1. Understanding embedded hardware

To become a good embedded software developer, first one should be a good C programmer. Next one should have a decent understanding of hardware used in an embedded system.

Every embedded system is a microprocessor or microcontroller based device. The microcontroller hardware and software are controlling the device. Take an example of electronic washing machine. The microcontroller is controlling the operation of washine machine. The software controls the water inlet valve to open. When water reaches to the required level, it closes the valve. Next it starts the motor, and starts a timer. When timer expires it stops the motor and opens the outlet valve. So that water will drain out. This is a simple description. But everything is handled by software and hardware.

There is not much difference between the hardware used in an embedded device and the hardware used in side a computer. Both will contain CPU(microprocessor), memory chips, IO chips and IO devices. They differ only in their size and procesing speed. It is like automobiles. The working of all automobiles are same, more or less all will have the same parts. Automobiles differ only by their size and power.

Every hardware, either used in computers, or embedded system contains the following 4 components:

- CPU
- Memory Chips (Both volatile and Non-volatile)
- IO or Peripheral Chips
- IO or Peripheral devices

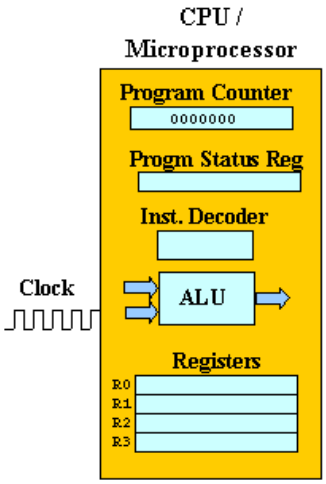
In the following topics, we are describing the functioning of a very simple hardware. This is a simple hardware, designed for study purpose. One should be able to understand this simple hardware easily. Once understand this simple hardware, one can understand any hardware easily.

- Understanding CPU
- Understanding CPU bus interface
- Interfacing ROM/Flash to CPU bus
- Interfacing ROM and RAM to CPU bus
- Understanding IO and Interfacing Digital IO chip with the CPU
- Understanding IO communication and Interfacing UART controller to CPU
- Understanding and Interfacing Counter/Timer chip to CPU
- Understanding how to load a program into ROM/RAM and run the program

2. Understanding CPU

The most important item in the computing hardware is the Central Processing Unit (CPU). CPU is also refered as Microprocessor. The main job of CPU is to fetch (i.e. read) instructions one by one from memory, decode them and execute them. Fetch, decodoe, execute; fetch, decode, execute; ... and this goes on forever.

Program Counter (PC)



Computing Hardware : CPU

Look at the CPU shown above. The Program Counter (PC) in the CPU is a special register. It holds the address of the next instruction, which it is going to fetch,decode and execute. This address will be the address of a memory location, from which it fetches the instruction. This memory can be either RAM or ROM. In some CPUs, program counter is also refered as Instruction Pointer (IP).

Once power is applied to the CPU, it goes through a reset process and all its registers will be initialized to some default values. In this reset process, the PC will get initialized to zero. So CPU fetches its first instruction from the memory address zero. After fetching the instruction, PC will automatically incremented to the address of next instruction.

Instruction Decoder and Instruction format

The instruction fetched from the memory is decoded by the instruction decoder. This decoder is shown in the diagram. Note that each instruction is a number occupying one or more bytes. We should view these instructions as 8 bit or 16 bit (or any multiple of 8 bit) numbers. Some bits in this number specify what operation to perform. Other bits specify on which this operation to perform. For example if operation bits specify ADD operation, other bits specify the register numbers. The bits in the instruction that specify operation is called opcode bits. Other bits which specify the registers is called operand bits. So instruction bits consist of opcode bits and operand bits.

The decoder unit inside the CPU, is responsible for decoding the opcode bits and operand bits. From decoding, it finds out the operation to perform and operands to be used. The execution unit takes this decoded information and performs the operation with the help Arithmetic and Logic Unit (ALU).

Most of the time, operands specified in the instructions are registers. So operations are performed on these register contents. Results are stored back in the registers. But some times, operand may be immediate value present in the operand bits. Such operands are called immediate operands.

In the CPU shown, it has got only four registers named R0 to R3. So to specify register operand, only two bits are sufficient. Following section describes the simple instruction format supported by this simple CPU.

Instruction Set Architecture (ISA) of CPU

Every CPU is designed to execute only a set of instructions called machine instructions. One can estimate the capability of a CPU by looking at the instruction set of CPU. Following is the instruction set of our simple CPU for study purpose. But note that this instruction set is very minimal when compared with any real CPU. However these simple instructions helps a lot in understanding the structure of a machine instruction.

This simple CPU has got just 4 registers R0, R1, R2 and R3. As this CPU is an 8 bit CPU, all these four register are 8 bit registers. The CPU has got program counter. Program counter also 8 bit.

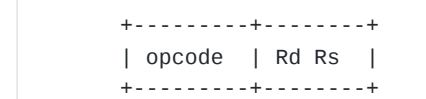
```
Instruction set of simple CPU for learning
```

MNEMONICS	OPERANDS	OPERATION
ADD	Rd, Rs	Rd <- Rd + Rs
SUB	Rd, Rs	Rd <- Rd - Rs
AND	Rd, Rs	Rd <- Rd & Rs
OR	Rd, Rs	Rd <- Rd | Rs
XOR	Rd, Rs	Rd <- Rd ^ Rs
CMP	Rd, Rs	Rd - Rs
JPR	K4	PC <- PC + K4 + 1
JPI	Rd	PC <- Rd
INC	Rd	Rd <- Rd + 1
DEC	Rd	Rd <- Rd - 1
CPL	Rd	Rd <- ~Rd
LD	Rd, [Rs]	Rd <- [Rs]
MOV	Rd, Rs	Rd <- Rs
ST	[Rd], Rs	[Rd] <- Rs
JZ	k4	if(Z == 1) then PC = PC + K4 + 1
JNZ	k4	if(Z == 0) then PC = PC + K4 + 1
JC	k4	if(C == 1) then PC = PC + K4 + 1
JNC	k4	if(C == 0) then PC = PC + K4 + 1
SLP		Goes to Sleep
LDI	Rd, k8	Rd <- k8

Rd : Destination register, where the result of operation is stored
Rs : Source register
k4 : 4 bit relative jump address in two’s compliment form.
k8 : 8 bit Immediate value.

Instruction Encoding format

All the above instructions except last instruction (LDI) is encoded into a single byte. The format of this byte is as follows:



opcode = 0000 -> ADD Rd, Rs
opcode = 0001 -> SUB Rd, Rs

```
 opcode = 0010 -> AND Rd, Rs
 opcode = 0011 -> OR  Rd, Rs
 opcode = 0100 -> XOR Rd, Rs
 opcode = 0101 -> CMP Rd, Rs
 opcode = 1001 -> MOV Rd, Rs

 opcode = 1000 -> LD  Rd, [Rs]
 opcode = 1010 -> ST  [Rd], Rs
```

```
+---------+--------+
| opcode  | Rd op  |
+---------+--------+
```

```
 opcode = 0111, op = 00 -> JPI Rd
 opcode = 0111, op = 01 -> INC Rd
 opcode = 0111, op = 10 -> DEC Rd
 opcode = 0111, op = 11 -> CPL Rd
```

```
+---------+--------+
| opcode  |  k4    |
+---------+--------+
```

```
 opcode = 0110 -> JPR k4
 opcode = 1011 -> JZ  k4
 opcode = 1100 -> JNZ k4
 opcode = 1101 -> JC  k4
 opcode = 1110 -> JNC k4
```

```
+---------+--------+
| opcode  | 00 op  |
+---------+--------+
```

```
 opcode = 1111, op = 11 -> SLP
```

```
+---------+--------+
| opcode  | Rd op  |
+---------+--------+
|        k8        |
+---------+--------+
```

```
 opcode = 1111, op = 00 -> LDI Rd, k8
```

Status Register

The CPU also have another special register called Program Status Register or simply Status Register (SR). The bits in this register are will get updated whenever arithmetic or logic instruction is executed. For example when SUB(subtract) instruction is executed, if result is zero, the zero bit in the status register will get set. If result is negative, the carry bit will get set. These status register bits are used by the conditional instructions like 'JZ'(Jump on Zero) and 'JNZ'(Jump on No Zero). While CPU executing 'JZ' instruction it checks if zero bit is set or not. If set the instruction causes a jump, else not. Note that to cause a jump, the jump instruction simply stores the new address to PC register.

General Purpose registers

The registers R0 to R3 are called, general purpose registers. These registers acts as operands for the operations. Most of the modern CPUs perform operations only on the registers. There are instructions called Load and Store, which will move memory contents to registes and vice versa. So typical programs uses following kinds of instructions.

- Instructions to move data from memory to registers
- Instructions to perform arithmetic and logic operations on the registers
- Instructions to move registers to memory.

8 Bit / 16 Bit / 32 Bit / 64 Bit CPU

We classify CPUs as 8 bit or 16 bit and so on. What desides the number of bits of a CPU? The arithmetic and logic unit (ALU) desides this. If ALU is capable of taking only two 8 bits numbers at a time and producing the output, then it is 8 bit CPU only. In 32 bit CPU, the ALU is capable of performing operation on two 32 bit numbers at a time. Note that inputs to ALU is coming from registers and output of ALU also goes back to register. So the ALU size and register size will be same. So we can say a processor is 64 bit, if its general purpose registers are 64 bit width. So register width gives the CPU bit capability.

Clock

CPU to perform fetch, decode and execution operations continuously, it must require some timing reference. The clock input to the CPU will provide such timing source. CPU on receiving first clock, it fetches the instruction from memory. On the second clock, it decodes the instruction. On receiving third clock it executes the instruction. With the fourth clock, it again fetches the next instruction and it goes on forever. However by using multiple pipeline stages, modern CPUs will perform all three steps in every clock cycle.

Understand that the clock speed decides the execution speed of the CPU. CPUs will support clocks of certain frequency range.

Questions

Answer the following questions to check the understanding of the subject

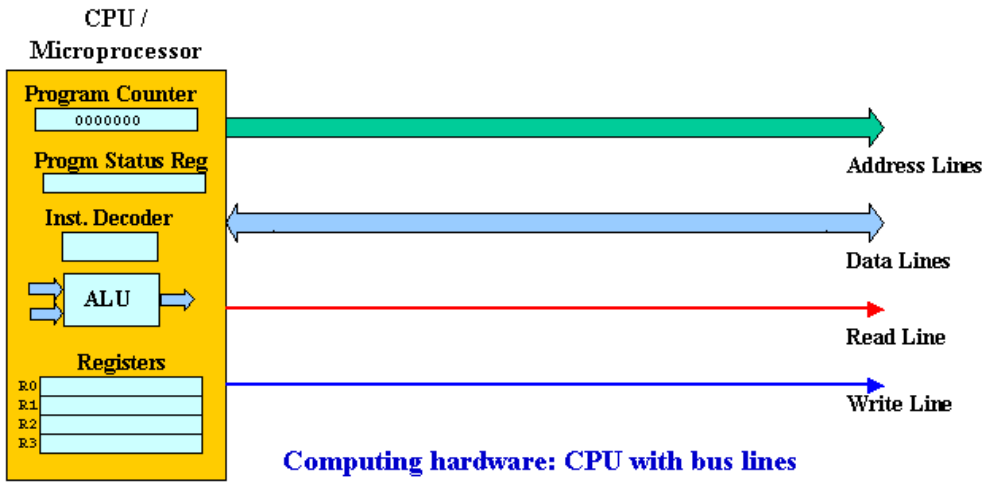
1. What is the purpose of Program Counter (PC) register in a CPU?
2. What is the use of Program Status Register in a CPU?
3. Explain the format of instruction, by explaining opcode bits and operand bits present in the instruction.
4. Write the instruction number in hex for the following instructions:

```
ADD R0,R3
AND R2,R1
OR  R2,R3
SUB R3,R1
```

5. What is immediate operand
6. How can you tell, a CPU is a 8 bit or 16 or 32 or 64 bit CPU?
7. What is the role of clock in the working of a CPU?

3. Understanding CPU bus interface

We have read about the CPU in the previous section. We have learned that the primary job of CPU is to fetch instructions from memory, decode and execute them. This memory could be either ROM or RAM. So CPU can't work alone. It must be connected to the memory chips. For connecting to the other chips, CPU will have a set of pins. These pins are divided into three groups. Address pins, data pins and control pins. In the same way all types of memory chips that are meant for interfacing CPU bus will also have address, data and control pins.



Computing hardware: CPU with bus lines

When CPU chip is soldered to the printed circuit board (PCB) these pins are connected to the address lines, data lines and control lines of the PCB. These lines on the PCB are also connected to the memory chips. In this way PCB is providing a connection between CPU and memory chips.

All address lines put together we refer as address bus, similarly data bus and control bus. Number of lines in the bus is refered as width of the bus. When someone says data bus width is 8, it means that there are 8 data lines in the data bus.

Data Bus

The width of the data bus indicates, how many bits of data, at a time can be transfered between CPU and memory. CPU typically needs to transfer data between registers and memory. So normally the width of the data bus is identical to the CPU register size in bits. If CPU registers are 32 bit in size, then there will be 32 data lines. This allows transfering of whole register at a time between CPU and memory. But in some rare CPUs, data lines may be less than register size. In such case, multiple bus transfers are used.

The Data pins or Data lines are numbered as D0 to D7 for 8 bit data bus, and D0 to D15 for 16 bit data bus. When 8 bit data is transfered over the bus, the Least Signficant Bit(LSB) is transfered on D0 and Most Significant Bit (MSB) on D7.

Address Bus

The number of address lines determines the addressing capacity of the CPU. If CPU got only 8 address lines, it can address only 2 power 8, that is 256 memory locations. If CPU has 16 address lines, then it can address 2 power 16, that is 64 Kilo Bytes (KB) of memory. Note that here Kilo means 1024 not 1000. CPU with 20 address lines can address 1 Mega Bytes of memory. Again here Mega means 1024 x 1024. Similarly CPU with 32 address lines can address 4 Giga Bytes (GB) of memory. One Giga is 1024 x 1024 x 1024.

The Address pins or Address lines are numbered from A0 to A15 for 16 bit address bus. Least significant bit of address is sent on the A0 line and most significant bit on A15.

Bus Cycles

As discussed above, CPU is connected to memory and other chips through its bus lines. Each transaction on the bus, which involves transfer of data is called a bus cycle. The bus cycle describes the hardware protocol for transfer of data. Based on the purpose and direction of data transfer, bus cycles are divided into three types:

- Instruction fetch cycle
- Memory read cycle
- Memory write cycle

Instruction fetch cycle and memory read cycle are identical. But differs only in purpose. CPU's job is always to fetch an instruction by using this instruction fetch cycle. So CPU continuously uses this cycle. The memory read and memory write cycles are used by the CPU, when it is executing Load and Store instructions. Execution of 'Load' instruction, involves movement of data from memory to register. So CPU uses memory read cycle. To execute 'Store' instruction, it uses memory write cycle. Store instruction needs to move data from register to memory.

CPU features compare

If we compare embedded hardware with a car, CPU is like engine of a car. To make a car, engine needs to be connected to wheels through transmission system. Breaking system, fuel system, steering control, seats and so many other things makes car. Similarly to make an embedded hardware board, CPU must be connected with many chips like memory chips and IO chips. This will be the topic of next sections.

When we compare engines of two cars, we compare features like engine volume, number of cylinders, power, torque etc.. Similarly when we wish to compare two CPUs, the following features are normally compared:

- Clock speed
- CPU processing ability (8/16/32/64 bit CPU)
- Data bus width
- Address bus width
- Number of registers
- Instruction set it supports

Our CPU features

The CPU so far we are studying is a simple CPU meant for educational purpose. A graphical simulation of this CPU was developed at DEPIK. Following are the features of this CPU.

- Clock speed 4 MHz
- CPU processing ability (8 bit CPU)
- 8 data lines
- 8 address lines
- 6 registers (4 general purpose registers, Program counter, status register)
- simple RISC model 20 instructions

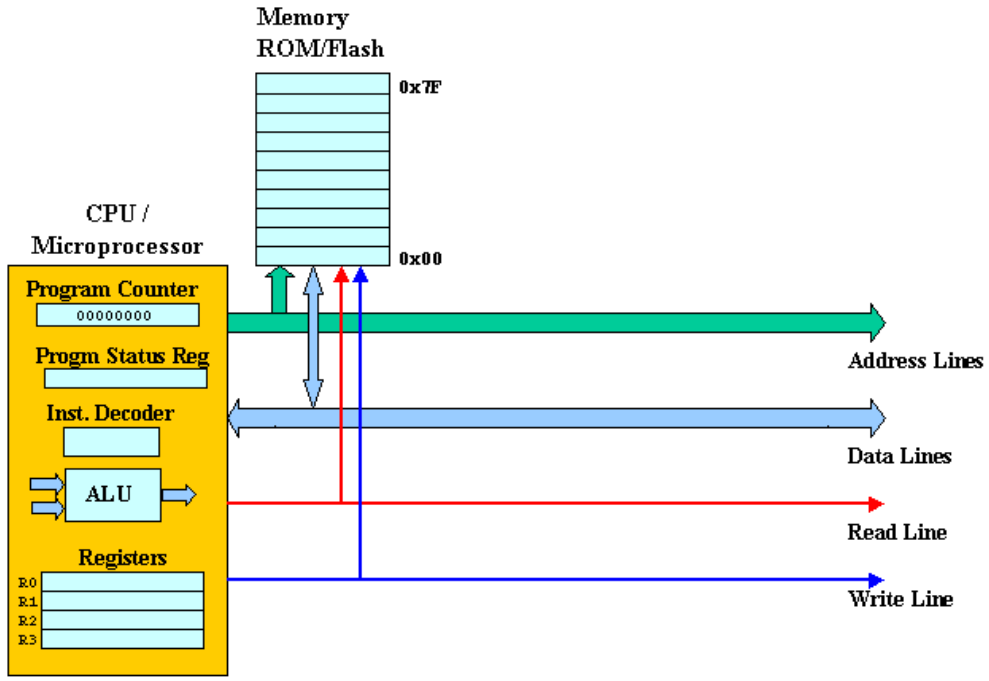
Questions

Answer the following questions to check the understanding of the subject

```
1. What is Data bus and explain its  purpose
2. What is address bus and explain its purpose
3. What is the difference between instruction fetch cycle and memory read cycle?
4. List the CPU features, that can be used to compare two CPUs
```

4. Interfacing ROM with the CPU

The following figure shows how ROM or Flash chip is connected to the CPU's bus lines. Note that Flash is a kind of ROM used extensively now a days. Basically both are non-volatile memory chips. The contents of non-volatile memory chips is not lost when power is off. Where as volatile memory chips like RAM, lose its contents when power is off. Every chip that is connected to the CPU, will have Address pins, data pins and control pins. Control pins are read and write pins. The number of address pins of a memory chip, depends on the size of memory. If memory has got 1 K(1024) locations, then it will have 10 address lines. This is because 2 power 10 is 1024.



Computing hardware: Flash/ROM interfaced to CPU's bus

Every chip also will have one important pin called 'Chip Select' or CS for short. This pin is not shown in the figure. The CS pins of all the chips are connected to one address decoder chip. This address decoder chip (not shown in the figure) takes most significant address lines from the CPU as input and generates individual chip selects lines to each chip.

When CPU places the address on the address lines, based on the address, decoder chip will activate only one chip select line. So for this address only that chip will get selected. Now this chip will be active and participates in the bus cycle. All other chips present on the bus, will not be active as their chip select pin is not selected. In this way, even though multiple chips are connected to the CPU's bus, only one chip will get selected for any bus cycle.

In the figure above, the ROM/flash chip has got 128 bytes of memory. So it will have 7 address lines. CPU's address lines A0 to A6 are connected to these 7 address pins of the memory chip. When CPU places address in the range of 0 to 0x7F, the decoder will activate the chip select pin of this chip.

The eight data pins of memory chip are connected to the 8 data lines of the CPU's bus. The read and write pins of memory chip are connected to the read line and write line of the bus respectively. For a ROM chip, write line may not be present. Flash is normally considered as read only memory, however it also supports write in some special way. But for our purpose, we always treat Flash as read only memory.

When CPU wishes to perform instruction fetch or memory read cycle, it palces the address on the address bus. Because of this address, memory chip will get selected through the address decoder. Once chip is selected the address pins of the chip will select the individual memory location on the chip. If address on the address pins A0 to A6 is 0x24, then 0x24th (36th) location will be selected. While placing the address on the address line, CPU also actives the read control line. This read control line, will inform the seleced chip, that CPU wish to read the contents of the selected memory location. On seeing read pin going active, the memory chip will place the contents of selected location (here 36th location) on the data pins. As these data pins of memory chip are connected to the, CPU's bus lines, the data will reach to the CPU. CPU will read whatever data is present on the data lines.

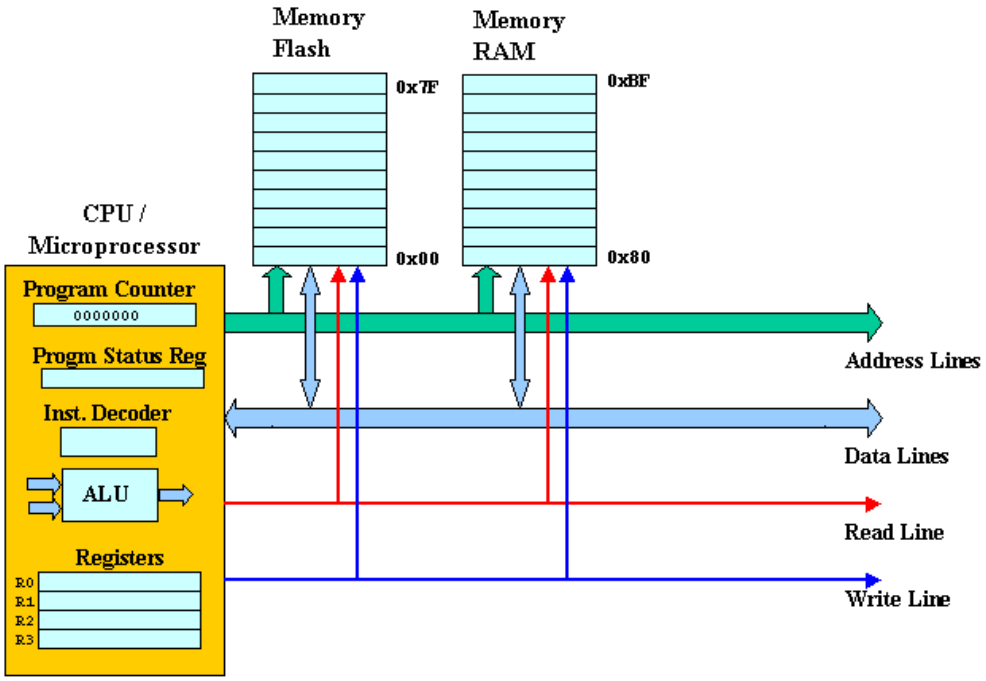
Questions

Answer the following questions to check the understanding of the subject

1. What is the size of ROM/Flash chip interfaced to the CPU ? How many address pins should be present for this ROM/Flash chip?
2. What is the purpose of read pin and write pin present in CPU and Memory chips?
3. Explain the purpose of 'Chip Select' pin present on the ROM chip

5. Interfacing RAM with the CPU

Now we have connected two chips to the CPU's bus. One is ROM/Flash, about which we discussed in the previous section. Now second chip, which is a RAM chip, also connected to the same bus. This RAM chip has got only 64 bytes of memory. CPU can both read and write into this RAM. Where as CPU can only read from the ROM. The RAM chip will have only 6 address pins. The CPU's address bus lines A0 to A5 are connected to these 6 address lines. All 8 data lines of CPU bus are connected to the 8 data pins of RAM chip. Read and Write control lines are also connected to the RAM chip.



Computing hardware: Flash and RAM interfaced to CPU's bus

When CPU places the address in the range of 0x80 to 0xBF, then address decoder will activate chip select line connected to the RAM chip. When RAM chip is selected with chip select line, the address pins (A0 t A5) will select one memory location.

The memory read cycle for RAM will work exactly as described in the previous section for ROM. Now let us look at the memory write cycle.

When CPU wishes to perform memory write to RAM, it places the address in the range of 0x80 to 0xBF on the address bus lines. On seeing this address, the address decoder will activate the chip select line connected to the RAM chip. The address lines A0 to A5 will select a particular memory location. While placing the address on the address lines, CPU also will place the data to write, on the data lines. In this case data lines are acting as output lines from the CPU. The data placed by the CPU will reach the RAM chip's data pins. Now CPU activates the write control pin. This write control pin reaches to the RAM chip and informs RAM to perform write operation. Then RAM will take whatever is there on the data lines and writes to the selected memory location.

Note that in the memory read cycle, CPU's data lines act as input lines. Memory chip drives these data lines, and CPU reads the data from the data lines. Where as in the memory write cycle, CPU's data lines act as output lines. CPU

drives the data lines and memory chip reads the data lines. Because of this, we call data bus as bi-directional bus. In the figure, data bus is having arrow end on both the sides. Address bus lines are always output lines. CPU only puts the address on the address lines. These lines are input to the address decoder chip and selected chip.

Also note that, selection will happen at two levels. On top level, chip will be selected first. Once chip is selected, the address pins of the selected chip will select the individual location in the chip. Finally read and write pins of the chip gets the direction of data move for the selected location.

Questions

Answer the following questions to check the understanding of the subject

1. What are the lowest and highest addresses of ROM and RAM memory locations?
2. What is the size of RAM chip and how many address pins should be present on the RAM chip?
3. Address bus is uni-directional bus, where as data bus is bi-directional, explain the reason for this.
4. When address is placed by the CPU on address lines, the selection of memory location will happen at two levels. What are these two levels? Why two levels are required?

6. Understanding IO and Interfacing Digital IO chip with the CPU

So far we have connected memory chips to the CPU. These two chips are sufficient for the CPU to start working. First we need to put (store) valid instructions (instruction codes) in the Flash. How to store instructions in the Flash is an important topic, we discuss it in a separate section. For the time being, we assume that there exist a valid instructions in the Flash. When power is applied to the CPU, it fetches an instruction from the address present in the program counter. On power on, the default address in the program counter is zero. So CPU reads its first instruction from address 0. Address zero belongs to the Flash.

Note that address zero must belong to Flash only. If address zero belongs to RAM, on power on, no valid contents are present in RAM. All the contents of RAM are lost when power is down. So on power-on RAM does not contain any valid data or instructions. So execution starting address must belong to Flash chip only.

Now CPU fetches instruction from address 0, decodes and executes it. Next it fetches from address 1 and execute. This goes on forever. Now let us think, what these instructions can do. Note that along with instructions, some numbers (data) also can be stored in the Flash. The program can move zero to register 1. Next it loads these numbers one by one into register 2, and add register 2 to register 1. At the end, register 1 will contain the sum of all the numbers present in Flash. Finally program can store this register to a location in RAM.

So the program present in the Flash is adding the numbers present in the data area of Flash and storing the result (sum) in to the RAM. Program can also calculate the average, largest and smallest numbers. And store them in the RAM. But main drawback is that, we can not see anything from outside.

Importance of IO chips and IO devices

CPU with Flash and RAM is like a human brain. Our brain works like a combination of CPU and Flash/RAM. Our brain also can remember (store) two numbers and it can calculate sum or differense. It can also remember(store) the result. But if brain needs to read these numbers from external world, it needs help of eyes, or ears, or hands (to read braille letters). Similarly for the brain to communicate the result to the external world, it needs help of mouth or hands (to write on paper).

Similar to brain, our CPU also needs IO devices for reading the numbers (data) and for displaying the results. So keyboard and LCD display are such IO devices. But CPU can not directly connect to the keyboard or LCD display. CPU can connect to others only through its bus lines. But all most all IO devices can not be connected to the CPU's bus lines. So we need IO interface chips. These IO interface chips are connected to the CPU's bus. And IO devices are connected to these interface chips. In this way IO chips are acting as a bridge between CPU and IO devices. IO chips are also refered as IO controllers or peripheral chips.

There exist so many types of IO chips and each IO chip is meant for connecting to some kind of IO devices. Every IO chip will have a set of IO locations inside. These are called IO registers or simply registers. CPU can read or write to these IO registers, just like memory locations. Similar to memory locations, every IO register in an IO chip will have unique address. By using this IO register address, CPU will read or write to these registers.

These IO registers acts as medium for communication between CPU and IO devices. When CPU wishes to send some data to IO device it writes data to this IO register. Then IO chip will send this data to the IO device. Similarly when IO device sends data, IO chip will receive and put it in IO register. Now CPU can read this data of IO device, from IO register.

CPU Bus interface pins and IO device interface pins

IO chips are also connected to the CPU bus, just like memory chips. So like memory chip, every IO chip will have chip select pin, address pins, data pins, read and write pins. These are the CPU bus interface pins present in every IO chip. Memory chips will have only bus interface pins. Memory chips does not need any other pins. But every IO chip, besides bus interface pins, it will have IO device interface pins. These pins are for connecting to the IO device or IO interface connector.

There exist different types of IO chips, for connecting to differnt types of IO devices. Each such IO chip will have different set of IO interface pins for connecting to a particular type of IO device. Think and look at the display monitor, how it is connected. It is connected through VGA connector. The VGA connector pins are coming from the VGA IO chip called graphics controller. Look at the PS/2 keyboard or PS/2 mouse, how they are connected.

Interface and communication protocol between IO chip and IO device

As an embedded systems engineer, one should understand these interfaces between IO chips and IO devices. That is how data or information is flowing between PS/2 keyboard and PS/2 IO chip. How information is flowing between graphics controller and display monitor.

IO devices in Destop computer vs IO devices in Embedded systems

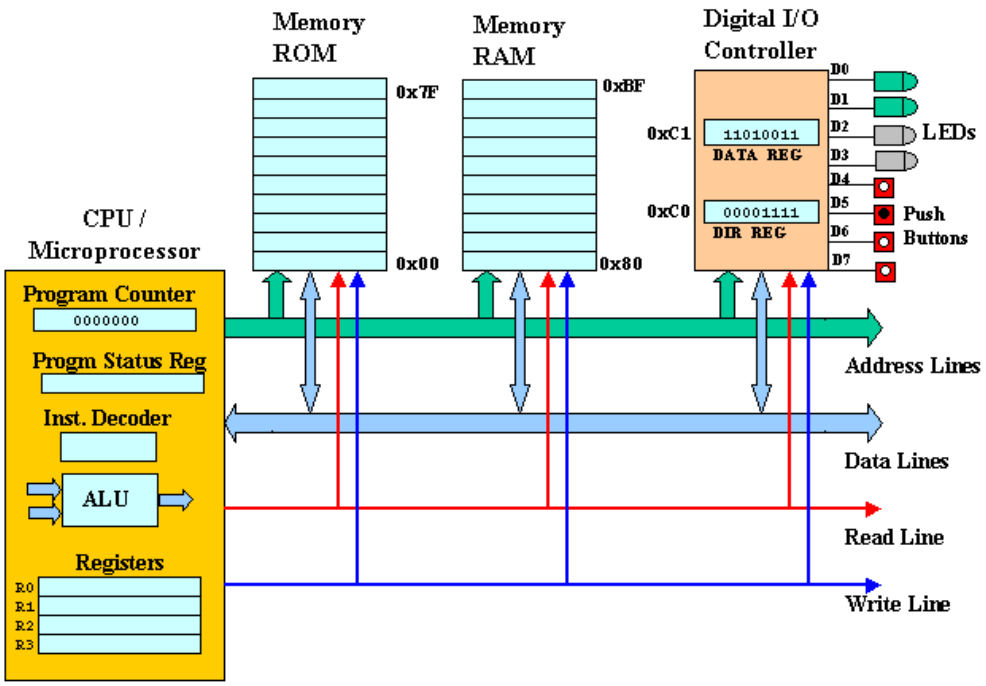
IO devices attached to embedded hardware is quite different from the IO devices attached to regular computers. IO devices attached to regular computer are keyboard, mouse, display monitor, hard disk, printer, speakers, microphone, web camera etc..

In the case of embedded devices, IO devices vary from embedded device to device. Typically they will have a set of buttons like in washing machine or microwave oven. It may also have small LCD or LED displays. Most of the devices are operated through relays. Relays are used to switch On and Off various things like motors, valves, heaters etc.. Other common devices in embedded devices are sensors. Temperature sensor, pressure sensor, humidity, gas, and smoke sensors. Infrared sensors are used for operating the device with a remote.

Digital IO or General Purpose IO controller

The most common IO chip found in every embedded hardware is digital IO also called general purpose IO (GPIO). The digital IO lines provided by the GPIO can be used for interfacing to the various IO devices. These digital IO lines are very simple to understand and use. At the same we can also use these digital IO lines to perform various communication protocols. These communication protocols could be simple to complex.

Look at the following figure. Here we have connected digital IO chip to the CPU's bus. This IO chip has got two IO registers in it. One is called Data register and other is called Direction register. Like memory locations, each of these registers are having unique address. The address of Direction register is 0xC0 and Data register address is 0xC1. As this chip has got only two registers, it will have only one address pin. CPU's address A0 pin is connected to this address pin.



Depending on the need, the digital IO pins of digital controller can be configured as input or output. If IO pin is configured as output, then we can set that line as low or high. When set as low, voltage on this pin will be at ground level, that is zero. When set to high, voltage is at Vcc level, that is 5 volts or 3.3 volts.

When IO pin is configured as input, we can not set the voltage on that pin. Instead we can read the voltage present on that pin. When a pin is configured as input pin, and no voltage is applied to that pin externally, then what voltage will we get ? To avoid this confusion, when pin is configured as input pin, internally a pull-up resister connected to that pin. Because of this pull-up resister, every input configured pin will read as high, when nothing is connected or high level is connected. If this pin is connected to ground (0 voltage), then it will read as low.

In the figure above, the D0 to D3 IO pins of digital IO controller are connected to the four LEDs. The D4 to D7 IO pins are connected to the push buttons. These push buttons, when pressed, will connect the IO pin to ground. When not pressed, it is open (not connected to ground).

Direction Register

The direction of pins can be configured by writing a value into direction register. This direction register is having 8 bits, each bit controls the direction of one IO pin. If bit 0 is 1, then D0 pin is configured as output. If bit 0 is 0, then D0 pin is configured as input pin.

Now look at the direction register in the figure above. The lower four bits of direction register are having ones, so IO pins D0 to D3 are output pins. LEDs are connected to these output pins. The upper 4 bits of direction register are zeros, so D4 to D7 pins are input pins. Push buttons are connected to these IO pins.

Data Register

The output pins can be controlled by writing to the data register. In the figure above, the lower four bits of data register is written with binary 0011. It makes D0, D1 outputs to high and D2,D3 pins to low. The LEDs connected to D0 and D1 are shown as glowing with green color. The LEDs connected to D2 and D3 output pins are shown in off condition with grey color.

The status of input pins can be can be found by reading the data register and checking the bits corresponding to input pins. In the data register the upper four bits gives the status of input pins. The upper four bits are having the binary 1101. It means D4,D6,D7 are in high state and D5 in low state. The push button connected to D5 is in pressed state.

So D5 pins is connected to ground and in low state. Remaining pins are not connected to ground, so they are in high state, because of pull-up resister.

Flashing LED Program

It became a custom to start teaching C programming with 'Hello World' program. Similarly in teaching or learning embedded programming, one should start with flashing LED program. Every embedded hardware will have GPIO controller with so many IO pins. At least one of them will be connected to an LED. So first embedded program we write will configure that IO pin as output. Next it writes 0 and 1 to that IO pin alternatively with delay in between. This delay also implemented very easily with an empty for loop of one million iterations. Now when we run this program, the LED will start flashing on and off with a delay of around 1 second. Now we can see our embedded programming working on the board.

Questions

Answer the following questions to check the understanding of the subject

1. Is it possible for the CPU to work, only when ROM and RAM are connected?
2. Is it possible for the CPU to work, only when ROM is connected?
3. Is it possible for the CPU to work, only when RAM is connected?
4. Explain the difference between IO devices and IO interface chips?
5. What is the purpose of bus interface pins and IO interface pins present in the IO interface chips?
6. Why memory (ROM/RAM) chips will have only bus interface pins?
7. Give list of different types of IO interfaces you aware. Also find out each of these IO interfaces uses how many pins.
8. List the type of IO devices typically found in embedded devices.
9. Explain how CPU can send and receive data to an IO device, through IO interface chip.
10. What is the purpose of digital IO, that is GPIO chip?
11. What are the two registers present in the GPIO interface chip and what are the addresses of each of these registers?
12. What is the purpose of direction register?
13. What is the purpose of data regisger?
14. What is the 'flashing LED' program?

7. Understanding IO communication in general and UART in particular

In the last session, we have connected GPIO controller to our little CPU. We discussed that, with this general purpose IO we can control digital output pins and we can read digital input pins.

Using output pins we can turn on and off LEDs, turn on and off compressor unit in air conditioners. We can also turn on and off water inlet and water outlet valves in washing machine. In fact we can also control the speed of a DC motor by making it on and off very fast. Making on and off very fast, will not stop the motor, but make it run slower than with continuous on. Similarly by making on and off an LED very fast, we can control LED light intensity.

Using input lines, we can read the status of switches or push buttons. We can read status of water tank, whether it is full or not. Door is open or closed. Seat belt in the car is taken or not. Temperature in the water heater reached maximum or not. The possibilities are limit less. Thats why it is called general purpose IO. we can do many things with it.

Data transfer or communication protocols

Another use of GPIO lines is to implement communication protocols. The communication protocols involve transfer of data between two embedded devices, or between embedded device and another chip or between embedded device and computer. Commnication means data transfer, data transfer means moving, bytes of data between the two items involved in the communication. Note that, undestanding this concept of communication (data transfer) is very important. There exist many ways of transfer, and each way is one protocol. In the last section, it was told to pay

attention on how data is flowing between IO chip and IO device. So between every IO chip and IO device there exist a data transfer protocol.

We can implement almost any kind of protocol with the help of GPIO lines. But to implement the data transfer protocol, we have to write lot of code (program) and CPU has to execute it continuously. So there will be a load on the CPU, then CPU will have less time to do more important things. So solution is to build a special IO chip to implement a communication protocol. Now this special chip is connected to the CPUs bus. CPU simply writes the data byte, it wish to transfer into the register of this special chip. Now chip will do all the protocol stuff and transfer that byte by following protocol.

This is like taking some license from goverment office. There are lot of formalities like filling lot of forms, attaching lot of proofs and also doing some other things. Instead we can approach an agent, agent will collect all the required things and follow all formalities and get a license for us. Agent is specialized in this kind of services and saves our time. We can spend our time on more important things. In the same way communication IO chips save the time of CPU.

IO Interfaces and supporting chips

There exist so many communication protocols used for interfacing

- UART/RS-232 serial communication
- Inter IC (I2C) communication
- Serial Peripheral Interface (SPI)
- Inter-IC Sound (I2S) protocol
- Controller Area Network (CAN)
- Secure Digital Input/Output interface (SDIO)
- Universal Serial Bus (USB)
- Ethernet LAN
- Serial ATA (SATA) used to communicate with current hard disks

Following interfaces are a bit out dated.

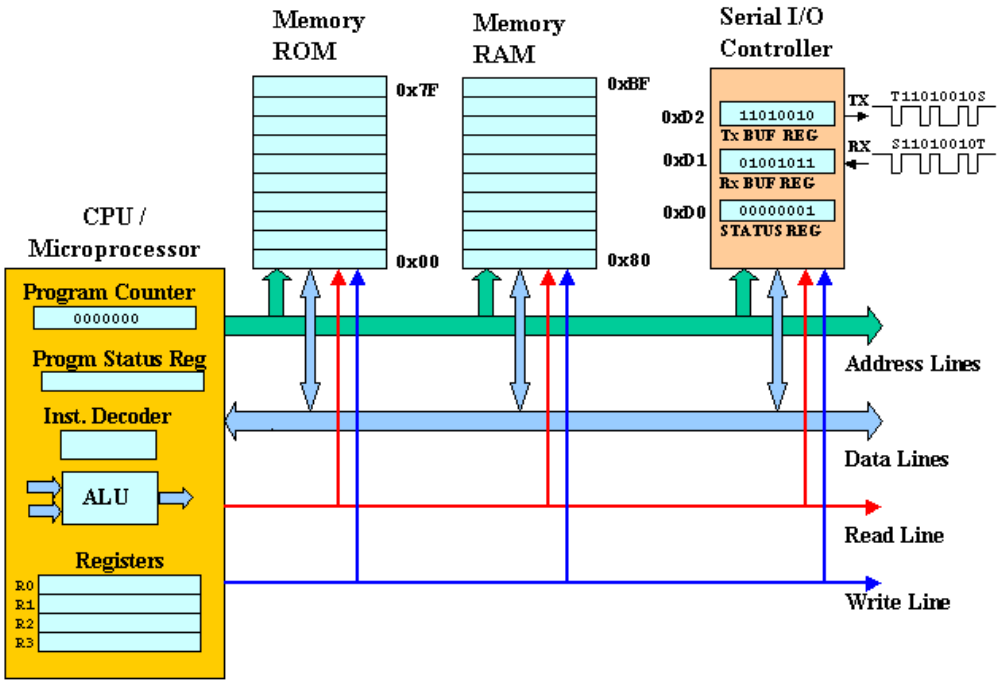
- PS/2 protocol used with PS/2 keyboard and mouse
- Centronics Printer Port or Parallel port protocol
- Parallel ATA (PATA) used to communicate with older hard disks
- Small Computer System Interface (SCSI)

There are many more protocols than listed above. All of them describe, how to transfer data between one computer/device/chip to other computer/device/chip. And there exist specialized IO chips to implement these data transfer protocols. All these chips can be interfaced to the CPU's bus. CPU sends and receives data through these chips, by writing and reading the registers of the chip. These IO chips are acting as bridge (agent) between CPU and IO devices.

RS-232 Serial communication and UART controller

OK, we have discussed enough about how CPU can do data transfers. Now let us discuss about the oldest and most widely used communication available in embedded hardware, that is RS-232 serial communication protocol. The IO chip that supports RS-232 serial communication is called Universal Asynchronous Receiver Transmitter (UART). RS-232 communication is called asynchronous communication, but there exist synchronous communication also, some chips support that mode of communication also. So these chips are called USART chips. Additional S in the name indicates synchrounous. In asynchronous communication each byte is transfered independently. But in synchronous mode a set of bytes will be transfered continuously without any gap between them. Right now we need not worry much about these differences, just remember that RS-232 is an asynchonous communication and we can transfer byte at a time.

The following figure shows how UART chip is interfaced to the CPU. Every IO chip is interfaced to the CPU's bus in the same way. So every IO chip will have chip select pin, read pin, write pin address pins and data pins for connecting to the CPU's bus. Number of address pins, depends on the number of registers present in the chip. The address placed on the address lines identifies the register.



Typical UART chips will have 8 to 10 registers. But our simple UART is having only 3 registers. These are transmit buffer register, receive buffer register and status register. The corresponding register addresses are 0xD2, 0xD1 and 0xD0. When CPU places these addresses and writes data, data will go to these registers. In this way CPU does not know or care, to which chip it is writing. It just places the address on address lines, puts the data on data lines and activates the write line. If this address happens to be RAM location it will get written to RAM. If address happens to be GPIO or UART register address, data will get written to register in that chip.

IO Interface pins

In the last section, we have noted that every IO chip will have two types of pins. One set of pins are bus interface pins for connecting to the CPU's bus. Another set of pins are IO interface pins for connecting to IO device. IO interface pins vary from IO chip to IO chip. In the case of GPIO chip discussed earlier, it got 8 digital IO lines. Now in the case of our simple UART, it got two IO pins called transmit pin and receive pin. Real UART chips may have some more IO pins called RTS, CTS, DSR, DTR etc..

Transmit pin is used to transmit data byte serially bit by bit. Similarly receive pin is used to receive data byte serially bit by bit. Each bit is transmitted on the transmit pin, for a particular duration. This duration desides the transfer speed. Transfer speed is measured as bits per second. Typical transfer speeds are 9600, 19200, 38400, 57600 and 115200 bits per second. If transfer speed is 9600, then each bit duration will be 1/9600. That is 104 micro seconds.

Serial communication devices

One most common use of RS-232 serial communication is between embedded hardware and PC (personal computer). Through this communication, from PC we can transfer the program code to the embedded board. Using RS-232 port, embedded hardware can connect to other devices like GPS (global positioning system) receiver and GSM/GPRS modem. In every device or PC that support RS-232 communication, will have UART controller in it. The transmit pin of embedded board UART is connected to the receive pin of UART in PC/GPS/GSM. Similarly embedded board UART's receive pin is connected to the transmit pin of UART in PC/GPS/GSM

Transfer protocol

When data tranfer is not happening, the transmit pin is in the high position. In this condition tranmit buffer register is empty. That is, there is nothing in the transmit buffer register to transmit. This state is indicated by a bit in the status register. The second bit (bit 1) of the status register is called TXEMPTY bit. If this bit is 1, it indicates that tx buffer is empty. That means we can transfer a byte, by writing a byte value in to this tx buffer register. In the figure above, the second bit of status register is shown as zero. It means that, UART is currently transmitting some byte present in the tx buffer.

If we wish to transfer a byte (char), first we should read the status register and ensure that TX_EMPTY bit of status register is 1. Next we should write the byte into tx buffer register. Then the UART will transmit first a zero bit called start bit. Because of this zero start bit, the tx line, which is high will become low. The tx line going from high to low, will indicate the receiver that a byte transfer is started. Now receiver will become ready to receive. Receiver ignores this start bit and receives next 8 data bits. Receiver must know the bit interval (transfer speed) of transmitter. Both UARTs should work at the same transfer speed. After transmitting the 8 bits of data, transmitter will transmit one bit called stop bit. Because of this stop bit, the tx line will come to high position and remain at high till a new byte transfer is started. So to transfer one byte of data total 10 bits are transfered, which includes one start bit and one stop bit.

When UART receives one complete byte, it places this byte in the receive buffer register and sets the bit 0 in the status register. This bit 0 of status register is called RX_READY bit. If we wish to read a byte from the UART, first we should read status register and check if RX_READY bit is 1. If this bit is 1, it indicates that UART has received some data byte and placed it in rx buffer register. Now we can read the rx buffer register and get the value.

It is very easy to use UART and transfer data to external PC or devices. Similarly we can receive data from PC or other devices, through UART controller.

Questions

Answer the following questions to check the understanding of the subject

1. List at least four communication protocols.
2. What is the difference between UART and USART communication?
3. How many IO interface pins are required for UART communication?
4. What does UART transfer speed 9600 refer to?
5. Calculate the amount to time taken to transfer 100 bytes with a transfer speed of 115200 bits per second.

8. Understanding and using Timer/Counter peripheral chip

Counter or Timer is the third IO or peripheral chip we are interfacing to the CPU. The other two IO chips we already interfaced the CPU are general purpose IO chip and serial communication chip. These three IO chips are very basic and fundamental chips. Every embedded hardware certainly will have these chips. But now a days, embedded hardware contains many more IO interface chips. Some of these chips are listed in the previous section, while discussing various interface or communication protocols.

Counter/Timer as Counter

The Counter/Timer chip is essentially a counter. It counts the clock pulses that received on the input clock pin. Every counter/timer chip will have a counter register in it and also a clock input pin. Whenever a pulse is received on the input clock pin, the value in the counter will be incremented. Basically thats all a counter will do. Program running on the CPU can read this counter register and can find, how many pulses are received so far. We can connect this input pin to a push button. Whenever user pushes and releases the button a pulse is generated, and counter increments by one. This may not be very interesting use of the counter, let us see the more interesting applications of a counter.

If we observe, electricity or energy meter in our home, it keeps producing pulses. We will observe an led is blinking. Led blinks when certain amount of energy is consumed. By counting these pulses, the embedded hardware can

maintain the amount of electricity consumed. In fact the embedded hardware present in the energy meter will do the same thing.

Similarly there are water or liquid flow meters. Whenever certain amount of liquid is passed through this meter it will produce a pulse. By counting these pulses we can measure the quantity of liquid.

It is possible to connect a pulse generating sensor to a rotating wheel of a car or auto. For each revolution of a wheel, the sensor will produce fixed number of pulses. By counting these pulses, we can find the distance travelled. By counting the pulses in every second, the speed of auto can be found.

We can setup an infrared sensor and infrared light source in the path of moving items. For example, items are moving on a conveyor built in a production industry. Whenever item obstructs the light for certain interval, a pulse is generated by the infrared sensor. By counting these pulses we can find the number of items passed across the light source.

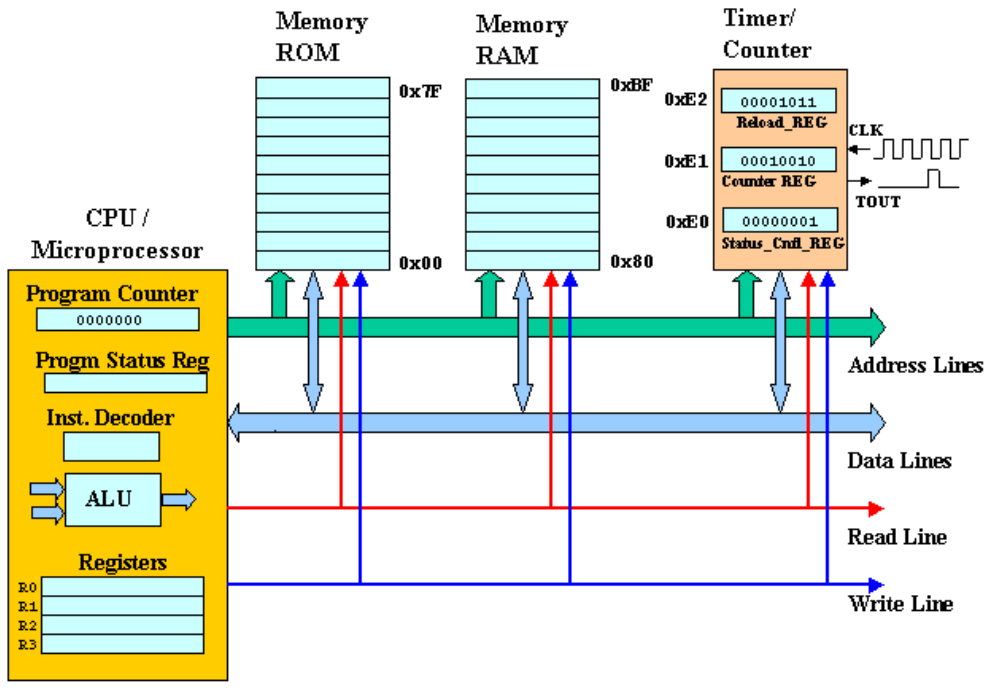
Now we hope that one will be convinced that there is a need for counting the pulses. And the counter chip is just made for this purpose. Also note that, without using any counter chip, we can also measure the pulses by using a digital input pin. But in this case CPU has to continuously read the pin status, and CPU wastes its precious time for this simple counting. So it is better to use counter chip instead of wasting CPU time.

Counter/Timer as Timer

We can use counter as a timer also. In this case, we will connect a fixed, known frequency clock pulses to the input pin of counter. For example if we connect 1 MHz clock source to the input of counter, the counter will increment by one in every micro second. So by reading the counter at two events, we can find the time difference between these two events. If we wish to glow LED for 100 milli seconds, we can switch on the led and start the counter. When counter value reaches 100,000, switch off the led.

Interfacing to CPU

The following figure shows how timer/counter chip is interfaced to the CPU. Note that for the lack of space, we are showing only timer chip in the figure. One should assume that all the three peripheral chips (GPIO, Serial, and Timer) are attached to the CPU bus. In the figure only timer chip is shown. Timer chip will have chip select pin, read pin, write pin, data pins and address pins. These pins are connected to the CPU's bus. CPU can read and write to the registers of this chip.



Interfacing to devices

For interfacing to the external world, the timer chip has two pins. One is clock input pin, through which counter receives the pulses. In counter application this pin is connected to the external device like energy meter or water meter. To use this as timer, this pin is connected to the on-board clock oscillator.

The second pin is timer output pin. This pin can be used to generate output pulses, based on the timer events. For example a pulse will be generated on this pin, whenever timer receives 1000 input clock pulses. So assuming timer clock is 1 MHz, a output pulse is generated once in every millisecond. One very common application of timer is to use this pulse to generate periodic CPU interrupts.

Registers

This chip has got the following three registers. Each of these register is having unique address, to access by the CPU.

- Counter Register
- Match Register
- Status and control register

By writing to control register, we can start and stop the counter from counting. With control register, we can also control the behaviour of timer and timer output pin. Using the status bits we can find whether counter is reached specified value or not. We can write the maximum count value to count, in the match register. When counter reaches this maximum value, one of the folliwing things can happen.

On reaching match register value, start the counter from 0 onwards. Also set the status bit in the status register, indicating that counter has reached the maximum value. Generate a pulse on the output pine.

Questions

Answer the following questions to check the understanding of the subject

```
1. Explain the difference between counter and timer applications
2. What is common application of output pulses generated by the timer?
```

9. Loading a program to Flash/RAM

We have learned about the hardware used in an embedded system. Now we know that, when power-on an embedded hardware, the CPU will fetch (i.e. read) instructions from the Flash, decode and execute them.

We also learned that each instruction is a number (machine code) stored in the Flash. Typical size of instruction is 16 bits or 32 bits. But our simple CPU uses 8 bit instructions. But few instructions are 16 bit in size. I hope you remembered that, in each instruction, some bits will specify the operation to perform. And other bits will specify the operands on which operation to perform. In most of the instructions, operands will be register numbers, but in some instructions immediate data operands also present.

Now we wish to write a small program and run it on our embedded hardware board. So first thing is that, we have to write a program. We can write a program in three possible ways:

- Directly write machine codes (i.e. instruction codes or numbers)
- Write in assembly language instructions and generate machine codes.
- Write in C language and again generate machine codes from C program

If we write a program in assembly language, then we need a software called assembler. This assembler takes the assembly language program, we have written and converts into an executable file containing machine codes. In a microprocessor lab, if anyone have used a software called MASM, it is an assembler software. It converts 8086 assembly instructions to machine codes.

If we write a program in C language, then we need a software called compiler. This compiler takes the C program file and converts it into an executable file containing machine codes. Important point is that, whichever way we write, finally we need machine codes to put in the Flash of embedded hardware.

Let us assume that, we have written a program and stored the final machine codes in a file with name 'program.hex'. Now we need to copy the contents of this file to Flash memory chip.

There are two popular methods to load a program to flash.

- Through JTAG port
- By using bootloader running on ROM, through UART/SPI/USB

For the first method a JTAG hardware converter is required. The corresponding JTAG software running on PC can download the program to Flash.

In the second method any serial to USB cable is sufficient. In this method the bootloader program present in the ROM runs first, this program receives the program over UART/SPI/USB and loads to the Flash. In this case also PC should run a program, which understands bootloader commands and send program file to it.

Questions

Answer the following questions to check the understanding of the subject

1. What are the three ways of writing a program?
2. In your opinion, what is easiest of writing a program and what is the tough way of writing program. Explain the reasons.
3. What are the two methods of loading a program to Flash.