MODULE 4

Embedded Firmware Design and Development-Firmware design approaches,Firmware Development languages.Integration of embedded hardware and firmware Embedded Product development Lifecycle-Objectives,Different phases,Modeling Techniques-Waterfall Model,Incremental Model,Evolutionary model,Spiral Model

Q.1.What is embedded firmware?

- Embedded firmware is responsible for controlling various peripherals of the embedded hardware and generating responses in accordance with the functional requirements mentioned in the requirements for the particular product
- Firmware is considered as the master brain of the embedded systems
- Imparting intelligence to an embedded system is a one time process and it can happens at any stage of the design
- Once the intelligence is imparted to the embedded product, by embedding the firmware in the hardware, the product start functioning properly and will continue serving the assigned task till hardware breakdown occurs or a corruption occurs.
- Firmware requires understanding of embedded product hardware like, various component interfacing, memory map details I/O port details, configuration and register details of various hardware chips used and some programming language.
- Embedded firmware development process start with conversion of firmware requirements into a program model using modeling tools like UML or flow chart based representation

Q.2.Explain embedded firmware design approaches.

- Firmware design approaches depends on the
  - –>Complexity of the function to be performed.
  - –>Speed of operation required,Etc.
- Two basic approaches for firmware design.

->Conventional Procedure based Firmware Design/Super Loop Design. (Non OS based design) *Ref Q.3,Q.4*

->Embedded Operating System Based Design. *Ref Q.5*

Q.3.Explain super loop based Design(Firmware design approach)

OR

Explain the advantages and disadvantages of super loop based approach

- This approach is applied for the applications that are not time critical and the response time is not so important .
- Similar to the conventional procedural programming where the code is executed task by task.
- Task listed at the top of the program code is executed first and task below the first task are executed after completing the first task.
- True procedural one.
- In multiple task based systems, each task executed in serial.

- ❖ Firmware execution flow of superloop based approach will be:
  1.Configure the common parameter and perform initialization for various hardware components, memory, registers etc.
  2.Start the first task and execute it
  3.Execute the second task
  4.Execute the next task
  5. 5. ….
  6. Execute the last defined task
  7. Jump back to the first task and follow the same flow.

Operations are infinite loop based approach In terms of C program code as:

```
Void main(){
configuration(); initializations(); while(1){ task1();
task2();
….. taskn();
```

- Almost all task in embedded applications are non-ending and are repeated infinitely throughout the operation.
- By analyzing C code we can see that the task 1 to n are performed one after another and when the last task is executed, the firmware execution is again redirected to task 1 and it is repeated forever in the loop.
- This repetition is achieved by using an infinite loop(while(1)). Therefore Super loop based Approach. Since the task are running inside an infinite loop, the only way to come out of the loop is either
- -Hardware reset=A Hardware reset brings the program execution back to the main loop.
  - Interrupt assertion= Whereas the interrupt suspend the task execution temporarily and perform the corresponding interrupt routine and on completion of the interrupt routine it restart the task execution from the point where it got interrupted.
- Super Loop based design does not require an OS, as there is no need for scheduling which task is to be executed and assigning priority to each task.
- In a super Loop based design, the priorities are fixed and the order in which the task to be executed are also fixed.
- Hence the code for performing these task will be residing in the code memory without an operating system image.
- This type of design is deployed in low-cost embedded products where the response time is not time critical.
- Some embedded products demand this type of approach if some tasks itself are sequential.
- Example of ― Super Loop Based Design‖ is -Electronic video game toy containing keypad and display unit.

- **Drawback of Super Loop based Design:**
  - Failure in any part of a single task will affect the total system.
  - If the program hang up at any point while executing a task, it will remain there forever and ultimately the product will stop functioning.
  - Lack of real timeliness-If the number of tasks to be executed within an application increases, the time at which each task is repeated also increases. This brings the probability of missing out some events.

- **Some remedial measures**

- •Use of Hardware and software Watch Dog Timers (WDTs) helps in coming out from the loop when an unexpected failure occurs or when the processor hang up may cause additional hardware cost and firmware overhead

Q.4.Explain execution flow in super loop based firmware design

Firmware execution flow of this will be:
1.Configure the common parameter and perform initialization for various hardware components, memory, registers etc.
2.Start the first task and execute it
3.Execute the second task
4.Execute the next task
5.5. ….
6.Execute the last defined task
7.Jump back to the first task and follow the same flow.

Operations are infinite loop based approach In terms of C program code as:

```
Void main(){
configuration();
initializations();
while(1)
{
task1();
task2();
… taskn();
} }
```

Q.5.Explain embedded OS based approach for firmware design

- Contains OS, which can be either a General purpose Operating System (GPOS) or real Time Operating System (RTOS).
- POS based design is very similar to the conventional PC based Application development where the device contain an operating system and you will be creating and running user applications on top of it
- Examples of Microsoft Windows XP OS are PDAs, Handheld devices/ Portable Devices and point of Sale terminals.
- RTOS based design approach is employed in embedded product demanding Real Time Responses.RTOS respond in a timely and predictable manner to events.
- RTOS contain a real time Kernel responsible for performing pre- emptive multi tasking scheduler for scheduling the task, multiple thread etc.
- RTOS allows a flexible scheduling of system resources like the CPU and Memory and offer some way to communicate between tasks
- Examples of RTOS are :Windows CE, pSOS, VxWorks, ThreadX, Micro C/OS II,Embedded Linux, Symbian etc…

Q.6.Describe about embedded firmware development languages.

For embedded firmware development you can use either:

1.Target processor/controller specific language(Assembly language) Ref Q7,8,9.
2.Target processor/ controller independent language (High level languages)RefQ10,11
3.Combination of Assembly and high level language.Ref Q12

Q.7.Explain Assembly language based development. (Target processor/controller specific language)

- Assembly language is human readable notation of machine language whereas machine language is a processor understandable language.
- Machine language is a binary representation and it consist of 1s and 0s.
- Machine language is made readable by using specific symbols called _mnemonics‘. Hence machine language can be considered as an interface between processor and programmer.
- Assembly language and machine languages are processor dependant and assembly program written for one processor family will not work with others.
- Assembly language programming is the task of writing processor specific machine code in mnemonics form, converting the mnemonics into actual processor instructions
- The general format of an assembly language instruction is Opcode followed by the Operand.Opcode tells what to do and Operand gives the informationto do the task The operand may be single operand, dual operand or more.
    - MOV A, #30(Single operand instruction)
        - Here MOV A is the opcode and 30 is Operand
        - Same instruction in machine language like this 01110100 00011110
    - INC A(operand implicitly)
        - ->The machine language representation is 00000100
            - ➔ This instruction increment the 8051 Accumulator register content by 1.
    - LJMP 16 bit address(dual operand)
        - ->The machine language for the same is –00000010 addr_bit15 to addr_bit8 addr_bit7 to addr_bit0
        - ->The first binary data is the representation of LJMP machine code
        - ->The first operand that immediately follow the opcode represent the bit 8 to 15 of the 16 bit address to which the jump is requested and the second operand represent the bit 0 to 7 of the address to which the jump targeted.
- Assembly language instructions are written in one per line.
- A machine code program thus consisting of a sequence of assembly language instructions, where each statement contains a mnemonic.
- Each line of assembly language program split into four field as given below
        - LABEL OPCODE OPERAND COMMENTS
- Label is an optional field. A label is an identifier to remembering where data or code is located
- LABEL is commonly used for representing :-
    - –A memory location, address of a program, sub-routine, code portion etc…
    - –The max length of the label differs between assemblers. Labels are always suffixed by a colon and begin with a valid character. Labels can contain numbers from 0 to 9 and special character _
    - –Labels are used for representing subroutine names and jump locations in Assembly language programming
        - DELAY: MOV R0, #255 ; load Register R0 with 255
        - DJNZ R0, DELAY ; Decrement R0and loop till R0=0 RET ; return to callingprogram
        - LCALL DELAY
- The assembly program contains a main routine which start at address 0000H and it may or may not contain subroutines.
- In main program subroutine is invoked by the assembly instruction LCALL DELAY. Executing this instruction transfers the program flow to the memory address referenced by the _LABEL‘ DELAY
- While assembling the code a ;‘ inform the assembler that the rest of the part coming in a line after the ;‘ symbol is comments and simply ignore it

- Each assemblyinstruction should be written in a separate line More than one ASM code lines are not allowed in a single line.
- We can directly replace the LABEL by putting desired address first and then writing assembly code for the routine:
    - ORG 0100H
    - MOV R0, #255   ; load Register R0 with 255
    - DJNZ R0, 0100   ; Decrement R0and loop till R0=0 RET   ; return to callingprogram
- ORG 0100H is not an assembly language instruction; it is an assembler directive instruction. It tells the assembler that the instruction from here onwards should be placed at location starting from 0100H.

Q.8.State various advantages of assembly language based development.

Advantage Of Assembly Language Based Development

- Thorough understanding of the processor architecture, memory organization, register set and mnemonics is very essential for Assembly Language based Development.

1.)Efficient Code Memory and data Memory Usage (Memory Optimization)

–Since the developer is well versed with the target processor architecture and memory organization, optimized code can be written for performing operations,this lead to the less utilization of code memory and efficient utilization of data memory

2.)High Performance

–Optimized code not only improve the code memory usage but also improve the total system performance

3.)Low level Hardware access

- –   Most of the code for low level programming like accessing external device specificregisters from the operating system kernel, device drivers and low level interrupt routine etc. are making use of direct assembly coding since low level device specific operation support is not commonly avail with most of the high level language compilers

4.) Code Reverse Engineering

–Reverse Engineering is the process of understanding the technology behind a product by extracting the information from the finished product.

–Reverse engineering is performed by 'hawkers' to reveal the technology behind the proprietary product.

–Though most of the product employ code memory protection, if it may be possible to break the memory protection and read the code memory, it can easily be converted into assembly code using dis-assembler program for the target machine

<u>Drawbacks Of Assembly Language Based Development</u>

1.)High Development time

–Assembly language programs are much harder to program than high level languages

–Developer must have thorough knowledge of architecture, memory organization and register details of target processor in use

–Learning the inner details of the processor and its assembly instructions are high time consuming and it create delay impact in product development

Solution

->Use a readily available developer who is well versed in target processor architecture assembly instructions

->Also more lines of assembly code are required for performing an action which can be done with a single instruction in a high level language like C

2.)Developer Dependency

–>There is no common rule for developing assembly language based applications whereas all high level language instruct certain set of rules for application development

–>In Assembly language programming, the developers will have the freedom to choose the different memory locations and registers

–>If the approach done by a developer is not documented properly at the development stage, it may not be able to recollect at later stage or when a new developer is instructed to analyze the code , he may not be able to understand what is done and why it is done

–>Hence upgrading/modifying on later stage is more difficult Solution

->Well documentation

3.)Non-portable

–Target applications written in assembly instructions are validonly for that particular family of processors

->Exmple—Application written for Intel X86 family of processors

–>Cannot be reused for another target processors

–>If the target processor changes, a complete rewriting of the application using assembly instructions for the new target processor is required.

Q.9.Explain assembly language to machine language conversion process

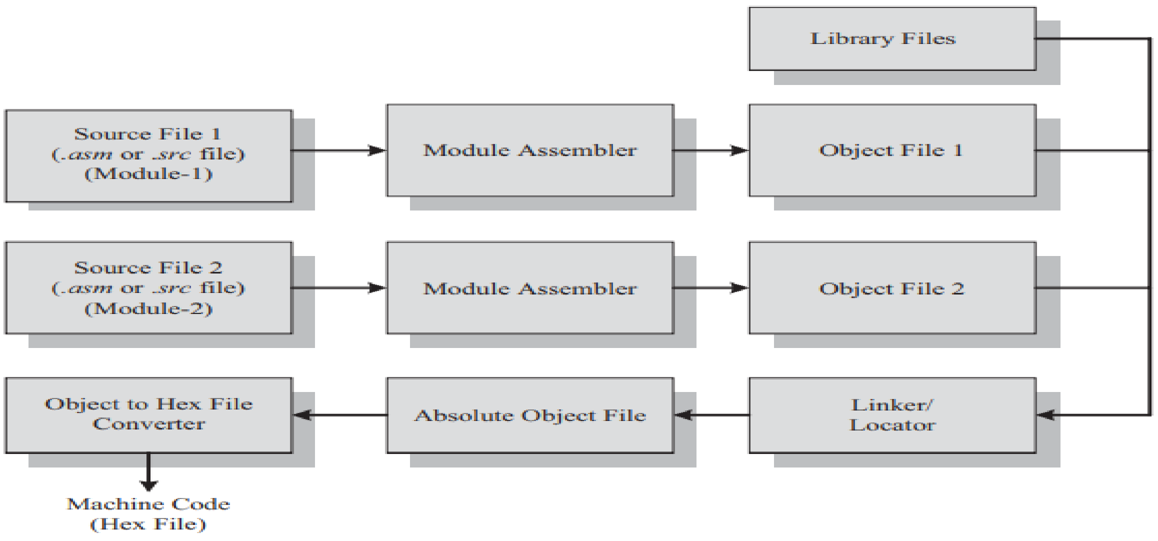


Fig. 9.1 Assembly language to machine language conversion process

- Conversion of assembly language to machine language is carried out by a sequence of operations
  - ❖ *Source file to object file translation*
  - ❖ *Library file creation*
  - ❖ *Linker and locater*
  - ❖ *Object to Hex file converter*

<u>Source file to object translation</u>

- Translation of assembly code to machine code is performed by assembler
- Assemblers from multiple vendors are available
- Some target processor assembler are freely available for downloading from internet
- A51 Macro assembler from Keil software is a popular assembler for 8051 microcontroller.
- Assembler performs the translation of assembly code to machine code.
- Assembly language program will be saved as .asm or .src file.
- Any text editor can be used for writing assembly language instructions.
- Multiple source files called modules exist.(modular programming)

- Each modules can be assembled separately to examine syntax errors.
- On successful assembling .src file will be converted to object file with extension .obj.
- Each module can share variables and subroutines among various modules by declaring variable or function as PUBLIC or EXTERN keywords.
- Object file(.obj) does not contain absolute address of where generated code needs to be placed in memory.
- Importing a variable or a function from a module is done by declaring variable or function as EXTERN in module where it is going to be accessed.
- PUBLIC keyword informs assembler that variables or functions need to be exported to other modules.
- For all those modules using variables or function with EXTRN keyword, there should be one and only one module which export those variables/functions PUBLIC keyword
- If more than one module in a project tries to export variables or functions with the same name using PUBLIC keyword, it will generate linker errors.
- If a variable or function declared as EXTRN in one or two modules, there should be one module defining these variables or function and exporting them using PUBLIC keyword
- If no module in a project export the variable or functions which are declared as EXTRN in other modules it will generate linker warnings or error depending on the error level/warning level setting of the linker.

Library file creation and usage

- Libraries are sepecially formatted ordered program collection of object modules
- Library files are generated with extension .lib
- Library files is some kind of source code hiding technique
- LIB 51 from Keil software is an example of library creator for A51 Assembler.

Linker and locater

- Software utility responsible for linking various object modules in a multi module project and assigning absolute address
- Linker link any external dependent variables or functions declared on various modules and resolvew the external dependencies among modules
- When linker processes a library, only those object modules in library that are necessary to create program are used.

BL51 from Keil software is an example for linker and locator for A51 assembler

Object to hex file converter

- Hex file is the repersentation of machine code and hex file is dumpted into the code memory of processor
- Hex file representation varies depending on target processor/controller.
- Absolute object file is used for creating hex files for dumping into code memory of the processor/controller.
- OH51 from keil software is an example of object to hex file converter.

Q.10.Explain any one of firmware development languages

OR

Explain high level language based development(Target processor/ controller independent language)

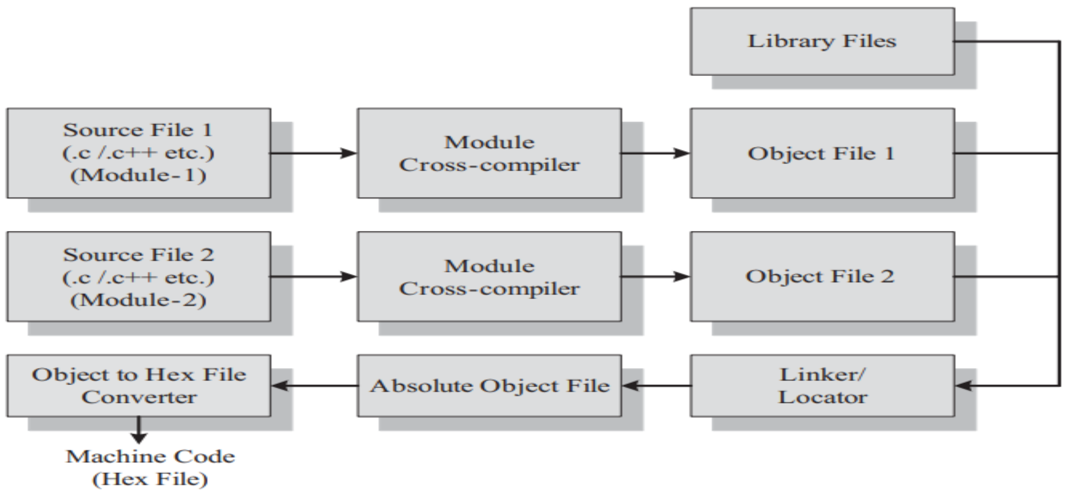


Fig. 9.2 **High level language to machine language conversion process**

- Conversion of assembly language to machine language is carried out by a sequence of operations
    - ❖ *Source file to object file translation*
    - ❖ *Library file creation*
    - ❖ *Linker and locater*
    - ❖ *Object to Hex file converter*

Source file to object translation

- Any High level language with a supported cross compilers for the target processor can be used for embedded firmware development
- Cross Compilers are used for converting the application development in high level language into target processor specific assembly code
- Most commonly used language is C
- C is well defined easy to use high level language with extensive cross platform development tool support.
- Any text editor provided by IDE tool supporting the high level language in use can be used for writing the program
- Most of the high level language support modular programming approach and hence you can have multiple source files called modules written in corresponding high level language.
- Programs in high level language can be written in any notepad or wordpad and are saved as .c or .cpp.
- Several source modules exist in the form of .c file.
- Translation of high level code to executable object code is done by a cross compiler.
- C51 is a cross compiler from Keil software for 8051.
- Object file(.obj) does not contain absolute address of where generated code needs to be placed in memory.
- Multiple source files called modules exist.(modular programming)
- Each modules can be compiled separately to examine syntax errors.
- On successful assembling .c or .cpp file will be converted to object file with extension .obj.
- Object file(.obj) does not contain absolute address of where generated code needs to be placed in memory.

Library file creation and usage

- Libraries are sepecially formatted ordered program collection of object modules

- Library files are generated with extension .lib
- Library files is some kind of source code hiding technique
- LIB 51 from Keil software is an example of library creator for A51 Assembler.

<u>Linker and locater</u>

- Software utility responsible for linking various object modules in a multi module project and assigning absolute address
- Linker link any external dependent variables or functions declared on various modules and resolvew the external dependencies among modules
- When linker processes a library, only those object modules in library that are necessary to create program are used.
- BL51 from Keil software is an example for linker and locator for A51 assembler

<u>Object to hex file converter</u>

- Hex file is the repersentation of machine code and hex file is dumpted into the code memory of processor
- Hex file representation varies depending on target processor/controller.
- Absolute object file is used for creating hex files for dumping into code memory of the processor/controller.
- OH51 from keil software is an example of object to hex file converter.

Q.11.Explain advantages and disadvantages of high level language based development.

Advantages Of High Level Language Based Development

1.)Reduced Development Time

–Developers requires less or little knowledge on the internal hardware details and architecture of the target processor

–>Syntax of high level language and bare minimal knowledge of memory organization and register details of target processor are the only pre- requisites for high level language based firmware development

–>With High level language, each task can be accomplished by lesser number of lines of code compared to the target processor specific assembly language based development

2.)Developer Independency

–>The syntax used by most of the high level languages are universal and a program written in high level language can be easily be understood by a second person knowing the syntax of the language

–>High level language based firmware development makes the firmware, developer independent

–>High level language always instruct certain set of rules for writing code and commenting the piece of code

3.) Portability

–>Target applications written in high level languages are converted to target processor understandable format by a cross compiler

–>An application written in high level language for a particular target processor can be easily converted to another target processor with little effort by simply recompiling the code modification followed by the recompiling the application for the required processor. This makes the high level language applications are highly portable.

Limitations of High Level Language Based Development

- Some cross compilers avail for the high level languages may not be so efficient in generating optimized target processor specific instructions
- Target images created by such compilers may be messy and no optimized in terms of performance as well as code size.

Q.12.Explain any one firmware development language

OR

Explain the usage of Mixing Assembly and high level language

High level language and assembly languages are usually mixed in three ways

1.)Mixing assembly language with high level language
2.)Mixing high level language with Assembly
3.)In line assembly programming

1.)Mixing Assembly Language with High level Language (Assembly Language with 'C')

- Assembly language instructions are added to programs written in C.
- Situations where assembly language is mixed with C code are:
- Where cross compilers don't have features like ISR(Interupt service routine).
- To increase speed and to get optimized code.
- Where time specification is critical.
- Main issue while mixing assembly language with C routine is passing of parameters from C routine to aseembly code and returning of values from aseembly routine to C code.
- C51 cross compiler from Keil helps in mixing assembly language with C.
- Steps involved in conversion are
    1. Write a simple function in C that passes parameters and returns values the way you want your assembly routine to.
    2. Use the *SRC* directive (*#PRAGMA SRC* at the top of the file) so that the *C* compiler generates an *.SRC* file instead of an *.OBJ* file.
    3. Compile the *C* file. Since the *SRC* directive is specified, the *.SRC* file is generated. The *.SRC* file contains the assembly code generated for the C code you wrote.
    4. Rename the *.SRC* file to *.A51* file.
    5. Edit the *.A51* file and insert the assembly code you want to execute in the body of the assembly function shell included in the .A51 file.

2.)Mixing high level language with Assembly

Mixing code written in high level language with Assembly language are used in situations like:

1.Source code is already in Assembly language and routines in high level language need to be included.

2.Entire source code is planned in assembly language for code optimization,optimal performance,memory utilization but some portion is difficult to code in Assembly language(like division operation in 8051).

3.To include built in functions written in C language by cross compiler.

- Main issue by mixing high level language with assembly code is while parameters are passed to the function and values are returned between high level code and assembly code using CPU registers,stack memory and fixed memory.
- C51 allows passing maximum of 3 arguments through general purpose registers R2 to R7.
- If parameters passed are char variables they are passed to functions using registers R7…R5.
- If parameters are int values,they are passed using (R7,R6)(R5,R4) and (R3,R2).
- If number of arguments is greater than 3,first 3 arguments pass values through registers and rest through memory locations.

3. **Inline Assembly:**
   - Technique of inserting target processor/controller specific assembly instruction at any location of source code written in high level language.
   - This avoids delay in calling assembly routine from C code.
   - Special keywords are used to represent start and end of assembly instructions.
   - Eg: #pragma asm(indicate start of block) #pragma endasm(indicate end of block).

```
#pragma asm
MOV A,#13
#pragma endasm
```

Q.13.What is meant by integration of hardware and firmware.

- Integration of hardware and firmware deals with the embedding of firmware(software code) into the target hardware board.
- Embedded hardware consist of PCB with all its necessary components.
- Embedded firmware represents control algorithm and configuration data to implement product requirements.
- Embedded firmware will be in machine language form.
- It is the process of 'Embedding Intelligence' to the product.
- A variety of techniques are used for embedding the firmware into the target board.
- Non OS based embedded system store firmware either in the on chip processor or off chip memory.
- Different firmware embedding techniques are
  i.Out of circuit programming
  ii.In system programming (ISP)
  iii.In Application Programming (IAP)
  iv.Use of factory programming

Q.14.What are the different techniques for embedding firmware into hardware

OR

What are the different techniques for embedding firmware into the target board.

OR

What are the different firmware embedding techniques.

- Various firmware embedded techniques are:
  - i.Out of circuit programming
  - ii.In system programming (ISP)
  - iii.In Application Programming (IAP)
  - iv.Use of factory programming chip

<u>i.)Out of circuit programming:</u>

- It is performed outside the target board.
- The processor or memory chip into which the firmware needs to embedded is taken out of the target board and it is programmed with the help of programming device.
- The programming device is dedicated unit when contains the necessary hardware circuit to generate the programming signals.
- The programmer contains a ZIF socket with locking pin to hold the device to be programmed.
- The programming device will be under the control of a utility program running on a PC.
- The programmer is interfaced to the PC through RS-232C/USB/Parallel Port Interface.
- The commands to control the programmer are sent from the utility program to the programmer through the interface.

<u>Sequence of Operations for embedding the firmware with a programmer:.</u>

1.Connect the programming device to the specified port of PC (USB/COM port/parallel port)
2.Power up the device.(Ensure that the power indication LED is ON)
3.Execute the programming utility on the PC and ensure proper connectivity is established between the PC and programmer. In case of error turn off device power and try connecting it again.
4.Unlock the ZIF socket by turning the lock pin.
5.Insert the device to be programmed into the open socket as per the insert diagram shown on the programmer.
6.Lock the ZIF socket.
7.Select the device name from the list of supported devices.
8.Load the hex file which is to be embedded into the device.
9.Program the device by 'Program' option of utility program.
10.Wait till the completion of programming operation.(Till busy LED of programmer is off)
11.Ensure that programming is successful by checking the status LED on the programmer (Usually 'Green' for success and 'Red' for error condition) or by noticing the feedback from the utility program.
12.Unlock the ZIF socket and take the device out of programmer.

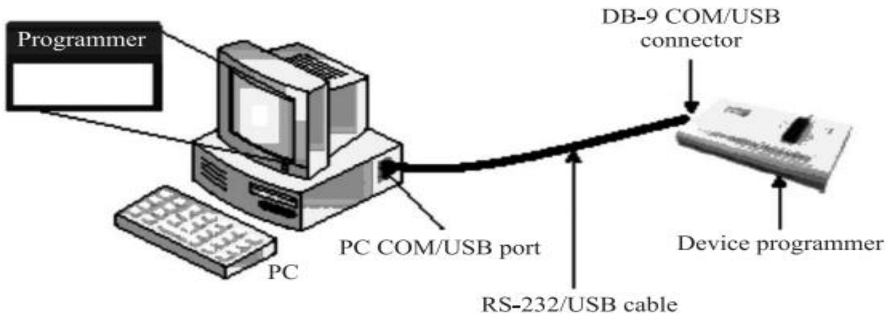


Fig. 12.2 Interfacing of Device Programmer with PC

<u>Drawbacks of out of circuit programming:</u>

- High development time.
  Whenever the firmware is changed, the chip should be taken out of the board for reprogramming. This tedious and prone to chip damages due to frequent insertion and removal. Better use a socket on the board side to hold the chip till the firmware modifications are over.
- Not suitable for batch production.
- Difficult in upgrading the firmware
  Once product is deployed in the market it is very difficult to upgrade the firmware.

<u>ii.)In System Programming (ISP)</u>

- In ISP, programming is done within the system.
- The firmware is embedded into the target device without removing it from the target board.
- It is the flexible and easy way of firmware embedding.
- The only pre-requisite is that the device must have an ISP support.
- A part from target board, PC, ISP utility and ISP cable, no other additional hardware is required for ISP.
- Chips supporting ISP generate programming signals using suppl voltage.
- In order to perform ISP operations the target device should be powered up in a special 'ISP mode'.
- ISP mode allows the device to communicate with an external host through a serial interface, such as a PC or terminal.

- Key player behind ISP is factory programmed memory(ROM) called Boot ROM.
- The device receives commands and data from the host, erases and reprograms code memory according to the received command.
- Once the operations are completed, the device is re-configured so that it will operate normally.

<u>In System Programming with SPI Protocol</u>

- Devices with SPI In System Programming support contains a built-in interface and a on-chip EEPROM or FLASH memory is programmed through this interface.
- The primary I/O lines involved in SPI-In System Programming are listed below,
  a. MOSI – Master Out Slave In
  b. MISO – Master In Slave Out
  c. SCK – System Clock
  d. RST – Reset of Target Device
  e. GND – Ground of Target Device.
    - PC act as master and target device act as slave in ISP.
    - Program data is sent to MOSI pin of target device and acknowledgement originated from MISO PIN.
    - SCK pin act as clock for data transfer.
    - Target device works under a supply voltage of 5Volt,so connect lines of target device with parallel port of PC.

<u>*Steps for the operation or power up sequence for In System Programming for Atmel's AT89S series microcontroller:</u>
1.Apply supply voltage between VCC and GND pins of target chip.
2.Set RST pin to "HIGH" state.
3.If a crystal is not connected across pins XTAL1 and XTAL2, apply a 3 MHz to 24MHz clock to XTAL1 pin and wait for at least 10 milliseconds.
4.Enable serial programming by sending the Programming Enable serial instruction to pin MOSI/P1.5. The frequency of the shift clock supplied at pin SCK/P1.7 needs to be less than the CPU clock at XTAL1 divided by 40.
5.The Code or Data array is programmed one byte at a time supplying the address and data together with the appropriate Write instruction. The selected memory location is first erased before the new data is written. The write cycle is self – timed and typically takes less than 2.5 ms at 5V.
6.Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO/P1.6.
7.After successfully programming the device, set RST pin low or turn off the chip power supply and turn it ON to commence the normal operation.

<u>iii.)In Application Programming (IAP)</u>
- In Application Programming (IAP) is a technique used by the firmware running on the target device for modifying a selected portion of the code memory.
- It is not a technique for first time embedding of user written firmware.
- It modifies the program code memory under the control of embedded application.
- Updating calibration data, look-up tables, etc., which are stored in the code memory, are typical examples of IAP.
- The Boot ROM resident API instructions which perform various functions such as programming, erasing and reading the Flash memory during ISP mode are made available to the end-user written firmware for IAP.
- A common entrypoint to API routines is provided for interfacing with end user applications.

- Functions are performed by seting specific registers for specific operation and performing a call to common entry point.(like subroutine call).
- After subroutine call control return to end user code.

<u>iv.)Use of factory programmed chip</u>

- Method of embedding the firmware into target processor at the time of chip fabrication.
- Once firmware design is over and operational stability is achieved,firmware files can be sent to chip fabrication to embed to code memory.
- Convenient for mass production.
- Reduces product development time.

Q.16.What are different firmware embedding techniques for non OS based embedded system.

OR

Explain Out of circuit programming

- It is performed outside the target board.
- The processor or memory chip into which the firmware needs to embedded is taken out of the target board and it is programmed with the help of programming device.
- The programming device is dedicated unit when contains the necessary hardware circuit to generate the programming signals.
- The programmer contains a ZIF socket with locking pin to hold the device to be programmed.
- The programming device will be under the control of a utility program running on a PC.
- The programmer is interfaced to the PC through RS-232C/USB/Parallel Port Interface.
- The commands to control the programmer are sent from the utility program to the programmer through the interface.

<u>Sequence of Operations for embedding the firmware with a programmer.</u>

1.Connect the programming device to the specified port of PC (USB/COM port/parallel port)
2.Power up the device.(Ensure that the power indication LED is ON)
3.Execute the programming utility on the PC and ensure proper connectivity is established between the PC and programmer. In case of error turn off device power and try connecting it again.
4.Unlock the ZIF socket by turning the lock pin.
5.Insert the device to be programmed into the open socket as per the insert diagram shown on the programmer.
6.Lock the ZIF socket.
7.Select the device name from the list of supported devices.
8.Load the hex file which is to be embedded into the device.
9.Program the device by ‘Program’ option of utility program.
10.Wait till the completion of programming operation.(Till busy LED of programmer is off)
11.Ensure that programming is successful by checking the status LED on the programmer (Usually ‘Green’ for success and ‘Red’ for error condition) or by noticing the feedback from the utility program.
12.Unlock the ZIF socket and take the device out of programmer.

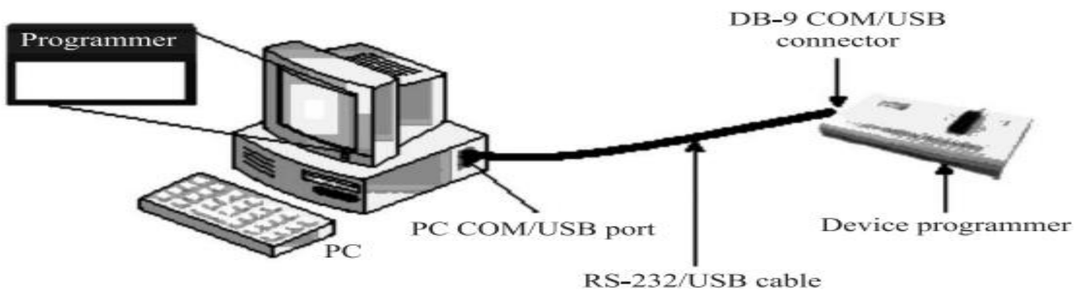


Fig. 12.2 Interfacing of Device Programmer with PC

Drawbacks of out of circuit programming:

- High development time.
  - Whenever the firmware is changed, the chip should be taken out of the board for reprogramming. This tedious and prone to chip damages due to frequent insertion and removal. Better use a socket on the board side to hold the chip till the firmware modifications are over.
- Not suitable for batch production.
- Difficult in upgrading the firmware

  Once product is deployed in the market it is very difficult to upgrade the firmware.

Q.17.What are different firmware embedding techniques for OS based embedded system.

OR

Explain In Systems Programming(ISP)

- OS based embedded system firmware are programmed using ISP(In systems programming).
- OS based embedded system contain special code called Boot loader which takes control of OS.
- In ISP, programming is done within the system.
- The firmware is embedded into the target device without removing it from the target board.
- It is the flexible and easy way of firmware embedding.
- The only pre-requisite is that the device must have an ISP support.
- A part from target board, PC, ISP utility and ISP cable, no other additional hardware is required for ISP.
- Chips supporting ISP generate programming signals using suppl voltage.
- In order to perform ISP operations the target device should be powered up in a special 'ISP mode'.
- ISP mode allows the device to communicate with an external host through a serial interface, such as a PC or terminal.
- Key player behind ISP is factory programmed memory(ROM) called Boot ROM.
- The device receives commands and data from the host, erases and reprograms code memory according to the received command.
- Once the operations are completed, the device is re-configured so that it will operate normally.

<u>In System Programming with SPI Protocol</u>

- *Devices with SPI In System Programming support contains a built-in interface and a on-chip EEPROM or FLASH memory is programmed through this interface.

- *The primary I/O lines involved in SPI-In System Programming are listed below,
    a. MOSI – Master Out Slave In
    b. MISO – Master In Slave Out
    c. SCK – System Clock
    d. RST – Reset of Target Device
    e. GND – Ground of Target Device.
        - PC act as master and target device act as slave in ISP.
        - Program data is sent to MOSI pin of target device and acknowledgement originated from MISO PIN.
        - SCK pin act as clock for data transfer.
        - Target device works under a supply voltage of 5Volt,so connect lines of target device with parallel port of PC.

<u>* Steps for the operation or power up sequence for In System Programming for Atmel's AT89S series microcontroller:</u>

1.)Apply supply voltage between VCC and GND pins of target chip.
2.)Set RST pin to "HIGH" state.
3.)If a crystal is not connected across pins XTAL1 and XTAL2, apply a 3 MHz to 24MHz clock to XTAL1 pin and wait for at least 10 milliseconds.
4.)Enable serial programming by sending the Programming Enable serial instruction to pin MOSI/P1.5. The frequency of the shift clock supplied at pin SCK/P1.7 needs to be less than the CPU clock at XTAL1 divided by 40.
5.)The Code or Data array is programmed one byte at a time supplying the address and data together with the appropriate Write instruction. The selected memory location is first erased before the new data is written. The write cycle is self – timed and typically takes less than 2.5 ms at 5V.
6.)Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO/P1.6.
7.)After successfully programming the device, set RST pin low or turn off the chip power supply and turn it ON to commence the normal operation.

Q.18.What is EDLC?

- EDLC is Embedded Product Development Life Cycle
- It is an Analysis – Design – Implementation based problem solving approach for embedded systems development.
- Analysis involves understanding what product needs to be developed Design involves what approach to be used to build the product Implementation is developing the product by realizing the design.

Q.19.Why do we need EDLC?

- EDLC is essential for understanding the scope and complexity of the work involved in embedded systems development
- It can be used in any developing any embedded product
- EDLC defines the interaction and activities among various groups of a product development phase including project management, system design ,System testing, Release management and quality assurance.

Q.20.What are the objectives of EDLC

- The ultimate aim of any embedded product in a commercial production setup is to produce Marginal benefit
- Marginal is usually expressed in terms of Return On Investment
- The investment for product development includes initial investment, manpower, infrastructure investment etc.
- EDLC has **three primary objectives** are:

a.Ensuring that high quality products are delivered to user
b.Risk minimization and defect prevention in product development through project management
c.Maximize productivity

a.Ensuring that high quality products are delivered to user:

- The primary definition of quality in any embedded product development is return on investment achieved by the product. The expenses incurred for developing the product the product are:-
- Initial investment, Developer recruiting Training
- Infrastructure requirement related
- In order to survive in market, quality is very important factor to be taken care of while developing the product.
- Qualitative attributes depends on the budget of the product so budget allocation is very important.
- Budget allocation might have done after studying the market, trends & requirements of product, competition.
- EDLC must ensure that the development of the product has taken account of all the quality attributes of the embedded system.

b.Risk minimization and defect prevention in product development through project management:

- There are projects in embedded product development which requires loose or tight management.
- If the development is a simple one a senior developer itself can take the charge of a management activity and there is no need for a skilled project manager to look after this with dedicated effort. But there will be an overall supervision form the skilled project management team for ensuring that the development process is going in the right direction.
- Project which are complex and requires timeliness should have a dedicated and a skilled project management part and hence they are said to be tightly bounded to project management.
- Project management -
  -Adds an extra cost on budget
  -But essential for ensuring the development process is going in right direction

- Project management is required for Predictability
- Analyze the time to finish the product (PDS = no of person days).this is estimated on the basis of past experience

Co-ordination

Resources (developers) needed to do the job Risk management

- Backup of resources to overcome critical situation
- Ensuring defective product is not developed
- Project management is essential for ' predictability co-ordination and risk minimization
- Resource allocation is critical and it is having a direct impact on investment
- Microsoft @ Project Tool is a typical example for CASE tools for project management

c.Maximize productivity:

- Productivity is a Measure of efficiency as well as ROI This productivity measurement is based on total manpower efficiency
- Different ways to improve the productivity are
- Saving the manpower
- X members – X period
- X/2 members – X period(the productivity is doubled)
- Use of automated tools where ever is required
- Re-usable effort – work which has been done for the previous product can be used if similarities present b/w previous and present product.
- Use of resources with specific set of skills which exactly matches the requirements of the product, which reduces the time in training the resource.
- Recruiting people with desired skill set for current project is another option,this is worth only if you expect to have more work on the near future on the same technology or skill sets.
- EDLC should take all this aspects into consideration to provide maximum productivity.

Q.21.What are the different phases of EDLC

OR

Give the function of each phase of embedded product development lifecycle.

The following figure depicts the different phases in EDLC:

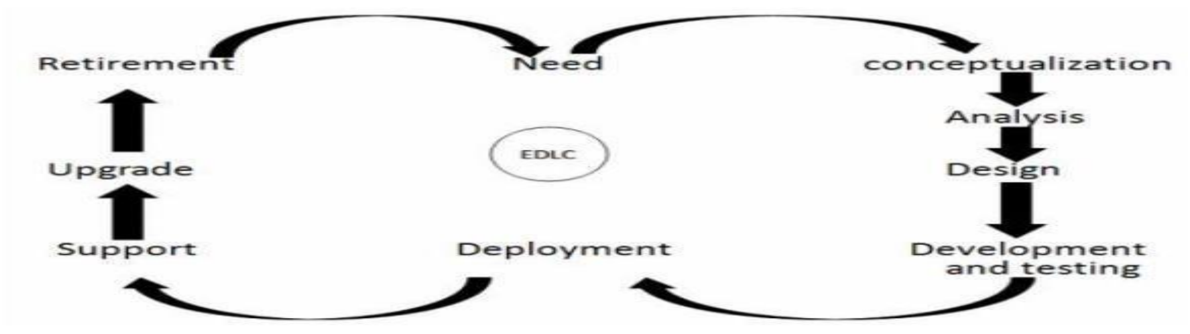


Figure : Phases of EDLC

- A life cycle of product development is commonly referred as the “model”.
- In real product development the phases given in this model may vary and can have sub models or model contain only certain important phases of classic model.
- The no of phases involved in EDLC model depends on the complexity of the product.

1. NEED

- The embedded product involves the output of a need
- The need may come from an individual or from the public or from a company.
- ‘Need’ should be articulated to initiate the Development Life Cycle;
- A ‘Concept Proposal’ is prepared which is reviewed by the senior management for approval.
- Need can be visualized in any one of the following three needs:

  a.***New or Custom Product Development.-***The need for a product that doesn’t exist in the market or a product which act as a competitor to an existing product in the current market will need to the development of a completely new product. Example of an embedded product which act as a competitor in market is Mobile handset

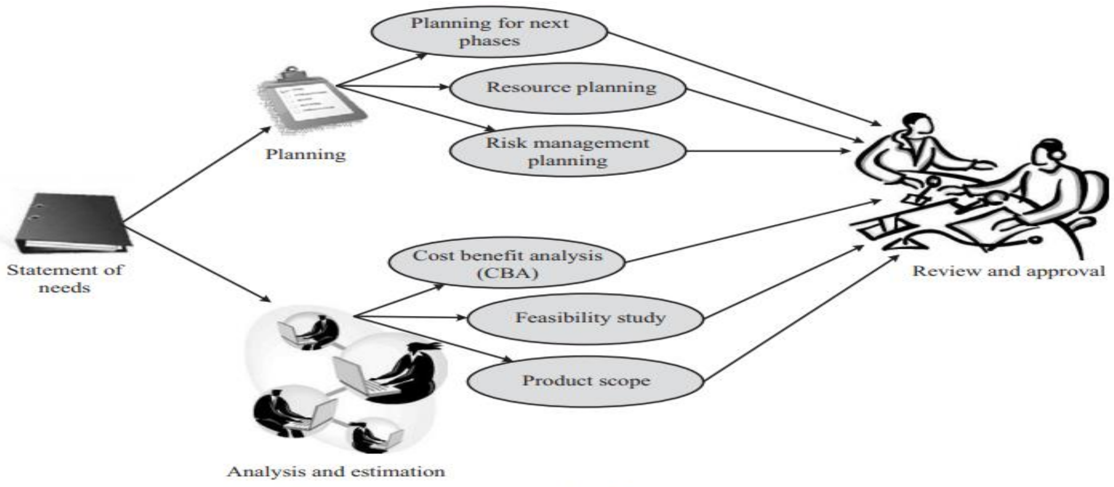
  b. Product Re-engineering.-The embedded product market is competitive and dynamic. Re- engineering an existing product comes as a result of following needs.

Change in Business requirement User interface Enhancements
Technology Upgrades

c.Product Maintenance.-This need deals with providing technical support to the end user for an existing product in the market. The maintenance request may comes as the result of product nonfunctioning or failure. Product maintanence is classified into 2

- ✓ **Corrective maintanence**:deals with making corrective actions following a failure or non functioning
- ✓ **Preventive maintanence** is used to avoid the failure or non functioning of the product.

2. CONCEPTUALISATION



- ✓ Fig. 15.2 Various activities involved in Conceptualisation Phase

Is the product concept development phase and it begins immediately after the concept proposal is formally approved.

- ✓ These phases define the scope of concept,performs cost benefit analysis(CBA) and feasibility study and prepare project management and risk management plans.
- ✓ The conceptualization involves two types of activities, planning activity and analysis and study activity.
  - ❖ **Feasibility Study :** Examine the need and suggest possible solutions.It analyses technical as well as financial feasibility.
  - ❖ **Cost Benefit Analysis (CBA):** Revealing and assessing the total development cost and profit expected from the product. Marginal profit is expressed in terms of money.
  - ❖ **Product Scope:** Deals with what is scope(functionalities need to be considered) and what is not in scope. Should be properly documented for future reference.
  - ❖ **Planning activities** : It covers various plans for product development,like plan and schedule for executing next phases.

Resource planning: How many resources should work on project.
Risk management plan:Kinds of risk involved and various mitigation plans.

- At the end of conceptualization phase,report is submitted to the client for approval.

3.ANALYSIS

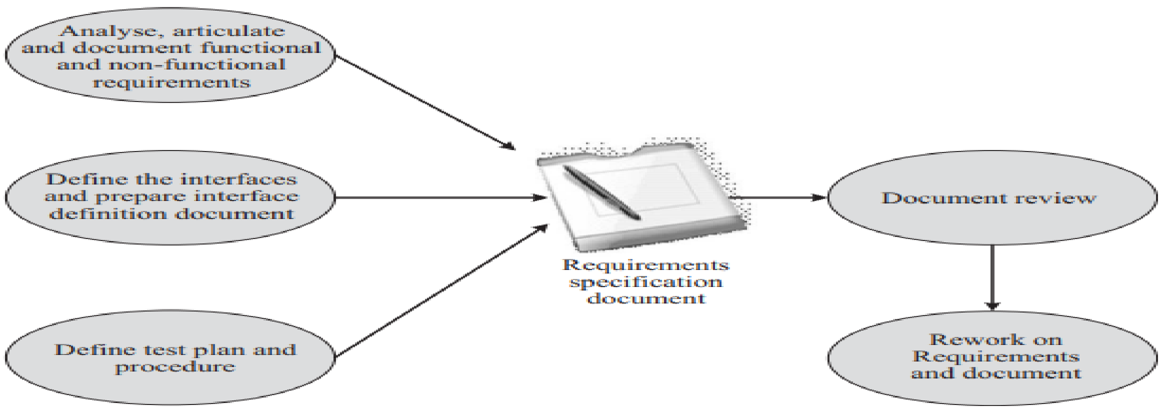

Analyse, articulate and document functional and non-functional requirements
Define the interfaces and prepare interface definition document
Define test plan and procedure
Requirements specification document
Document review
Rework on Requirements and document

Fig. 15.3 Various activities involved in Requirements Analysis Phase

Requirement analysis is done to develop a detailed functional model of product.

- Product is defined in detail with respect to inputs,processes,outputs and interface at functional level.
- Used for determining what functions mus be performed by the product.
- Various activities during this phase are:
  - ❖ Analysis and documentation:It analyses the functional as well as quality attributes of the product,defines functional and data requirements,connects functional and data requirements. Requirements that need to be addressed during this phase are
    1.Functional Capabilities like performance
    2.Operational and non-operational quality attribute
    3.Product external interface requirements
    4.User manualsData requirements
    5.Operational requirements
    6.Maintenance requirements
    7.General assumptions.
  - ❖ Interface definition and documentation: This activity should clearly analyse on the physical interfaces as well as data exchange through this interfaces and should document it.
  - ❖ Defining Test Plan and Procedures: Identifies various tests to be performed and what all to be covered in testing the product.

❖ Various types of testing to be performed in product development are:

Unit testing – Testing Individual modules
Integration testing – Testing a group of modules for required functionality
System testing- Testing functional aspects or functional requirements of the product after integration. Different tests are

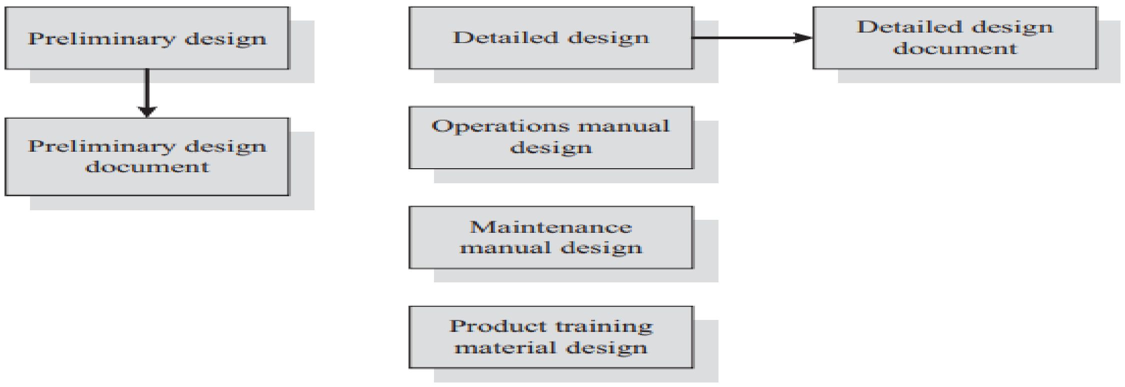
Usability testing: test the usability of the product
Load testing :test the behavior of the product under different loading conditions
Security testing: test the security aspects of the product Scalability testing: test the scalability aspects of the product Sanity testing: Superficial testing is performed to ensure
that the product is functioning properly
Smoke testing: ensures the crucial requirement of the product are functioning properly
Performance testing: test the performance aspects of the product after integration

Endurance testing:durability test of the product

User acceptance testing- Testing the product to meet the end user requirements. At the end all requirements are put in a template suggested by a standard process model.

4.DESIGN

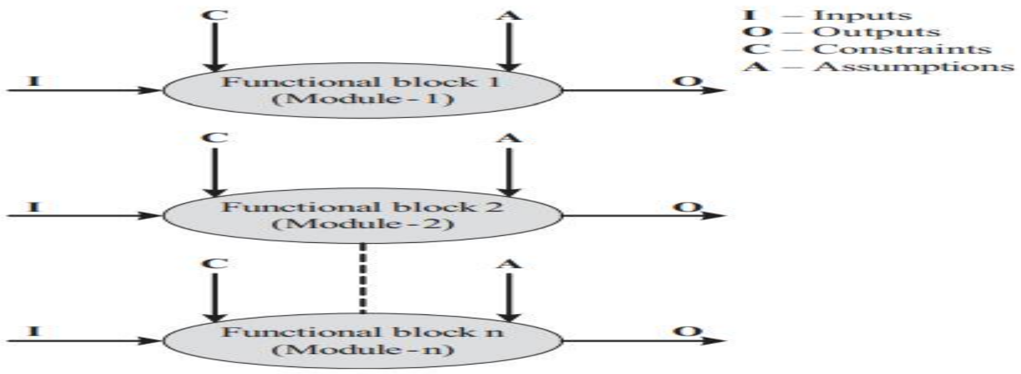
■ Various activities involved in design phase



Preliminary design
Preliminary design document
Detailed design
Detailed design document
Operations manual design
Maintenance manual design
Product training material design

Fig. 15.4 Various Activities involved in Design Phase

- The design phase identifies application environment and creates an overall architecture for the product.
- It focuses on how the req uired functionalities can be delivered to the product.
- It starts with the Preliminary Design. It establishes the top level architecture for the product. On completion it resembles a 'black box' that defines only the inputs and outputs. The final product is called Preliminary Design Document (PDD).
- Once the PDD is accepted by the End User the next task is to create the
- 'Detailed Design'.
- It encompasses the Operations manual design, Maintenance Manual Design and Product Training material Design and is together called the 'Detailed Design Document.
- Detailed design generates detailed architecture,identify and list various components of each functional block,interconnection among various functional blocks,control algorithm requirements etc..



I — Inputs
O — Outputs
C — Constraints
A — Assumptions
C
A
I
O
Functional block 1 (Module - 1)
Functional block 2 (Module - 2)
Functional block n (Module - n)

Fig. 15.5 Preliminary Design Illustration

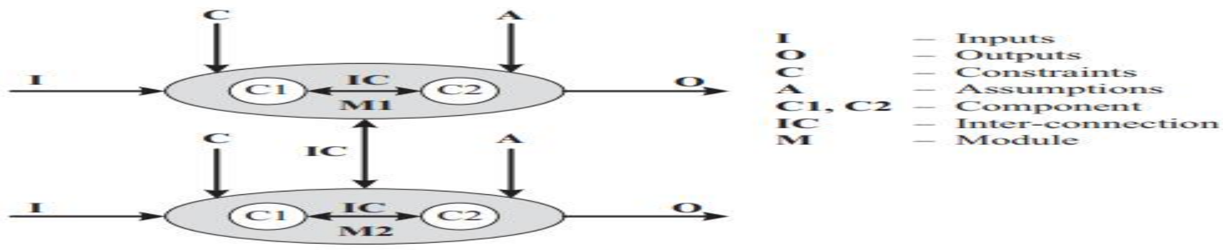


Fig. 15.6 Detailed Design Illustration

5. DEVELOPMENT AND TESTING

- It transforms the design into a desirable product.
- Detailed specifications are translated into hardware and firmware.
- Development activities can be partitioned into hardware development,embedded firmware development and product enclosure development.
- Embedded hardware development refers to development of component placement platform like PCB and its fabrication.
- Embedded firmware development is performed using firmware development tools.
- Mechanical enclosure design shows the look and feel of the product.
- Other activities are preparation of test case procedures,preparation of test files,deployment,operatios and maintanence manual.
- Testing phase can be divided into independent testing of firmware and hardware, testing after integrating firmware and hardware,testing whole system on the basis of functionality and non functionality basis,testing of product against all acceptance criteria.

6. DEPLOYMENT

❖ Deployment is the process of launching the first fully functional model of the product in the market. It is also known as First Customer Shipping (FCS).

❖ Tasks performed during this phase are:

1.Notification of Product Deployment:

- Launching ceremony details should be communicated to stakeholders.The notification can be sent through e- mail,medial etc.
- Tasks performed here include:

  Deployment schedule
  Brief description about the product
  Targeted end user
  Extra features supported
  Product support information

2.Execution of training plan : Proper training should be given to the end user top get them acquainted with the new product.
3.Product installation :Install the product as per the installation document to ensure that it is fully functional.
4.Product post Implementation Review: After the product launch, a post implementation review is done to test the success of the product.

7. SUPPORT

- The support phase deals with the operational and maintenance of the product in the production environment
- Bugs in the product may be observed and reported.
- The support phase ensures that the product meets the user needs and it continues functioning in the production environment.
- Activities involved under support are

  1.Setting up of a dedicated support wing: Involves providing 24 x 7 supports for the product after it is launched.
  2.Identify Bugs and Areas of Improvement: Identify bugs and take measures to eliminate them.

8. UPGRADES

- Deals with the development of upgrades (new versions) for the product which is already present in the market.
- Product upgrade results as an output of major bug fixes.

- During the upgrade phase the system is subject to design modification to fix the major bugs reported.
- Some bugs may be easily fixed by modifying firmware and hardware.
- Firmware modification consist of a version number which starts with version 1.0 and after bug fix,version is changed accordingl.
- Version numbering is good for backward traceability.

9. RETIREMENT/DISPOSAL

- The retirement/disposal of the product is a gradual process.
- This phase is the final phase in a product development life cycle where the product is declared as discontinued from the market.
- The disposal of a product is essential due to the following reasons
  -Rapid technology advancement
  -Increased user needs.

Q.22.Explain the need of product reengineering in embedded product development.

- Product reengineering is major phase of the NEED phase in the embedded product development lifecycle.
- Product market is dynamic and competitive.
- Product developer should be aware of market trends,technology changes and should constantly improve existing product by adding new technologies.
- Reengineering is the process of making changes in an existing product design and launching it as a new version known as product upgrade.
- Rengineering occurs due to changes in business requirements,user interface enhancements and technology upgrades.

Q.23.What are the different EDLC approaches

OR

What are the different methods in modeling EDLC

- Modeling EDLC refers to the interconnection of various phases involved in the development of an embedded product.
- Following are some of the different types of approaches that can be used to model embedded products.

1.Waterfall or Linear Model
2.Iterative/ Incremental or Fountain Model
3.Prototyping Model/Evolutionary Model 4.Spiral Model
4.Spiral Model

Q.24.Explain various types of testing in embedded product development

Various types of testing to be performed in product development are:

1 .Unit testing – Testing Individual modules

2 .Integration testing – Testing a group of modules for required functionalities.

3 .System testing- Testing functional aspects or functional requirements of the product after integration.

Different tests are

Usability testing: test the usability of the product

Load testing :test the behavior of the product under different loading conditions

Security testing: test the security aspects of the product Scalability testing: test the scalability aspects of the product

Sanity testing: Superficial testing is performed to ensure that the product is functioning properly

Smoke testing: ensures the crucial requirement of the product are functioning properly
Performance testing: test the performance aspects of the product after integration
Endurance testing:durability test of the product
4 .User acceptance testing- Testing the product to meet the end user requirements . At the end all requirements are put in a template suggested by a standard process model.

Q.25.Explain linear or waterfall model

OR

Explain any one EDLC approach
Also explain the advantages and disadvantages of waterfall model

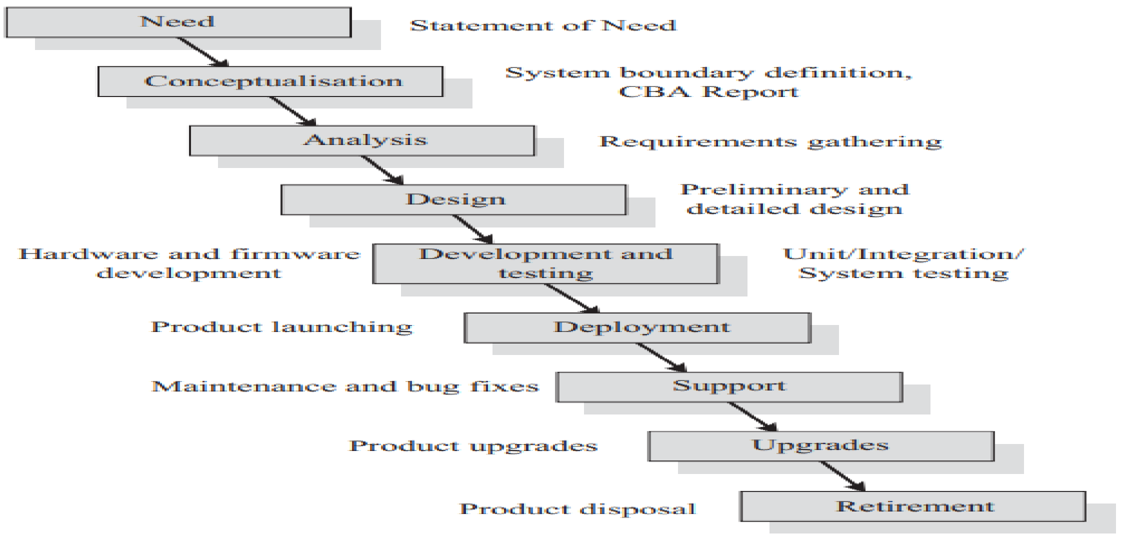


Fig. 15.7 Linear (Waterfall) EDLC Model

- Linear or waterfall model is the one adopted in most of the olden systems.
- In this approach each phase of EDLC (Embedded Development Product Lifecycle) is executed in sequence.
- It establishes analysis and design with highly structured development phases.
- The execution flow is unidirectional.
- The output of one phase serves as the input of the next phase
- All activities involved in each phase are well planned so that what should be done in the next phase and how it can be done.
- The feedback of each phase is available only after they are executed.
- It implements extensive review systems To ensure the process flow is going in the right direction.
- One significant feature of this model is that even if you identify bugs in the current design the development process proceeds with the design.
- The fixes for the bug are postponed till the support phase.

Advantages
-Product development is rich in terms of: Documentation
-Easy project management
-Good control over cost & Schedule

Drawbacks
-It assumes all the analysis can be done without doing any design or implementation The risk analysis is performed only once.
-The working product is available only at the end of the development phase
-Bug fixes and correction are performed only at the maintenance/support phase of the life cycle.

Q.26.Explain iterative or incremental or fountain model

OR

Explain any one EDLC model
Also list the advantages and disadvantages of the iterative model

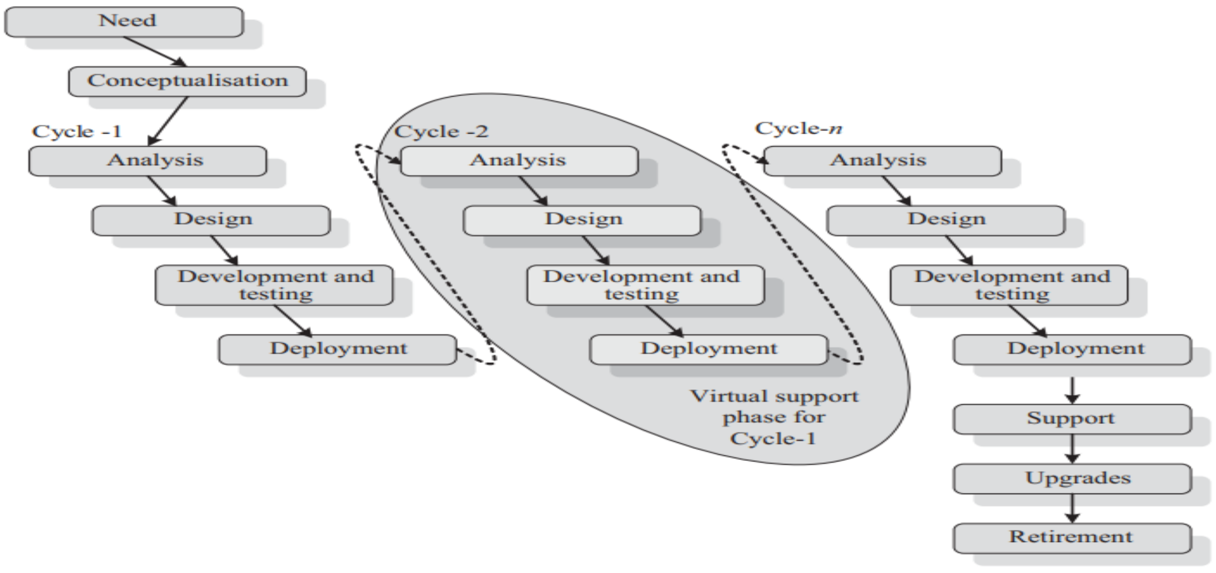


Fig. 15.8 Iterative/Incremental/Fountain EDLC Model

- Iterative and Incremental development is at the heart of a cyclic software development process developed in response to the weaknesses of the waterfall model.
- The iterative model is the repetitive process in which the Waterfall model is repeated over and over to correct the ambiguities observed in each iteration.
- The core set of functions for each group is identified in the first cycle, it is then built, deployed and release. This release is called as the first release.
- Bug fixes and modification for first cycle carried out in second cycle.
- Process is repeated until all functionalities are implemented meeting the requirements.

Advantages

- Good development cycle feedback at each function/feature implementation
- Data can be used as reference for similar product development in future.
- More responsive to changing user needs.
- Provides working product model with at least minimum features at the first cycle.
- Minimized Risk
- Project management and testing is much simpler compared to linear model.
- Product development can be stopped at any stage with a bare minimum working product.

Disadvantages

- Extensive review requirement each cycle.
- Impact on operations due to new releases.
- Training requirement for each new deployment at the end of each development cycle.
- Structured and well documented interface definition across modules to accommodate changes.

Q.27.Explain prototype or evolutionary model

OR

Explain any one EDLC model

Also list the advantages and disadvantages of prototype model

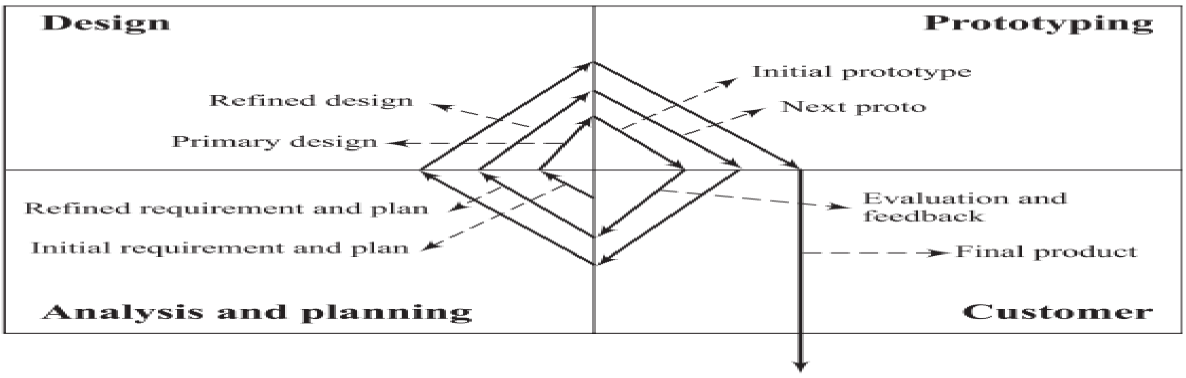


Fig. 15.9 Proto-typing/Evolutionary EDLC Model

- It is similar to iterative model and the product is developed in multiple cycles.
- The only difference is that, Prototyping model produces a refined prototype of the product at the end of each cycle instead of functionality/feature addition in each cycle as performed by the iterative model.
- There won't be any commercial deployment of the prototype of the product at each cycle's end.
- The shortcomings of the proto-model after each cycle are evaluated and it is fixed in the next cycle.
- After the initial requirement analysis, the design for the first prototype is made, the development process is started.
- On finishing the prototype, it is sent to the customer for evaluation.
- The customer evaluates the product for the set of requirements and gives his/her feedback to the developer in terms of shortcomings and improvements needed.
- The developer refines the product according to the customer's exact expectation and repeats the proto development process.
- After a finite number of iterations, the final product is delivered to the customer and launches in the market/operational environment
- In this approach the product undergoes significant evolution as a result of periodic shuttling of product information between the customer and developer
- The prototyping model follows the approach: Requirement definition
- Proto-type development Proto-type evaluation Requirements refining

Advantages

-Easy to incorporate new requirements at an stage.
-Takes user feedback
-Risk is spread across each development cycle.
-Development is well under control.

Disadvantages

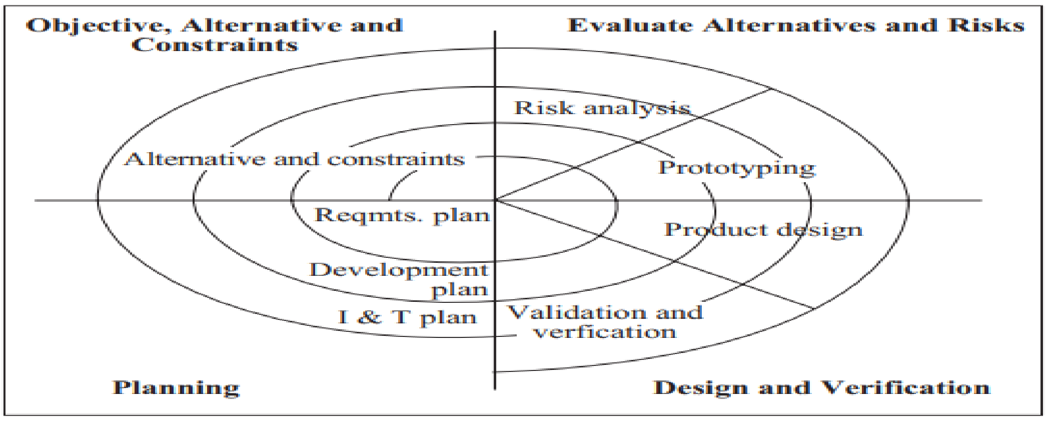
-Deviations from expected cost and schedule due to requirements refinement
-Increased project management
-Minimal documentation on each prototype may create problem in backward prototype traceability.
-Increased configuration management activities.

Q.28.Explain spiral model

OR

Explain any one EDLC model

Also list advantages and disadvantages of spiral model



Objective, Alternative and Constraints
Evaluate Alternatives and Risks
Risk analysis
Alternative and constraints
Prototyping
Reqmts. plan
Product design
Development plan
I & T plan
Validation and verfication
Planning
Design and Verification

Fig. 15.10 Spiral Model

- Spiral model is developed by Barry Boehm in 1988.
- The Product development starts with project definition and traverse through all phases of EDLC(Embedded Product Development Life Cycle).
- It is a combines the concept of Linear Model and iterative nature of Prototyping Model
- The activities involved are:

1.Determine objectives, alternatives, constraints
2.Evaluate alternatives, identify and resolve risks
3.Develop and test
4.Plan

Requirement:

- This process is focused specifically on embedded software, to understand the nature of the software to be build and what are the requirement for the software.
- And the requirement for both the system & the software is documented & viewed to customer.

Analysis:

- Analysis is performed to develop a detailed functional module under consideration.
- The product is defined in detailed with respect to the input, processing & output.
- This phase emphasis on determining 'what function must be performed by the product' & how to perform those function.

Design:

- Product design deals with the entire design of the product taking the requirement into consideration.
- The design phase translates requirement into representation.

Implementation:

- In this process the launching of first fully functional model of the product in the market is done or handing over the model to an end user/client
- In this product modifications are implemented & product is made operational in production environment.

<u>IMPORTANT QUESTION AS PER MODEL QUESTION PAPER</u>

1..An embedded product under consideration is very complex in nature and there is a ossibility for change in requirements of the product. Also the risk associated with the development of this

product is very high. Which is the best suited life cycle method to handle this product development? Justify your answer.
Explain about spiral model

Ref Qn.28.

2..Explain the similarities and differences between iterative and incremental life cycle models.

Ref Qn.26.

3..When do you prefer a super-loop-based firmware design approach over an RTOS based approach? What are the limitations of the super-loop based approach and how do you overcome them?

Ref Qn.3,4,5.

4..Briefly explain the different approaches used for embedding firmware into the hardware of an embedded device.

Ref Qn.13,14,15,17.