input is coming back from the register file or it might be immediate data. The shifter has different control inputs coming back from the instruction register. The Shift field within the instruction controls the operation of the barrel shifter. This field indicates the kind of shift to be performed (logical left or right, arithmetic right or rotate right). The quantity by which the register ought to be shifted is contained in an immediate field within the instruction or it might be the lower 6 bits of a register within the register file.

The shift Val input bus is 6-bits, permitting up to 32 bit shift. The shift type indicates the needed shift sort of 00, 01, 10, 11 are corresponding to shift left, shift right, an arithmetic shift right and rotate right, respectively. The barrel shifter is especially created with multiplexers.

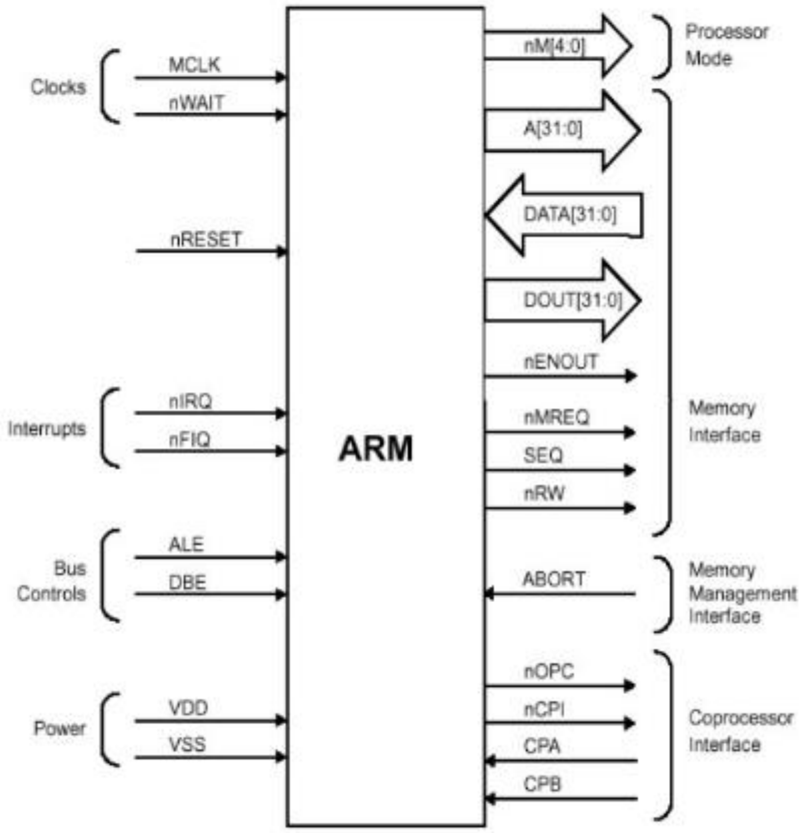
Control Unit

For any microprocessor, control unit is the heart of the whole process and it is responsible for the system operation, so the control unit design is the most important part within the whole design. The control unit is sometimes a pure combinational circuit design. Here, the control unit is implemented by easy state machine. The processor timing is additionally included within the control unit. Signals from the control unit are connected to each component within the processor to supervise its operation.

Please refer to this link to know more about ARM MCQs

ARM7 Functional Diagram

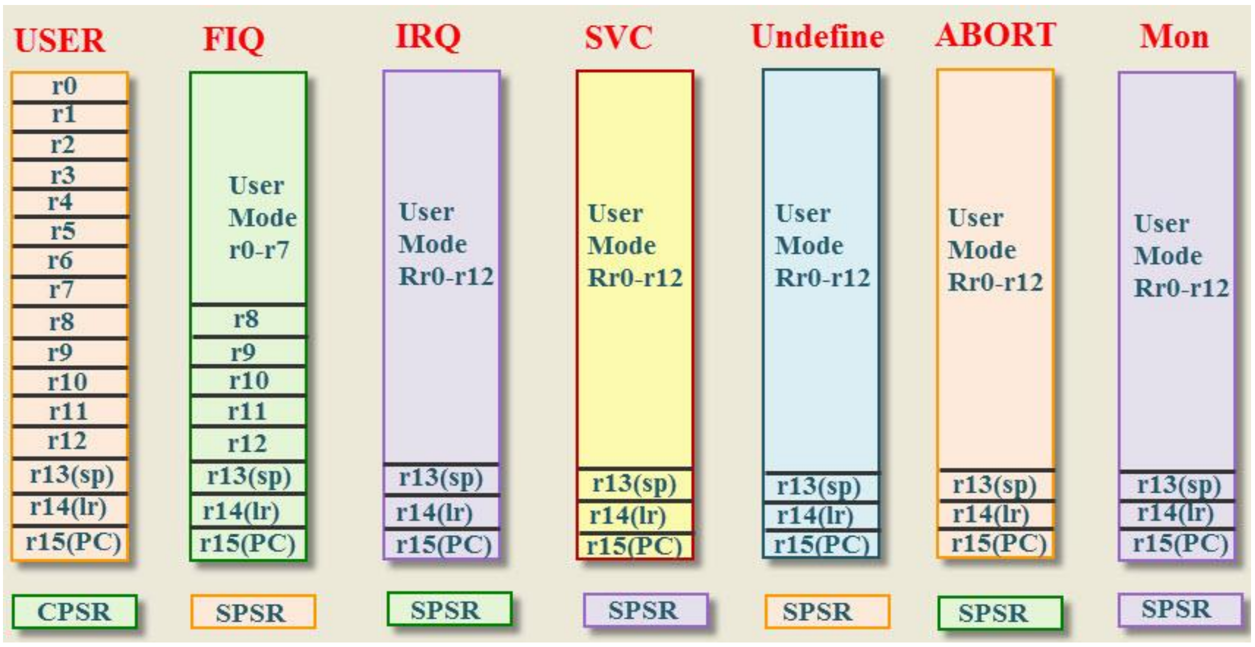
The final thing that must be explained is how the ARM will be used and the way in which the chip appear. The various signals that interface with the processor are input, output or supervisory signals which will be used to control the ARM operation.



ARM Microcontroller Register Modes

An ARM microcontroller is a load store reducing instruction set computer architecture means the core cannot directly operate with the memory. The data operations must be done by the registers and the information is stored in the memory by an address. The ARM cortex-M3 consists of 37 register sets wherein 31 are general purpose registers and 6 are status registers. The ARM uses seven processing modes to run the user task.

- USER Mode
- FIQ Mode
- IRQ Mode
- SVC Mode
- UNDEFINED Mode
- ABORT Mode
- Monitor Mode



USER Mode: The user mode is a normal mode, which has the least number of registers. It doesn't have SPSR and has limited access to the CPSR.

FIQ and IRQ: The FIQ and IRQ are the two interrupt caused modes of the CPU. The FIQ is processing interrupt and IRQ is standard interrupt. The FIQ mode has additional five banked registers to provide more flexibility and high performance when critical interrupts are handled.

SVC Mode: The Supervisor mode is the software interrupt mode of the processor to start up or reset.

Undefined Mode: The Undefined mode traps when illegal instructions are executed. The ARM core consists of 32-bit data bus and faster data flow.

THUMB Mode: In THUMB mode 32-bit data is divided into 16-bits and increases the processing speed.

THUMB-2 Mode: In THUMB-2 mode the instructions can be either 16-bit or 32-bit and it increases the performance of the ARM cortex –M3 microcontroller. The ARM cortex-m3 microcontroller uses only THUMB-2 instructions.

Some of the registers are reserved in each mode for the specific use of the core. The reserved registers are

- Stack Pointer (SP).
- Link Register (LR).
- Program Counter (PC).
- Current Program Status Register (CPSR).
- Saved Program Status Register (SPSR).

The reserved registers are used for specific functions. The SPSR and CPSR contain the status control bits which are used to store the temporary data. The SPSR and CPSR register have some properties that are defined operating modes, Interrupt enable or disable flags and ALU status flag. The ARM core operates in two states 32-bit state or THUMBS state.