UNIT 5

EMBEDDED SYSTEM DEVELOPMENT

PART-A
SHORT QUESTIONS WITH SOLUTIONS

Q1. Define IDE.

Ans:

Integralid Development Envioronment (IDE) is a software package which initegrates the environment to develop and debug the target processor specific embedded firmware.

Q2. List the different files generated during the cross compilation. (Model Paper-I, Q1(e) | April-18, Set-1, Q1(e))

(or)

What are types of files generated on cross-compilation?

Ans: April/May-17, Set-1, Q1(e)

The various files generated during the cross compilation process are,

1. List file (.lst)
2. Hex File (.hex)
3. Pre-processor output file
4. Map file
5. Object file (.obj).

Q3. Explain the various details stored in an object file generated during the cross compilation.

Ans: April-18, Set-2, Q1(e)

The various details stored in an object file generated during the cross complition are as follows,

1. Public symbol names
2. External symbol references
3. Reserved memory for global variables
4. Library files
5. Debugging information.

Q4. What is absolute object file?

Ans: April-18, Set-3, Q1(d)

The object file that is created after assigning it to an absolute or specific address by linker and locator is referred as an absolute object file or module. It does not hold relocatable codes or data. This file generates the hex files that are embedded into the processor/controller code memory.

Q5. What is a decompiler?

(Model Paper-III, Q1(e) | April-18, Set-4, Q1(e))

Ans:

Disassembler is a program designed for converting machine codes into target processor which is nothing but assembly instructions and codes. This implies that machine language codes are converted into assembly language code.

Q6. List the features of simulator based debugging.

Ans:

The features of simulator based debugging are,

1. It does not possess real time behaviour.
2. It does not possess advanced I/O support features as it is a conventional one.
3. It does not need real target system.
4. It is purely a software oriented system.

Q7. Define hardware/software co-simulator.

Dec.-13, Set-1, Q7(a)

Ans:

Hardware/software co-simulator is the simulator that integrates software and hardware simulators of the system. This type of simulator is adapted for the embedded system that makes use of both general purpose and single purpose processors because the best simulation level for general purpose processor is instruction level, while for the special purpose processor is system level.

Thus, it is preferred to perform software simulation for general purpose processors while hardware simulation for single purpose processors. Hence, we use an integration of these two simulators together called hardware/software co-simulator. The functioning of co-simulator is faster than HDL simulator, but the drawback of co-simulator is that it becomes slower, when the communication between the two processors becomes too frequent. Hence, an ideal co-simulator should integrate the two simulators effectively and also minimize the communication between the two simulators and thus maintain good speed.

Q8. What is a key method for speeding up such simulator?

Ans:

Dec.-13, Set-1, Q7(b)

A co-simulator performs both hardware and software simulations for the two processors (general purpose and single purpose) of an integrated embedded system. The main reason for reduction in the simulation speed is due to the communication between the processors and sharing of a common memory.

The possible technique is to minimize this communication is by allotting a memory section to each simulator (i.e., ISS and HDL simulator). Hence each simulator uses its own memory without interfering with the other, when each simulator is provided its own memory it need not interact with the other to get the permit (to use the memory) and can continue with its process. This technique improves the speed of the simulator nearly by a factor of 100 or even more.

Q9. Define debugging and list its types.

Ans:

Debugging is the process of examining firmware implementation and supervising the target processor's registers and memory, when the firmware is executing and verifying the signals from different buses present in embedded applications.

Debugging is mainly divided into two types,

1. Hardware debugging
2. Firmware debugging.

Q10. Differentiate between linker and locator.

Ans:

The differences between linker and locator are mentioned below:

	Linker		Locator
1.	The linker is a program that creates the linked file, saves it on the disk and allocates the memory address to it.	1.	The locator is a program that uses the copy of linked file and places it on EEROM or flash for execution.
2.	Linker addresses are the relative addresses that are used by host system processor. Further, these addresses need to be reallocated by the loader.	2.	Locator addresses are the addresses that are used by target system processor. Further, these addresses need not be reallocated.
3.	It makes use of relative addresses and actual addresses that are allocated to it by the OS.	3.	It makes use of addresses which are permanent because the file records are burned into the system.
4.	It creates the output file whose file format is according to the file system specified by the disk.	4.	It creates the output file whose file format is specified by Motorola-S or Intel hex or any other format.

Q11. Explain the role of IDE for embedded software development. (Model Paper-II, Q1(e) | April-18, Set-4, Q6(a))

(or)

Explain the role of Integrated Development Environment for embedded software development.

Ans: (April/May-17, Set-3, Q6(a) | Nov./Dec.-15, Set-1, Q7(a) | Nov./Dec.-14, Set-4, Q7(a))

Integrated Development Environment (IDE) is a software package consists of text editor, cross-compiler linker and a debugger. In embedded software development, it integrates environment to develop and debug the target processor specific embedded firmware. Some of the IDEs are also used as interface to target board emulators. Target processor's/controller's flash memory programmer.

IDE may be command line based or GUI based command line based IDEs are also supported by GUI.

Example

TURBO CIDE in C + + for x 86 plat form.

GUI based IDEs are used to provide visual development environment with each action of mouse click. Hence, they are also called visual IDEs.

Example

Microsoft® visual studia for Visual C + +, Net beans, Eclipse.

In embedded firmware development, IDE is supplied as open source (code warrier by metroworks for ARM processors) or third party vendor (keil μ vision 3 from keil software) or target processor controller manufacturer (MPLAB by microchip for PIC family of micro controllers).

Since a single IDE is required for a specific family of controllers or processors, special IDE is built around eclipse open source IDE for embedded system development.

PART-B
ESSAY QUESTIONS WITH SOLUTIONS

5.1 THE INTEGRATED DEVELOPMENT ENVIRONMENT

Q12. Explain the steps in creating new project. Also discuss the procedure for selecting target CPU vendor for Keil μ vision3 IDE.

Ans:

Creation of New Project

Steps to create a new project in Keil μ vision 3 are given as follows,

1. Click on project tab on the menu bar and select 'new project' from the list of options. A dialog box appears with 'create new project.
2. Enter the name of the project in the 'file name' section with the extension .μv2
3. Select a directory, where the project is to be saved.
4. Click on save button as shown in figure (1).

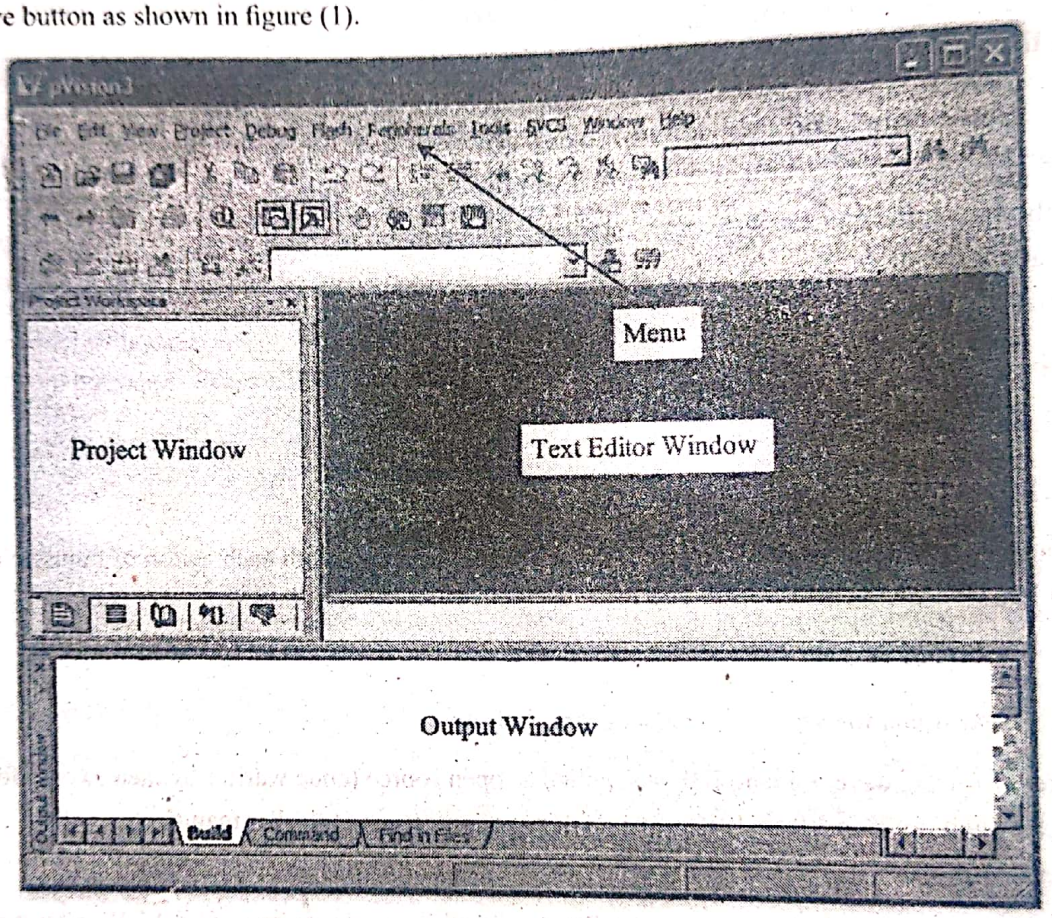


Figure (1): Creating a New Project

Selection of Target CPU Vendor for Keil μ Vision3 IDE

After saving the project, a device selection dialog box appears on the screen. It shows a list of all the vendors for 8051 family microcontroller which the IDE supports. Targeted CPU vendor for Keil μ vision3 IDE can be selected in the below manner,

1. Select the manufacturer of the chip from the available list of manufacturers shown in the database column (For example, Atmel) as shown in figure (2).
2. A list of supported cpu's by the selected vendor appears under the vendor node. Select the part number of the device which will be used as the target processor for the design. This will show the vendor name, device and tool set in the appropriate fields along with a small description about it in the description column.

3. Click on OK as shown in figure (3).

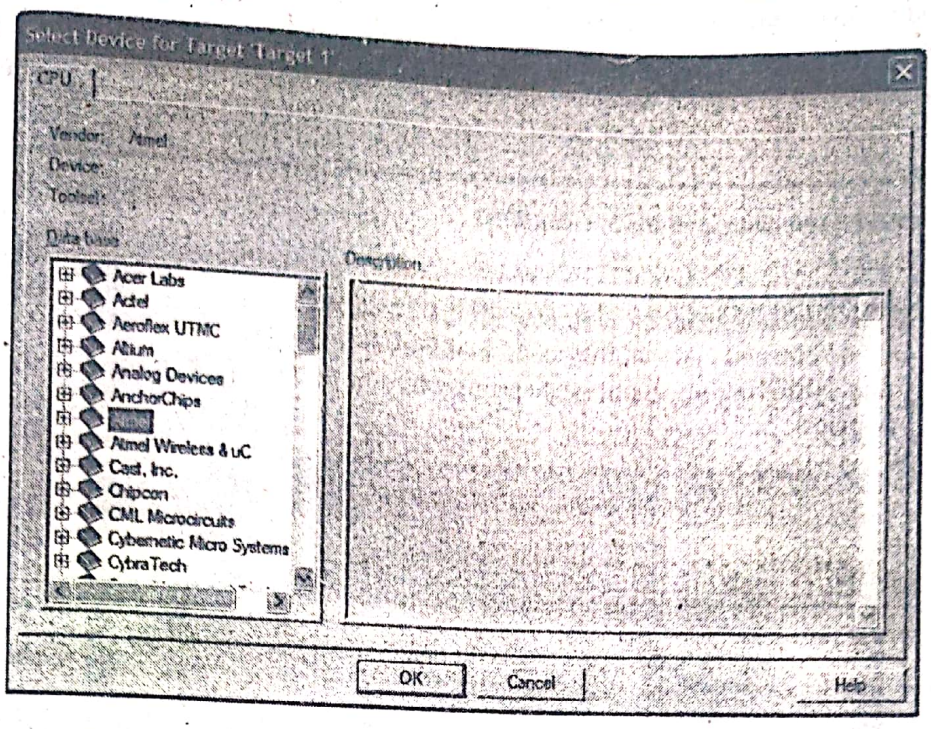


Figure (2): Selecting Target CPU Vendor

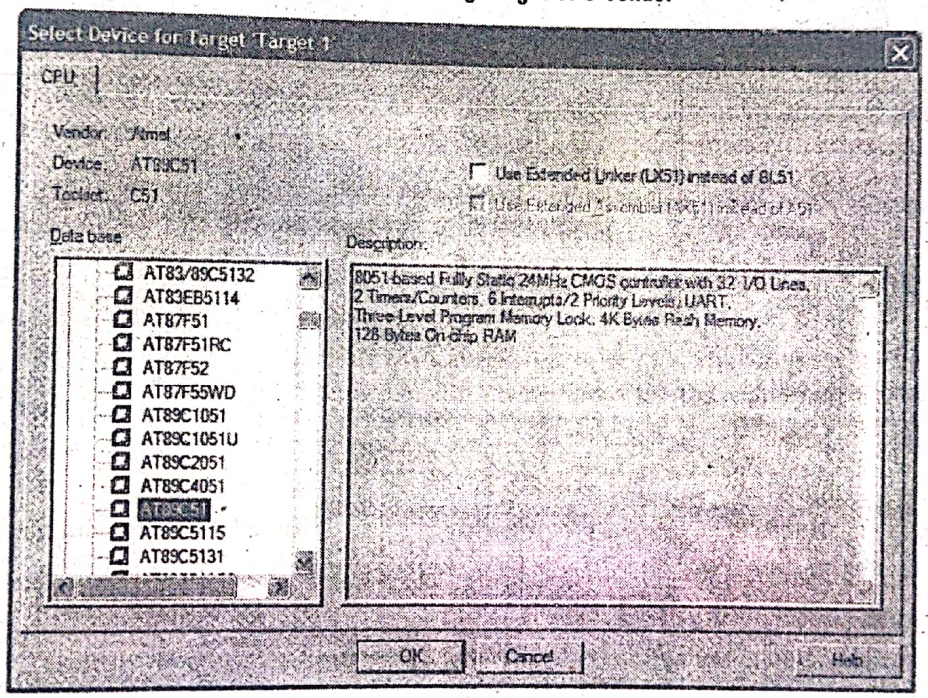


Figure (3): Selecting Part Number of the Target Processor

4. After selecting the target processor, the IDE will add all the necessary start-up code for the firmware automatically. And then, it asks the user whether to add the start code or not as shown in figure (4)

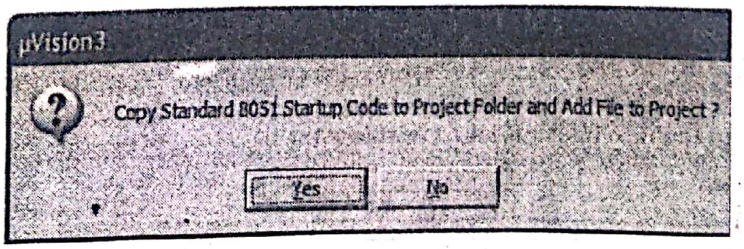


Figure (4): Adding Start-up File to the Project

5. Click 'yes', to add the code, otherwise click 'NO'.

After this, in project window files a group gets created automatically with the name 'target 1' which includes source group with the name source group 1 which in addition contains start-up file, with the name STARTUP.A51 as shown in figure (5).

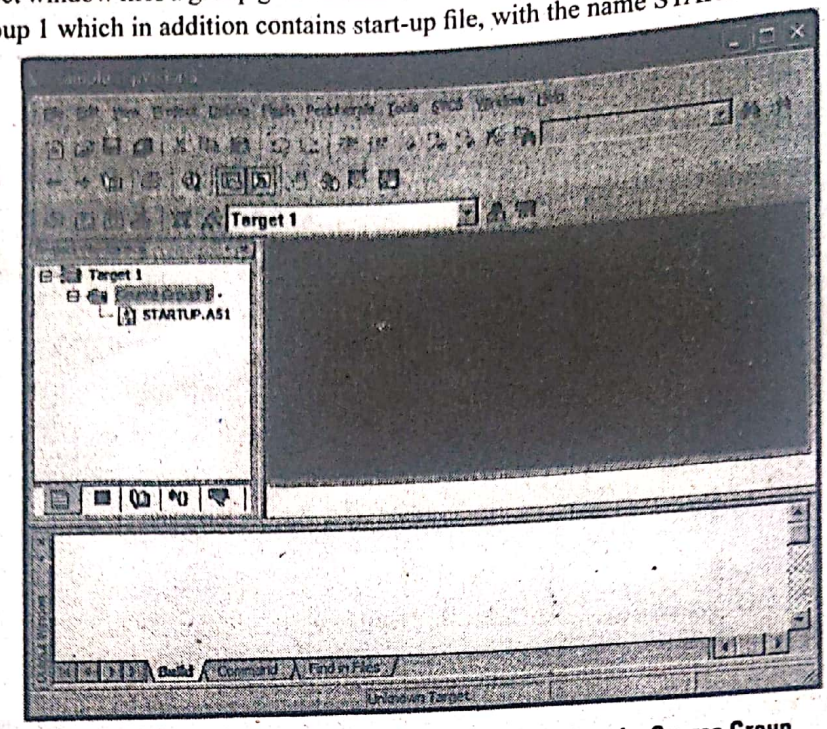


Figure (5): Window showing the Start-up File Added to the Source Group

Q13. List and discuss the tabs available in project window of Keil IDE's.

Ans:

The Keil IDE project window contains several tabs which are given as follows,

1. Files tab
2. Register Tabl (Regs tab)
3. Books tab
4. Functions tab
5. Templates tab.

1. Files Tab

The files tab displays the details of all the files for the ongoing project. It lists all the source and header files in the project workspace as shown in figure (1).

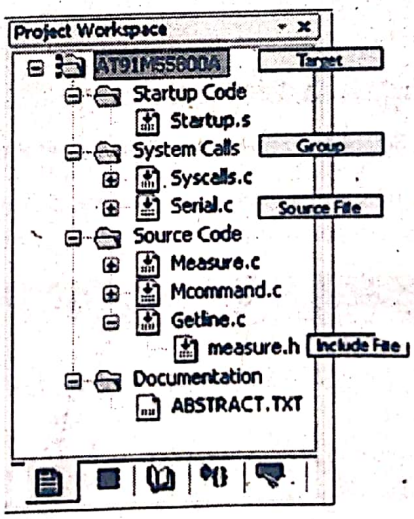


Figure (1): Files Tab

2. **Register Tab ('Regs' Tab)**

The register tab displays all the microcontroller register's and their contents in the project workspace window. When the code is being debugged as shown in figure (2).

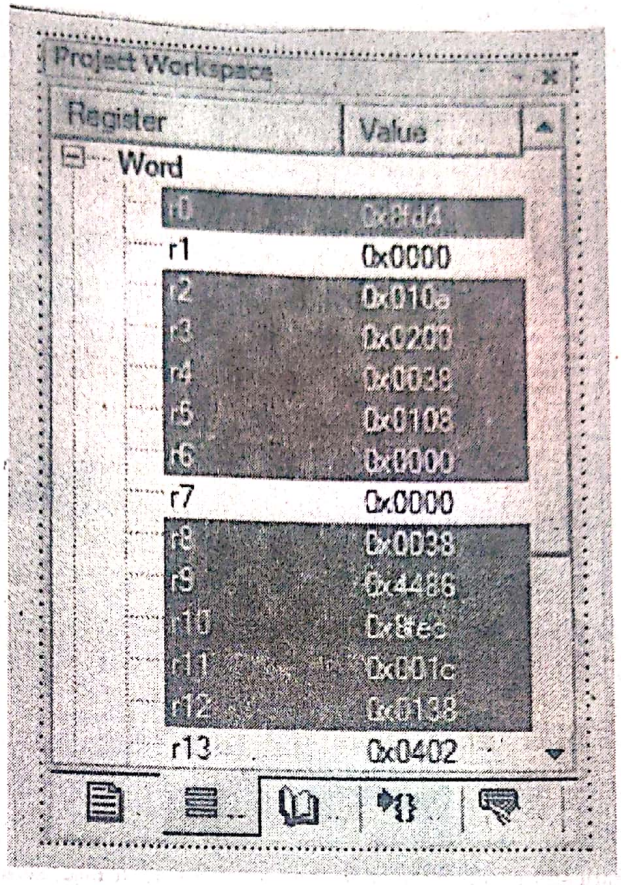


Figure (2): Register Tab

Books Tab

The books tab provides online manuals, documentation for tools and specific microcontroller in order to help the users. Users are allowed to add any additional books of required as shown in figure (3).

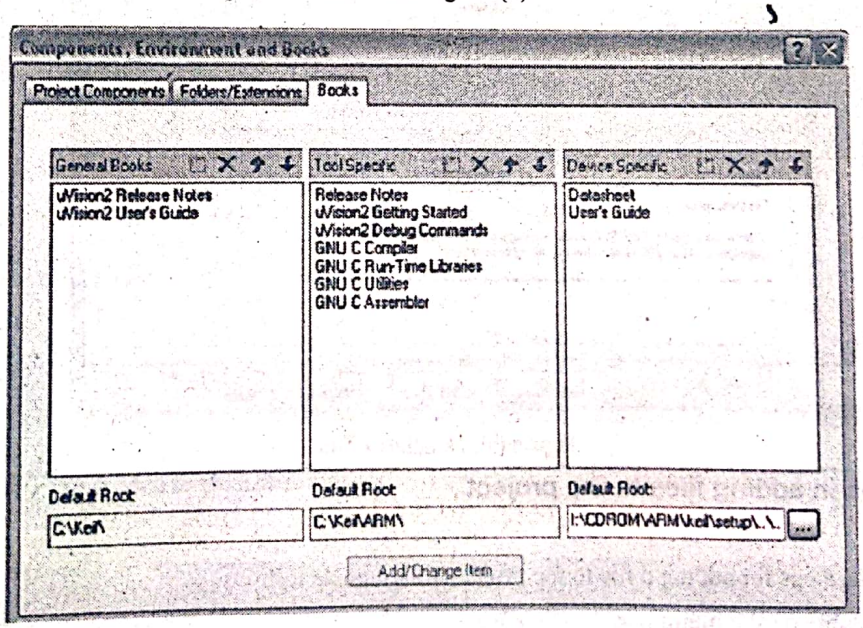


Figure (3): Books Tab

4. Functions Tab

The function tab displays all the functions available in 'C' source file in the project or in files only when they are open. The code of the specific function can be viewed by clicking on it as shown in figure (4).

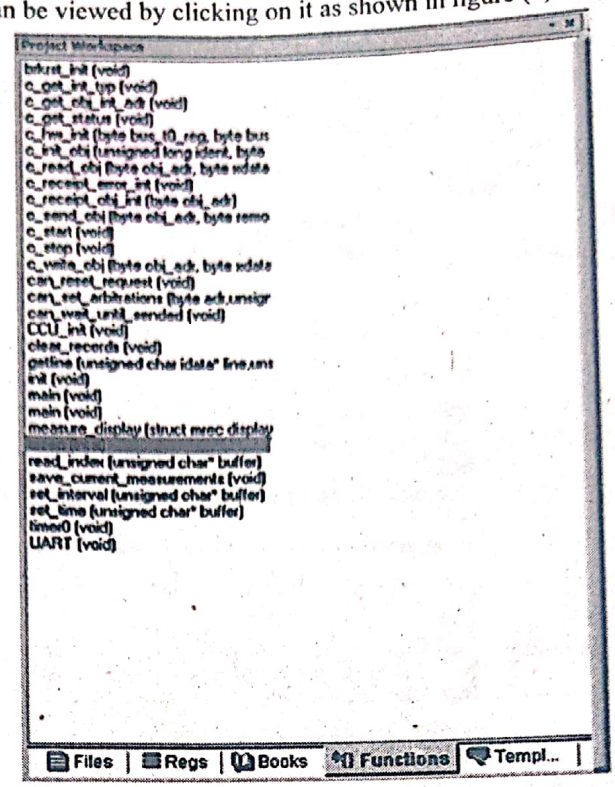


Figure (4): Functions Tab

5. Templates Tab

The templates tab will automatically produce the code framework for if, if else, switch case and code documentation template as shown in figure (5)

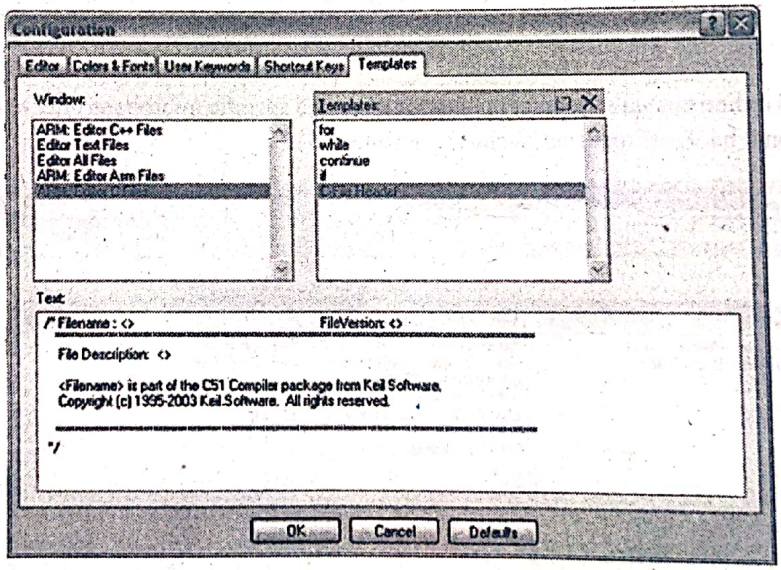


Figure (5): Templates Tab

Q14. Explain the steps in adding files to the project.

Ans:

The following are the steps for adding a file to the project are given as follows,

1. Go to 'file' tab available on the menu bar.
2. Select 'new' option from the drop down list. This displays a blank text editor at the right side of the window.

3. Write a program in ANSI C and 8051 specific codes in the text editor by using Keil specific embedded C codes.
4. Add all the required header files to the text editor by making use of # include compiler directive.
5. Make use of the standard template files which are available under 'templates' tab in project workspace, to add functions, loops, conditional instructions etc., in the code as shown in figure (1).

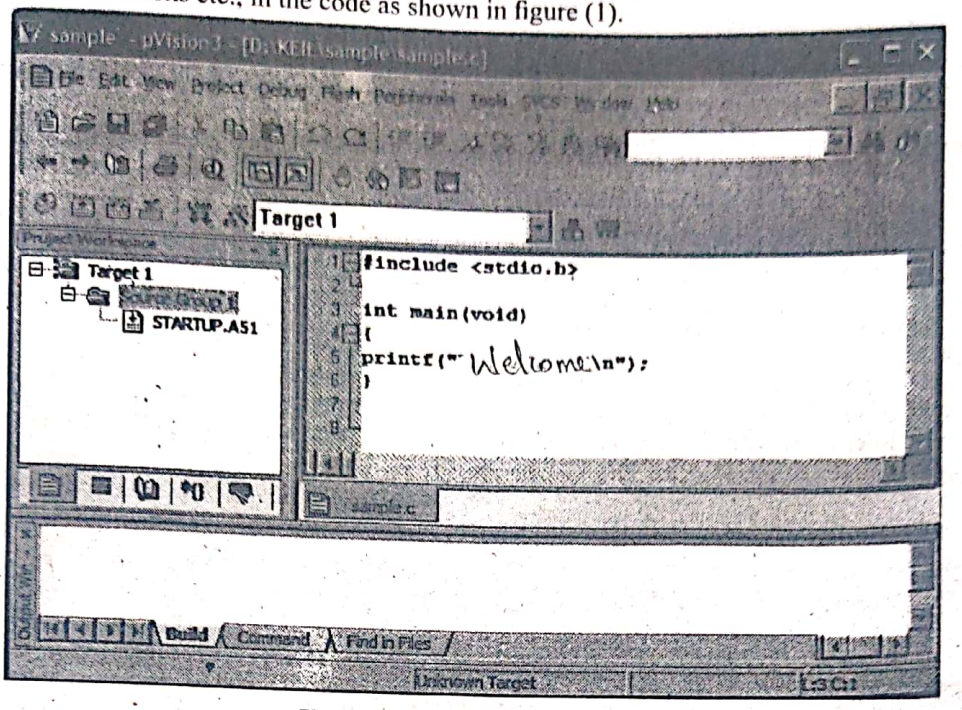


Figure (1): C Program in the Editor

6. Provide title to the file and save it with .C extension in the preferred folder.
7. Now the next step is to add the generated file to the project, which can be done by right clicking on the 'source group' in the project workspace.
8. Select 'add files' option in order to add a group with title 'source group'.
9. Select the file name to be added to the project from the file selection dialog box.
10. Click on 'add' button.
11. Click on 'close' button to close the selection dialog box. The selected file will be added to the target project as shown in figure (2).

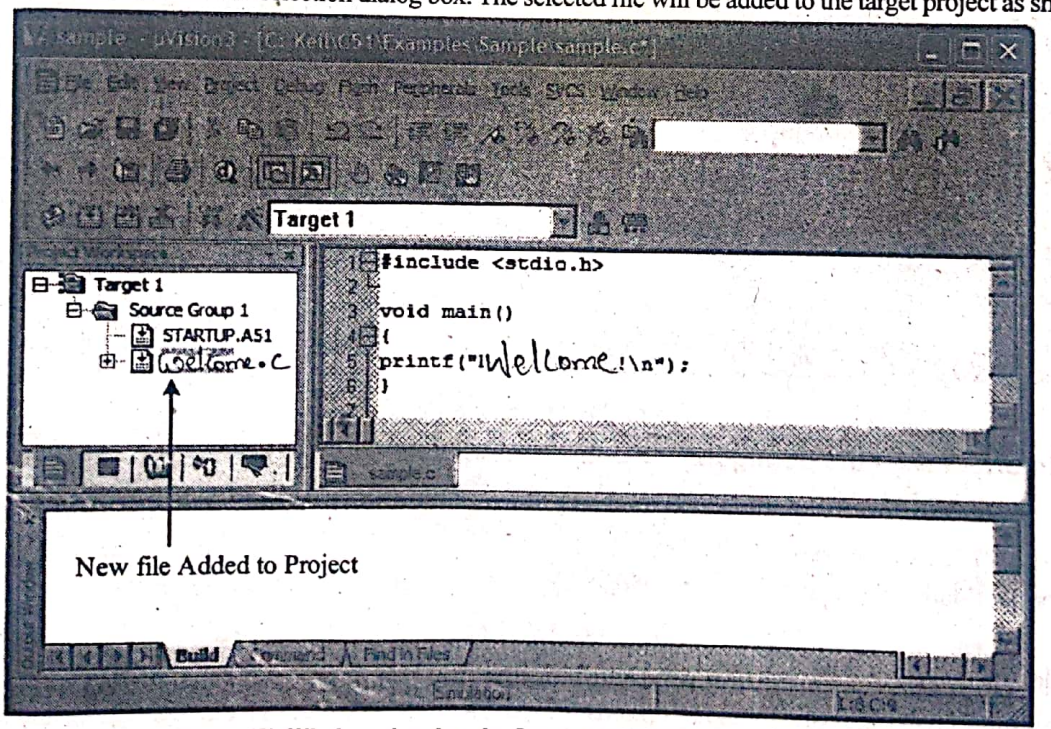


Figure (2): Window showing the Created File Added to the Project

Q15. Discuss the steps in performing target configuration.

Ans:

The steps for configuring the target are,

1. Click on 'project tab' and select 'options for target' from the drop down list.
2. Select 'target' tab and enter the clock frequency of the system in Xtal (MHz) text box.
3. Select the checkbox 'use on-chip ROM' if the system uses the code memory of the processor.
4. Select the memory model from the drop down list for the target.
5. Select the size of the code memory from the drop down list.
6. Select the type of OS based on the target application and hardware design.
7. Enter the starting address and size of the code memory under 'off-chip code memory' section, if external code memory is used.
8. Enter the starting address and size of the external memory under 'off-chip X data memory' section, if external working memory is used.
9. Click on 'OK' button as shown in figure.

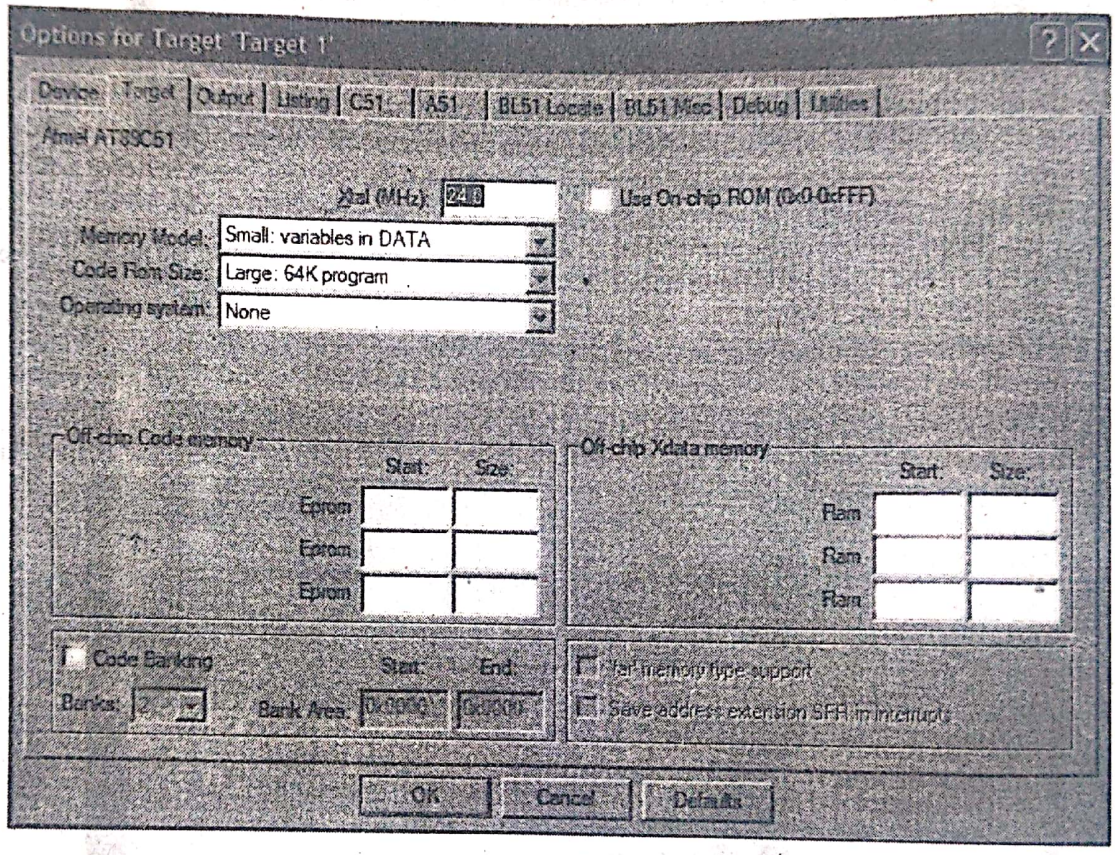


Figure: Configuring the Target

Q16. Discuss in brief about the configuring the firmware debugging.

Ans:

The firmware debugging can be configured as,

1. Click on 'project' tab and select 'options for target' from the drop down list. A configuration window is shown on the screen.
2. Click on 'debug' tab.
3. The window shows two types of debugging options they are, simulator based firmware debugging and a target firmware level debugging under separate sections as shown in figure (1).

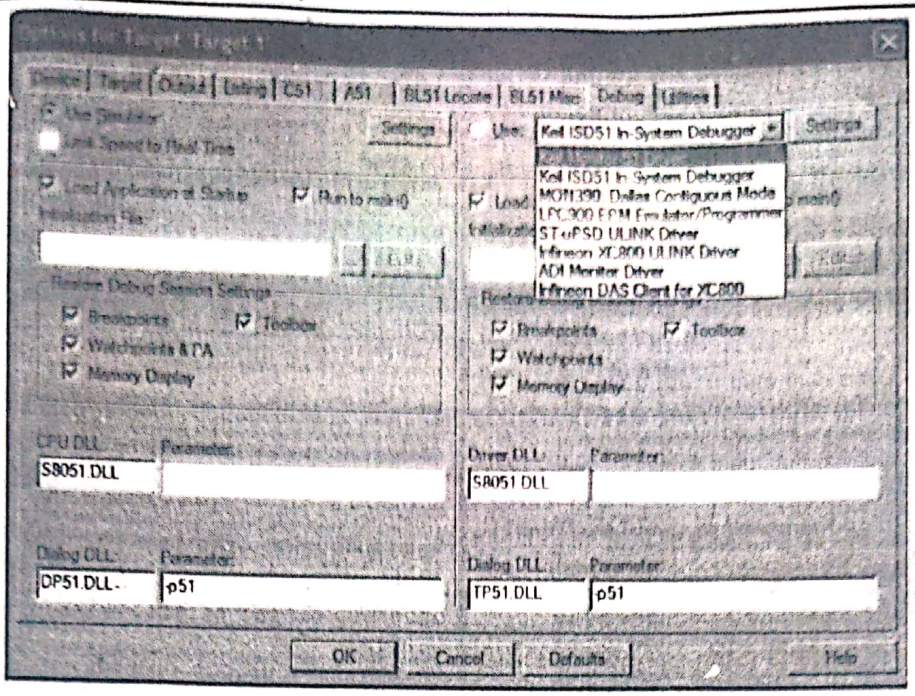


Figure (1): Options for Firmware Debugging

4. If simulator based debugging is chosen then it does not require downloading to select simulator firmware on the target machine. On the other hand if debugging the application is chosen then it requires downloading the firmware of debugging.
5. The target level debugging can be selected from the drop down list provided in the right most section (For example, Keil supported monitored programs or emulator interface).
6. Click on settings button to configure the serial interface to establish a link between the target hardware and IDE. A dialog appears with name target setup.
7. Select 'comm port' and 'baud rate' from the drop down lists provided under 'comm port settings' section.

This enable the binary file created to be downloaded on to the target board through serial connection which is configured as shown in figure (2).

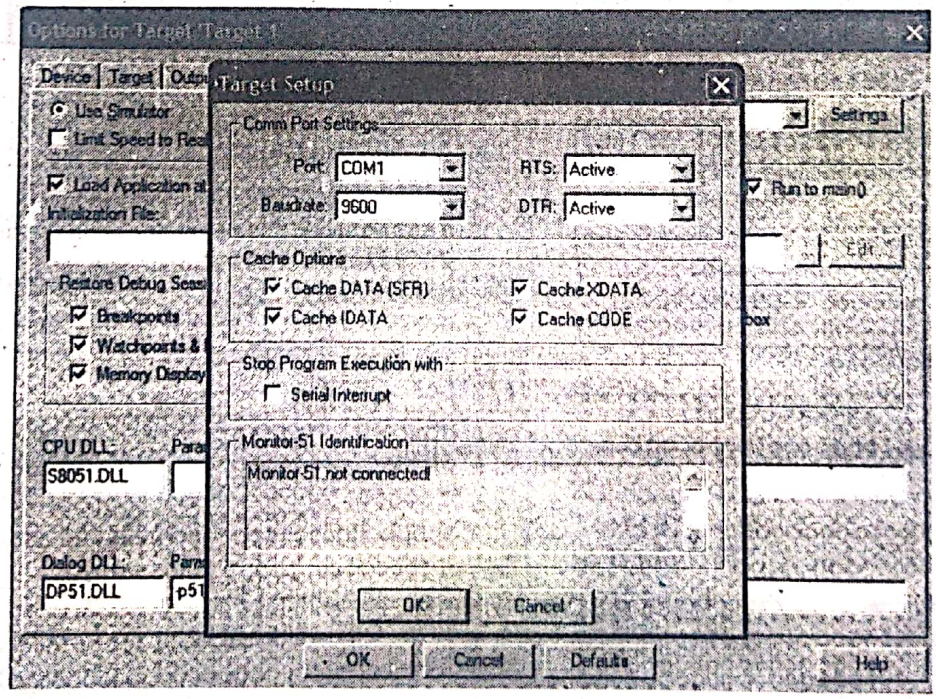


Figure (2): Configuring the Serial Interface

Q17. Discuss in detail the steps involved in performing cross compilation.

Ans:

Cross-compilation is a process of converting the firmware which is written in embedded C into machine language code. There are three different methods available to recompile a file. They are as follows,

Method 1

1. Click on 'project' tab in the menu bar.
2. Select 'rebuild all target files' from the drop down list or select 'rebuild all target files' icon on the toolbar. All the files which are present in the target group are cross compiled and a link is established between the codes of the file to create binary file as shown in figure (1).

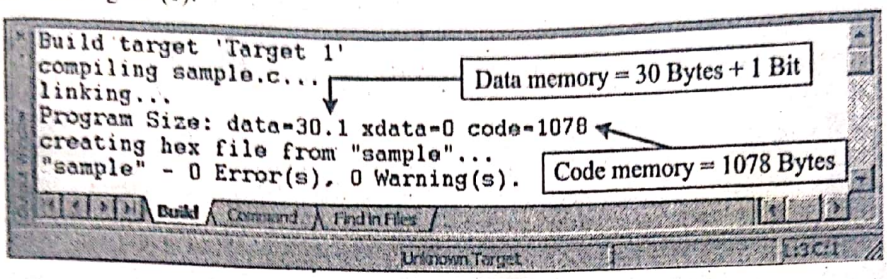


Figure (1): Output of "Rebuild all Target Files" Operation

Method 2

1. Click of 'project' tab in the menu bar.
2. Select 'build target' from the drop down list or select 'build target' icon on the tool bar.

This method links all the relocatable object files (i.e., the files which are cross compiled individually) in order to produce absolute object file and finally generates a binary file as shown in figure (2).

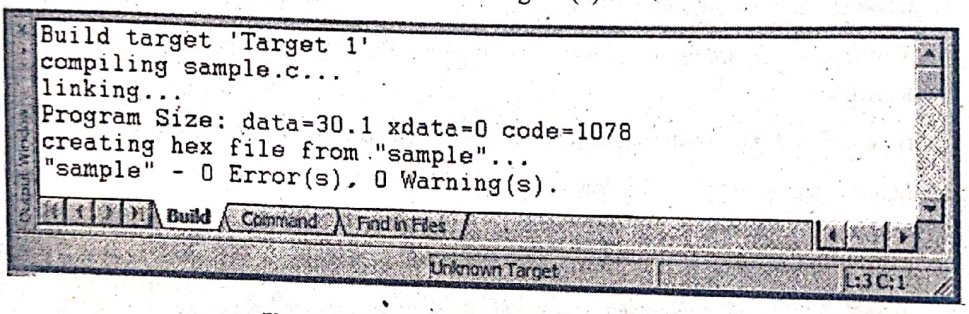


Figure (2): Output of "Build Target" Operation

Meth od 3

1. Click on 'project' tab on the menu bar.
2. Select 'translate current file' option from the drop down list or select 'translate current file' icon on the tool bar.

This method is called as selective compilation wherein each file is cross-compiled individually which creates corresponding re-locatable object files for the current file. This generated file is combined using "build target" with the other files in order to generate final executable file as shown in figure (3).

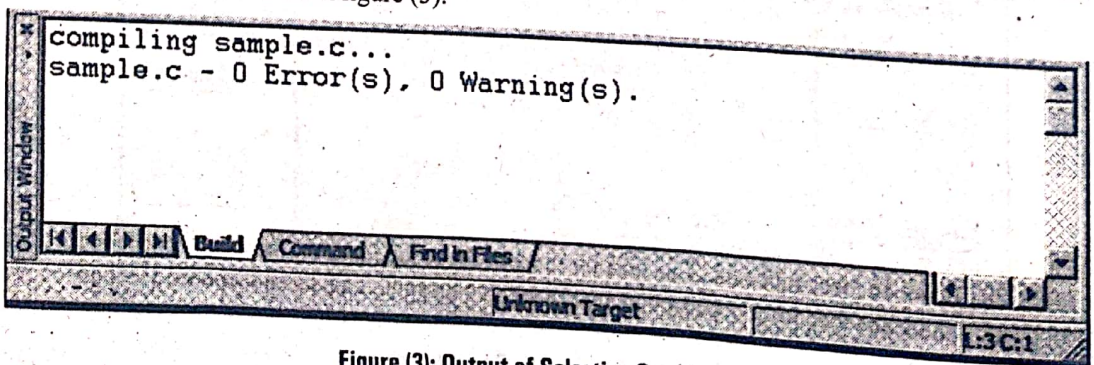


Figure (3): Output of Selective Compilation

Q18. List the advantages of hardware and software implementation.

Ans:

Advantages of Hardware Implementation

1. The hardware implementation enables the program to consume less amount of memory.
2. It facilitates processing speed.
3. It offers less number of chips, but this increases the cost.
4. It restricts the device driver to be programmed with easy code.
5. It provides greater security by allowing us to embed the code internally instead of placing it at external ROM.

Advantages of Software Implementation

1. A software is flexible to be implemented among different versions of hardware.
2. It reduces the time required for code development.
3. It facilitates the simple systems at minimum cost.
4. It is capable of performing most complex operation.
5. It can be machine independent (i.e., portability).
6. It allows the code to be divided into modules (i.e., modularity).
7. It facilitates even the complex function to be performed quickly through high speed microprocessor.
8. It allows the usage of standard tools such as modelling, RTOS and software engineering tools.

5.2 TYPES OF FILES GENERATED ON CROSS-COMPILATION

Q19. What is meant by cross-compilation? Also, list the various files generated during the cross-compilation process.

Ans: Nov./Dec.-15, Set-2, Q7(a)

Cross Compilation

The process of converting the source code into target machine code using an intermediate software (called cross-compiler) is known as cross-compilation. The cross compiler runs on the software platform whose configuration is different from that of the configuration of target machine. For example, the source code written in Embedded C (a high level language) is converted to target code written in ARM processor specific code by using the cross complier running on X86 machine.

During the process of cross-compilation several different intermediate files are generated in between the source file to the target machine code file. They are,

1. List File (.lst)
2. Hex File(.hex)
3. Pre-processor Output File
4. Map File
5. Object File(.obj)

Q20. Discuss in brief about list files.

Ans: Model Paper-I, Q6(a)

The list file is produced during the process of cross compilation. It has the extension (.lst). It contains the information related to cross-compiler such as,

1. **Page Header**

   The page header of it includes the information such as version number of the compiler, name of the source file, date and time of the file created and the page number.

2. **Command Line**

   The command line is present just below the page header and includes the information that is needed to invoke the compiler.

3. **Source Code**

   The third part is the source code in the list file. It includes the line numbers and the corresponding statements, cross compiler directives (describe all the content of include files), #if block, comments include files.

4. **Assembly Listing**

   This section includes the assembly code of the corresponding source C code which is generated by the cross compiler.

5. **Symbol Listing**

   This part includes the information about the various symbols available within the cross compiled source file such as,

   (i) The NAME (name)

   (ii) The CLASS symbol indicates to which it belongs such as type def, static, public, auto, extern etc.

   (iii) The MSPACE (memory space) i.e., whether it belongs to code memory or data memory.

   (iv) The TYPE (data type) such as int, char etc.

   (v) The starting address of the code memory i.e., OFFSET.

   (vi) The SIZE, (size) that is represented in bytes.

6. **Module Information**

   This part includes the information about the sizes of all the allocated and deallocated memory areas.

7. **Warnings and Errors**

   This part specifies the warnings and errors occurred during the process of cross-compilation. Warnings may be ignored but, the results obtained might be incorrect. The errors can never be ignored.

Q21. Explain in detail the hex file.

Ans:

The hex file can be described as an executable binary file generated from the source code. The source code comprises of an object file which is generated by linker/locator this file is however converted into a binary code.

For this purpose, a utility program called object to the hex file converter is employed that converts the object into a hex file. Furthermore, hex file follow various formats for variety of processors and controllers and it stores the machine code in a specific format. The most preferred hex file formats in embedded applications is,

1. Intel Hex
2. Motorola Hex.

1. Intel Hex

The intel hex comprises an ASCII test file that represents the hex data is ASCII format in the form of lines. These lines are in accordance to the hex record. These records are usually composed of hexadecimal numbers representing machine languages codes or constant data and the termination of each record is performed with a carriage return and a line feed. The utilization of these intel hex file is the transfer of data and program into ROM or EPROM.

Intel Hex File Format

The file consist of numerous hex records and each individual record has six fields arranged in the form of

: llaaattdd.....cc.

Here each group of letter signifies different fields and each individual letter denotes single hexadecimal digit. And at least two hexadecimal digits are reserved for each field.

The description of the format is as follows,

(i) **:** – It is a colon and represents the initialization of each intel hex record.

(ii) **ll** – It is a record length field that signifies number of records present in the record.

(iii) **aaaa** – It is a address field that denotes the initial address for the data present in the record.

(iv) **tt** – It is a field that represents the hex record type. Depending upon the value, it can be of following types.

00 : It signifies data record.

01 : It signifies end of file record.

02 : It signifies 8086 segment address record.

04 : It signifies extended linear address record.

(v) **dd** – It is a data field that signifies one byte of data. Each record can possess one or many data bytes and these data bytes must be equal to number specified by 'll' field.

(vi) **cc** – It is a checksum field that denotes the checksum of the record. The computation of checksum is performed by including the values of all hexadecimal digit pairs within the record and then calculating the modulo 256. Therefore, the generated result is 2's complemented to obtain checksum.

2. Motorola Hex File

In a way similar to Intel hex, motorola hex file comprises an ASCII text file that represents the hex data in ASCII format in lines. These lines are in accordance to the hex record. These records are usually composed of hexadecimal numbers representing machine language code or constant data.

The general format of motorola hex record is,

SOR RT length startaddress data/code checksum.

Also, it is represented as stllaaaaddddd.....cc.

(i) **SOR**

This field stands for start of record 's' denotes start of.

(ii) **RT**

This field stands for record type and 't' denotes the type of record specified in general format. According, to the value of 't' the different meanings for the record are,

0 : It is a header and indicates the beginning of the record.

1 : It indicates data record with 16 bit start address.

2 : It indicates data record with 24 bit start address.

9 : It indicates end of the file record.

(iii) **Length(ll)**

This field represents the total number of character pairs present within the record. However, it does include the type and record length during the representation. The '*l*' signifies values from 0 to 9 and *A* to *F* where as '*ll*' signifies length field.

(iv) **Start Address(aaaa)**

It is an address field that denotes the initial address for the data in the record.

(v) **Code/Data (dd)**

It is a data field that signifies one byte of data. Each record can possess one or many data bytes and these data bytes must be equal to the number specified by '*ll*' field.

(vi) **Checksum (cc)**

It is a checksum field that denotes the checksum of the record. The computation of checksum is performed by including the values of all hexa decimal digit pairs within the record and then calculating the modulo 256. Therefore, the generated result is 1's complemented to obtain checksum.

Q22. Explain the various details held by map file generated during the cross compilation.

April-18, Set-3, Q1(e)

(or)

Explain in detail about map file.

Ans:

Map files comprises different number of sections which are based on cross-compiler. It generally includes information about the link/locate process. It is not a compulsion that all the map files created by linker/locator consist of same information. It may differ depending upon the type of linker/locator used.

The information that is included in a map file is given below:

1. **Page Header**

   It specifies version number, date, time and page number on each page of the map file.

2. **Command Line**

   It indicates the entire command line used for initializing the linker.

3. **CPU Details**

   It includes the details regarding the target CPU and memory model such as internal data memory, external data memory, paged data memory and other.

4. **Input Modules**

   Input modules comprises of library files, name of the object modules and those modules that are involved in the process of linking. However, it makes sure that all the necessary modules in the linking process are lined.

5. **Memory Map**

   It classifies the different segments of a program such as starting address, length relocation type and segment name.

6. **Symbol Table**

   It includes symbol name, symbol type and symbol values of various input modules.

7. **Inter Module Cross Reference**

   It involves section name, memory type and module name in which it is specified and accessed.

8. **Program Size**

   It contains the information regarding the different memory sizes, space occupied by constants and code for complete application.

9. **Warning and Errors**

   Warnings and errors which are encountered during the process of linking a program are displayed in this section. It helps in debugging link errors.

Q23. Explain in detail about different files generated during the cross compilation of an embedded C file.

Ans: April/May-17, Set-2, Q6(a)

The different files generated during the cross compilation of embedded *C* file are as follows,

1. **List File**

   For answer refer Unit-V, Q20.

2. **Hexfile**

   For answer refer Unit-V, Q21.

3. **Preprocessor Output File**

   The preprocessor output file is produced during the process of cross-compilation. It includes the output of the preprocessor instructions used in the source file. Hence it is referred as a preprocessor output file. It is saved with the extension decided by the cross compiler. It is used during the checkings of conditional processor instructions and the processing of MACROS.

4. **Map File**

   For answer refer Unit-V, Q22.

5. **Object File**

   The Object File is produced during the cross compilation process with the extension "obj". Cross-compiling/ cross-assembling of the source code is written in assembly. It is saved with the extension .OBJ. It includes the conversions of instructions written in C source file or directives written in assembly language source file.

   .OBJ file format is based on the cross compiler and it defines two object file formats such as OMF51 or OMF2. It includes the information such as, library references, external symbol references, debugging detail, object code, names of variables and functions, library files and memory allocated for global variables.

   The absolute memory location for the .obj file is created by the linker/locator. The linker also links the external references that is a very important task in the execution of code produced by the cross-compiler.

5.3 DEASSEMBLER/DECOMPILER

Q24. Write a short notes on disassembler/decompiler.

Ans: (Nov./Dec.-14, Set-1, Q7(a) | Nov./Dec.-14, Set-2, Q7(a))

Disassembler

Disassembler is a program designed for converting machine codes into target processor which is nothing but assembly instructions and codes. This implies that machine language codes are converted into assembly language code. Also the functionality of the disassembler is reciprocal to the functionality of assembling/cross assembling.

Decompiler

Decompiler is a program designed for converting machine codes into high level language instructions. The functions performed by it is reciprocal to that of compiler/cross compiler.

Each family of processor/controller comprises different disassemblers and decompilers which are, generally used in reverse engineering process that deals with displaying of the technology used in developing the product and also determine the secrets behind the development of the product.

There are many disassemblers and decompiler tools freely available on internet for determining and examining the existence of viruses in the executable image files. Although it is considered to be a powerful tool but it is quite difficult for a disassembler to develop exactly same copy of the code similar to that of the original assembly code. Therefore only code which is to some extent a counterpart of the original assembly code is created.

Q25. State the uses of assembler and deassembler in embedded application development.

Ans: (Model Paper-I, Q6(b) | April-18, Set-1, Q6(a))

Assembler

The uses of an assembler in embedded application development are as follows,

1. It converts assembly language mnemonics into their equivalent binary code referred as executable file or object file.
2. It generates a list file which consists of information such as address, assembly language mnemonics and object codes (in hexadecimal format).

Deassembler

The uses of deassembler/disassembler in embedded application development are as follows,

1. It converts binary codes or machine codes into assembly language mnemonics.
2. This software is mostly used in reverse engineering technique to detect the proprietary products functioning.
3. It aids the reverse engineering technique by converting embedded software to assembly/high-level language.
4. This software helps in detecting the existence of virus in the program/image that has to be executed.
5. It creates a source code that is approximately similar to the actual source code.

5.4 SIMULATORS, EMULATORS AND DEBUGGING

Q26. Explain the advantages and limitations of simulator based debugging.

(Model Paper-II, Q6(a) | April-18, Set-1, Q6(b))

(or)

What are the limitations of simulator base debugging?

(Refer Only Limitations)

April/May-17, Set-4, Q1(e)

(or)

What are the advantages of simulator base debugging?

(Refer Only Advantages)

April/May-17, Set-2, Q1(e)

(or)

Mention the advantages and limitations of simulator based debugging.

(Nov./Dec.-14, Set-1, Q7(a) | Nov./Dec.-14, Set-2, Q7(a))

Ans:

Advantages

1. **Simulating Abnormal Conditions**

The developer can experience the change of behaviour of the firmware by providing any input values for any parameter when performing debugging. This will be useful for the developer in gaining experience of the abnormal conditions appeared and can help in studying the behaviour to overcome such affects.

2. **Donot Require Original Target Board**

Simulator based debugging donot require original target board as it is purely software based.

IDE's software creates an imitation of the target board by which users can do memory mapping as well as interface devices and write firmware depending on the target board which saves lot of development time.

3. **Simulate I/O Peripherals**

Simulator facilitates the advantage to simulate I/O peripherals without using connecting I/O devices instead it makes use of I/O support to edit and register input output values at the time of implementing firmware.

Limitations of Simulator Based Debugging

1. **Donot Possess Real Time Lines**

It is the major drawback that the simulator do not possess a real time behaviour as the simulated based debugging is developer driven and cannot generate a real time behaviour. On the other hand in case of real time applications I/O conditions may be different and unpredictable. Whereas performing simulation for known conditional values is different.

2. **Difference in the Output at Development and Real Time Environment**

A different output is achieved when debugging is performed by the developer in development environment when compared to the debugging performed in real time environment.

Q27. What is debugging and why debugging is required?

Ans:

Debugging is the process of examining firmware implementation and supervising the target processor's registers and memory, when the firmware is executing and verifying the signals from different buses present in embedded applications.

Debugging is mainly divided into two types,

1. Hardware debugging
2. Firmware debugging.

1. **Hardware Debugging**

It holds the responsibility of monitoring different bus signals as well as to check the status lines of the target hardware.

2. **Firmware Debugging**

It holds the responsibility of diagnosing the firmware execution, alterations done to different CPU registers, execution flow and status registers which provides the status on execution of firmware in order to make sure that the design is implemented according to the design plan.

Need of Debugging

Debugging is needed in order to determine the errors that occurred while executing firmware which can show unusual behaviour. It is similar to human body which shows unusual behaviour when not well or sick. At present many debugging tools have been evolved which can be used for diagnosing the problem which occurs in the firmware execution but traditionally it was a different scenario there was no proper tool available for debugging therefore burning process was used in order to check the product. While burning the code if the firmware crashes then the developer has to rework on the firmware and if it doesn't crash then it is understood that it is working properly.

Q28. What is a monitor program? Explain its role in embedded firmware debugging.

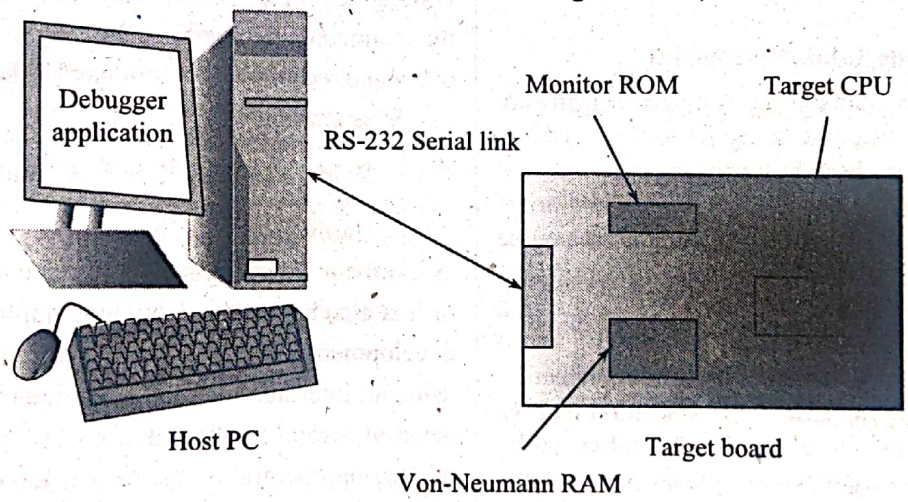
Ans:

(April/May-17, Set-2, Q6(b) | Nov./Dec.-15, Set-2, Q7(b) | Nov./Dec.-14, Set-3, Q7(a))

Monitor Program

This technique develops a monitor program capable of handling the entire source code, its commands and responses to it. The monitor program controls a set of commands that downloads firmware, inspects or modifies the memory or register locations performs single stepping etc. After receiving the commands for their corresponding operation, develop the code that implements them. The commands are received by the external interface of the target processor like RS-232C, USB, serial or parallel interface. The monitor program receives the commands by querying the interface or by controlling the command reception. The command is then examined to perform its corresponding task.

Monitor program based target firmware debug setup is shown in figure below,



Figure

The features exhibited by monitor program are,

1. Command set interface to communicate with the debugging application.
2. Firmware download option to code memory.
3. Processor registers and RAM (working memory) is examined and modified.
4. One step program execution.
5. In firmware execution, break N/S points is specified.
6. Debug information is sent to the debug application running on host machine.

Q29. Describe in detail the improvements over firmware debugging starting from the most primitive type of debugging to the most sophisticated on chip debugging.

Ans:

April-18, Set-3, Q6(a)

The improvements over firmware debugging starting from the most primitive type of debugging to the most sophisticated on chip debugging are listed below,

1. Incremental EEPROM Burning Strategy

This technique involves division of code into several functional code units. They are then burned in an incremental order such that the codes for their respective functionalities are coded separately, cross-compiled and burned into the chip one after another. There must be some indication support in the code like LED or BUZZER to indicate whether the code is working according to the expectations or not. The arrangement for indication support is done by fixing a LED or a BUZZER into the target board during the hardware designing phase. The functionality of the code is examined in a step wise manner. In the first step it is checked whether the code is functioning properly on the target board with the code burned into the EEPROM. In the next step, burning process is performed on the code and checked if the code is functioning perfectly. The above steps must be repeated until all the functional code units are covered. However, developer should make sure that it obtains the correct output before moving from one step to another step. If expected outputs are not obtained then modify the code before moving to the next step once the developer obtains the desired output from all the functional code units then it should integrate and recompile the code to check the functioning of the entire code.

2. Debugging Based on Inline Breakpoint

This technique shows that the execution of the firmware has reached upto some specific points using debug codes. Debug codes are inserted at each such point whenever the execution of the firmware reaches any point. This is notified by a printf function that prints the string given by the firmware. The source code is cross-compiled with debug codes embedded within it on burning the hex file into the EEPROM, the data that is generated by printf( ) can be seen on the hyperterminal. The serial communication settings of the "HyperTerminal" connection must be configured similar to the serial communication settings configured in the firmware. Using an RS232 cable, make a connection between the target board's serial port (COM) and PC's COM port supply power to the target board. This arrangement displays the debug information on the hyper terminal. Following is the example of inline debug codes and the extraction of debug data.

```
Printf ("Configuration has been started\n");
Configurations
----------
----------
Printf ("Configuration has been ended\n");
Printf ("Starting of firmware execution\n");
Code Section
----------
----------
Printf ("Ending of code section\n");
```

The following hyper terminal screen is obtained if the firmware has no error and the execution goes well.

Target Debug-Hyper Terminal					
File	Edit	View	Call	Transfer	Help
Configuration has been started					
Configuration has been ended					
Starting of firmware execution					
Ending of code section					

Figure: Debug Messages on the Hyper Terminal

3. Debugging Based on Monitor Program

This technique develops a monitor program capable of handling the entire source code, its commands and responses to it. The monitor program controls a set of commands that downloads firmware, inspects or modifies the memory or register locations performs single stepping etc. After receiving the commands for their corresponding operation, develop the code that implements them. The commands are received by the external interface of the target processor like RS-232C, USB, serial or parallel interface. The monitor program receives the commands by querying the interface or by controlling the command reception. The command is then examined to perform its corresponding task.

4. Debugging Based on In Circuit Emulator (ICE)

Hardware device that emulates the target CPU is known as emulator. It comprises of an emulation logic and one end of it is attached to the debugging application that runs on the development PC. The other end is linked to the target surface using an interface. Emulators are the hardware devices which generally emulates the functions of a processor and performs debugging operations like suspension of firmware execution, setting of breakpoints, getting or setting internal RAM /CPU register etc. There are some software applications which works as a hardware emulator. Such applications are known as "Emulators". The emulator application that emulates the functioning of a PDA phone for developing an application is considered as a "software emulator". The debugger application might be a component of the Integrated Development Environment (IDE) or some other supplied tool.

Emulator POD is the basic element of an emulator system. It has the following functional units.

(i) Emulation Device

It is a mimic of the target CPU. The target CPU receives signals from the target surface via device adaptor linked to the target surface. It executes the firmware using debug commands of the debug application types,

(a) Standard chip same as target processor.

(b) Programmable Logic Device (PLD) which functions same as that of target CPU.

The standard chip, if used as an emulating device facilitates real time execution but they cannot be reused for other target processors. In contrast, PLD emulators can be reconfigured and reused for different derivatives of the target CPU but it has the disadvantage of accuracy and logic implementation.

(ii) Emulation Memory

A memory (in the form of Random Access Memory) is provided inside the emulation device which is used for the replacement of target board's EEPROM. This memory is referred as emulation memory. The original memory stored in the EEPROM is therefore emulated by RAM of emulator device. This process is known as "ROM Emulation". The "ROM emulation" process prevents the chances of ROM burning and facilitates a number of reprogramming.

(iii) Emulators Control Logic

These are the logic circuits used for the following purposes,

(a) For implementation of different hardware breakpoints.

(b) For tracing buffer trigger detection.

(c) For tracing buffer control.

(d) For developing functions of logic analyser in an emulator device.

(iv) Device Adaptors

A pin to pin compatible socket that provides an interface between emulator POD and target CPU are known as device adaptors. These pin-to-pin compatible sockets are plugged to the target board. In order to route different signals that are received from pins allocated to the target processor. There are various types of device adaptors and their selection depends upon the package of target's processor chip.

2. On Chip Firmware Debugging (OCD)

On-chip debugging is a target firmware debugging method which comprises of built-in debug module known as on-chip debug support. The On Chip Debug (OCD) which provides fast and efficient debugging on the target firmware is used by almost all the processors even though it adds hardware complexity. This method owns technologies such as background debug mode, On CE etc., as it is chip vendor dependent. The debugging is controlled with the help of dedicated registers created by OCD.

The debugger uses registers for debugger control operations such as enabling or disabling of single stepping, freezing of execution etc. Two types of interfaces are used for communicating between the target CPU and the debug application (running on development PC).

(a) Background Debug Mode (BDM)

(b) Joint Test Action Group (JTAG).

The background debug mode was developed by Motorola to define how communication interface is established among chip resident debug core and host PC. It uses a 10 or 26 pin connectors for connecting with the target board. The debugger application running in the host PC should be BDM compatible. BDM use three signal lines such as DSI (Serial Data In), DSO (Serial Data Out) and DSCLK (Serial clock) for debugging. The DSI is used for transmitting signals from remote debugger application to target processor serially while DSO transmits the debug response to the host PC from the processor. The serial clock synchronises the serial transmission. In BDM, debug commands (17-bit wide) are used for controlling the debugging process.

JTAG interface consists of a built-in JTAG port which facilitates communication between the debugger application and target CPU. It is also a serial interface and uses TDI, TDU, TCK, TMS and TRST as signal lines. The data transfer rate depends upon the chip used in debugging.

5.5 TARGET HARDWARE DEBUGGING

Q30. Explain the different tools for hardware debugging.

(April-18, Set-3, Q6(b) | April/May-17, Set-3, Q6(b))

(or)

What are the different techniques available for embedded firmware debugging? Explain them in detail.

(April-18, Set-4, Q6(b) | Nov./Dec.-15, Set-1, Q7(b))

(or)

What do you mean by hardware debugging? Explain different tools used for hardware debugging.

(Nov./Dec.-14, Set-1, Q7(b) | Nov./Dec.-14, Set-4, Q7(b))

(or)

How the target hardware debugging done in design of embedded system design?

Dec.-13, Set-3, Q7(b)

Ans:

Hardware debugging holds the responsibility of monitoring different bus signals as well as to check the status lines of the target hardware.The different types of hardware debugging tools used in embedded product development are discussed as follows,

1. Magnifying Glass (Lens)

A magnifying glass or lens is the primary hardware debugging and powerful tool used for visual inspection in embedded product development. Magnifying glass can be used for examining target board's surface operations such as dry soldering of components, missing components, improper placement of components, improper soldering track (PCB connection) damage, short of tracks etc. At present many high quality magnifying stations are being used for visual inspection. It comprises of different magnifying lenses fixed to a stand in order to provide better illumination for proper inspection. The main lens is the major tool used for visual inspection of the entire hardware board while the other lenses are used for magnification of comparatively smaller areas of the board that needs a proper inspection.

2. Digital CRO

Cathode Ray Oscilloscope (CRO) is a tool used for capturing and analysing waveforms. It is also used for measuring the signal strength. The behaviour of the waveforms can be analysed and captured by observing the point on the target board associated to oscilloscope channels. The crystal oscillators signal monitored from the target board is one of the examples of waveform capturing and analysing CROs may be analog or digital. Digital CROs are expensive and work well for target board applications. They are used in the measurements of phase, amplitude etc. Some of the modern techniques adopted by digital CROs are as follows,

(i) It is used for recording waveforms for a specific period of time.

(ii) It is used for capturing waves based on the specific configuration from the target board.

(iii) It includes two or more channels so as to capture and analyse signals using different channels present on the target board.

3. Multimeter

Multimeter is an important tool for embedded hardware developer as it is used for measuring different electrical quantities like current, voltage, resistance, capacitance, checking continuity and transistor and identifying cathode and anode diodes etc. Hardware developers use the multimeter for debugging the hardware. It checks the continuity of the circuit between various points on the board, measures the voltage supply, checks the polarity, signal value etc. Although multimeter has analog and digital versions, but most of the developers use digital version compared to analog version because of its features such as readability and accuracy.

4. Logic Analyser

Logic analyser is especially used for capturing digital data with the use of some special connectors and clips fixed to the target board whereas CRO is used for capturing different waves forms. CRO has the drawback of capturing restricted number of logic signals or waveforms depending on the number of channels present. The debugging applications of the target board uses logic analyser for capturing the states of different port pins, address bus and data bus of the processor or controller of the target. It provides the exact description of the target board when a specific line of firmware is implemented which is generally obtained by capturing the data line logic and address line logic of target hardware.

5. Function Generator

Function generator is an input signal simulator tool used for generating different periodic waveforms such as sine wave, square wave, saw-tooth wave consisting of distinct amplitudes and frequencies. At some point of time, target board needs a specific form of frequency with a particular type of waveform to be supplied hence function generation serves this requirement.

5.6 BOUNDARY SCAN

Q31. Explain in detail about boundary scan.

April/May-17, Set-4, Q6(a)

(or)

Explain the boundary scan based hardware debugging in detail.

(April-18, Set-2, Q6(a) | Nov./Dec.-15, Set-4, Q7(b) | Nov./Dec.-14, Set-2, Q7(b) | Dec.-13, Set-2, Q7(b))

(or)

Write a short note on boundary scan.

Ans: Nov./Dec.-14, Set-2, Q8

Boundary scan is a method utilized for testing the interconnection between the number of chips present on target board. A JTAG interface is used in order to perform the test in an efficient manner. The JTAG interface basically consists of the following,

1. Various chips that contain boundary scan cell that correspond to its cell.
2. A TAP (Test Access Port) formed by the 5 signal lines (TD_1, TD_0, TCK, TRST and TMS) of JTAG port.
3. A PCB for TAP in order to connect JTAG signal line to enternal world.

The devices present in the board are interconnected with signal lines like TD_1, TD_0, TCK, TRST and TMS to form a boundary scan path.

This interconnection is performed in the following manner,

1. Initially connect the TD_1 pin of TAP present in PCB to the TD_1 pin of the first JTAG device.
2. Similarly, connect the TD_0 pin of the JTAG first device to the TD_1 pin of second device.
3. Interconnect all the other devices in the similar manner and simultaneously connect the TD_0 pin of last device to the TD_0 pin of TAP present in PCB.
4. Finally connect the TCK (Clock line) and the TMS (Test Mode Select) line to the clock and TMS line of TAP present in PCB.

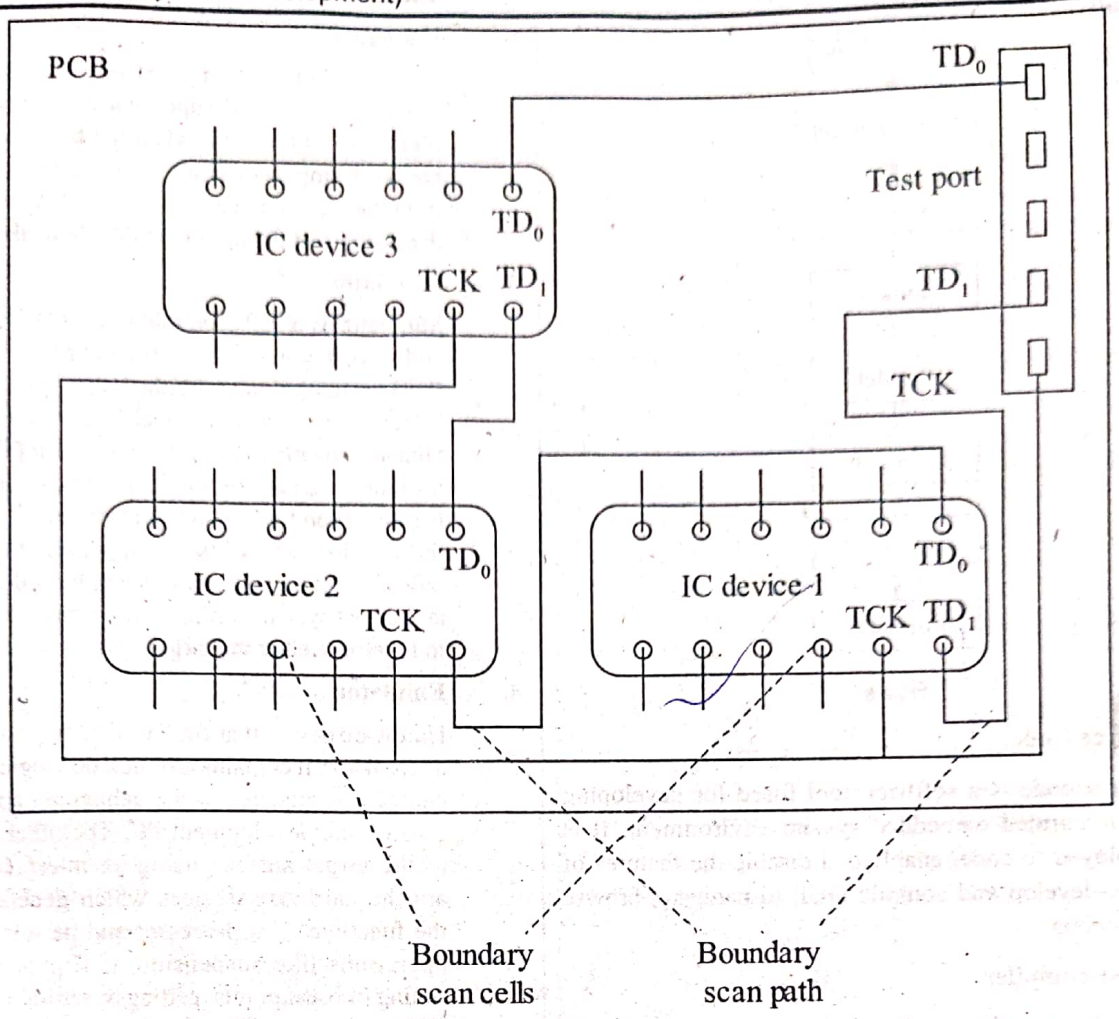


Figure: Boundary Scanning Based on JTAG

Boundary scan cell is a multipurpose memory cell that is associated with two pins namely I/P and O/P pins. These input and output pins of an IC are known as input cells and output cells respectively. The boundary scan cell performs the following functions.

1. Initially, it captures the signal state of the input pin and then forwards it to the internal circuitry.
2. It then captures the signals from IC and forwards them to the output pin.
3. Finally, it shifts the received information from TD_1 which is present in TAP.

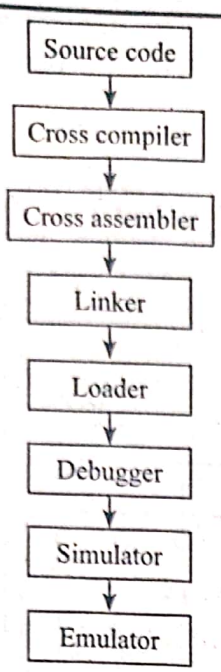
The cells corresponding to the pins are interlinked to form a chain from all along TD_1 pin to TD_0 pin. The boundary scan cells can be managed in four modes namely normal, capture, update and shift. In addition to the above, special registers are provided by the IC's that facilitate the scan operation. Some of those registers are instruction register, bypass register, identification register etc.

Q32. Draw and explain the integrated embedded system development environment.

Ans: April/May-17, Set-1, Q6(a)

Integrated Development Environment (IDE) is a software package consisting of text editor, cross-compiler, linker, loader simulator, emulator and a debugger. In embedded software development, it integrates environment to develop and debug the target processor specific embedded firmware.

The figure below illustrates the block schematic of IDE.



Figure

1. **Source Code**

Source code is a software tool fused for developing the integrated embedded system environment. It is employed to code, enable and disable the features of C++, develop and compile GUI, to nanigate, browse and debug.

2. **Cross-compiler**

The compiler that converts source code (written in C, C++ i.e., high level languages) to binary executable files, which are understandable by the target system processor is referred to as a cross-compiler.

3. **Cross-assembler**

It converts object codes of executable codes used by a processor into other codes that are to be used by another processor and vice versa. It first assembles the code of the target processor and then generates the object code for that target processor.

4. **Linker**

A linker links the required object and library code files. A linker permits programmer/user to create a program separately compiled or assembled files i.e., a linker constructs a single program by combining each machine instructions including standard library routines.

5. **Loader**

A loader reallocates the available address and places them into the memory at some particular physical address. Binary code must be loaded into different available addresses and then the program is allowed to run on the system. A loader is a part of the operating system and performs multiple operations on host machines.

6. **Debugger**

Debugging is the process of examining firmware implementation and supervising the target processors registers and memory, when the firmware is executing and verifying the signals from different buses present in embedded applications. The software tool used for the process of debugging is known as debugger.

7. **Simulator**

Simulator is a software that cross-compiles the source code (written in C/assembly) and places it at the RAM (host system) along with the behaviour of the registers (of the target system). It also has linker/locator that allocates the memory for the cross-compiled code, links it to the external references and allows it to run as if it is running on the actual target system. It is used during the development phase of application software. It is independent of the target system. It produces the results at the host system similar to the results that would have to be produced at the target system.

8. **Emulator**

Hardware device that emulates the target CPU is known as emulator. It contains an emulation logic, in which one end of it is attached to the debugging application that runs on the development PC. The other end is linked to the target surface using an interface. Emulators are the hardware devices which generally emulates the functions of a processor and performs debugging operations like suspension of firmware execution, setting of breakpoints, getting or setting internal RAM/ CPU register etc. There are some software applications which works as a hardware emulator such applications are known as "Emulators".

5.7 EMBEDDED SOFTWARE DEVELOPMENT PROCESS AND TOOLS

Q33. List the features of,

(i) Source code engineering tool

(ii) IDE.

Ans:

(i) Source Code Engineering Tool

This tool enables compiling, development and cross-compiling of the source code. This tool possess the following features.

1. It enables search and replacement in an automated and easier way.
2. It provides the list for the following,
   (a) Definitions, symbols, hierarchy of the class and class inheritance trees.
   (b) Dependencies of the symbols as well as defined symbols, variables, functions and other symbols.
   (c) Dependencies and hierarchy of the included header files.

3. It manages and monitors the implementation of virtual functions in order to utilize it for dynamic binding.
4. It automatically eliminates the operations (or tasks) that are not in use and Susceptable to errors.
5. It determines impact on the source code which is due to modifications in the code.
6. It moves forward and backward between the following,
   (a) Implementation and symbol declaration
   (b) Overridden and overriding methods.
7. It searches for the following information,
   (a) Instantiation of a class
   (b) Hiding variables among the members
   (c) Visibility of the members
   (d) Relationship between the component and the object.

(ii) **IDE**

IDE (Integrated Development Environment) is a software application comprises of compilers, emulators, logic analysers, EPROM/EEPROM assemblers and editors. It provides the following features,

1. It performs verification on the performance of a target system.
2. It performs debugging through single stepping and facilitate synchronization of internal peripherals.
3. It simulates the hardware unit available on the host system, this unit may comprise of emulator, peripherals and I/O devices.
4. It allows a user to define assembler depending on the version or type of processor.
5. It facilitates the multiuser environment.
6. It provides facilities for defining the version and the family of a processor.
7. It supports conditional and unconditional break points.
8. It facilitates with test vectors.
9. It provides the following details with the help of windows on the screen.
   (a) Source code part with labels
   (b) The register during execution
   (c) Symbolic arguments
   (d) Status of the peripheral devices, RAM and ports, stack as well as the program flow.

Q34. Discuss in detail about host systems.

Ans:

Host Systems

Host system is a system where the desired hardware and software is developed. It may be a PC, workstation or Laptop. It comprises of the following hardware and software tools,

Hardware Tools

1. Monitor
2. High-performance processor along with cache memories
3. High ram capacity memories
4. High-capacity memory on disk
5. Mice
6. Keyboard
7. Network connection
8. Read Only Memory Basic Input/Output System (ROM-BIOS).

Software Tools

1. **Cross-assembler**

   It converts object codes or executable codes used by a processor into other codes, to be used by another processor and vice versa. It first assembles the codes of the target processor and then generates the object code for the target processor.

2. **Cross-compiler**

   The compiler that converts source code (written in C, C++, i.e., high level languages) to binary executable file understandable by the target system processor is referred to as a cross-compiler.

3. **Program Development Tool Kit**

   This kit allows us to write the source code in any programming language such as, C, C++, Visual C++, Java code or assembly mnemonics. It allows us to perform the following functions on source code such as,
   (i) Insertion, deletion, addition, merging records files and appending lines or files, etc..
   (ii) Storing, creating and modifying the files.

Moreover this kit comprises of different tools such as assembler, compiler, loader and linker for generating the code.

Q35. Write short notes on target systems.

Ans:

Target system is a system where the required hardware and software is get tested, debugged and finalized during the development of embedded system. This system is given connection with a computer. Once the code is finalized, either the target system itself is considered to be as the final product (embedded system) or different copies of it are generated that acts as embedded system.

However, target system contains the following components.

1. Processor
2. ROM for processing the ROM image of the embedded software.
3. RAM is used for stack, temporary variables and memory buffers
4. Peripherals and interfaces.

In addition to the above components some target systems possess flash memory (8 or 16 MB), SDRAM (64 MB) and RS 232 and 10/100 base ethernet connectivity or a USB port.

The figure below depicts the diagrammatic representation of simple target system.

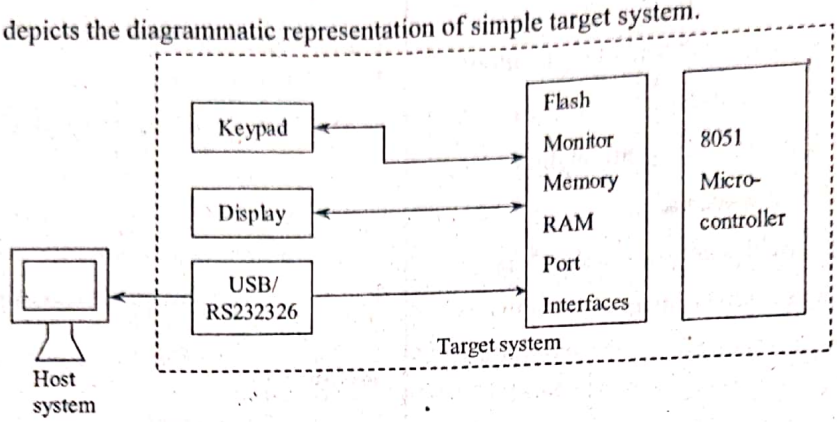


Figure: Target System

Q36. Define linker and loader. Also discuss the features of locator.

Ans:

Linker

A linker links the required object and library, code files. A linker permits a programmer/user to create a program in separately compiled or assembled files i.e., a linker constructs a single program by combining each machine instructions including standard library routines.

Advantages

A linker minimizes the size of the program by,

1. Eliminating binary code associated with functions and procedures that are not invoked.
2. Removing the space assigned to the variables that are declared but not used.

A locator for an embedded system is called locator or cross-linker. It locates codes of a hardware driver diver (such as ROM) and places it permanently at an address available in the ROM. In embedded systems, a locator acts as a program which keeps track of available memory and creates files for permanently locating the codes. A locator uses output of the cross-assembler and a memory allocation map for locating permanent locations of codes. A locator performs various functions on a target system.

Loader

A loader reallocates the available addresses, and places address into the memory at some particular physical address. Binary code must be loaded into different addresses available and then the program is allowed to run on the system. A loader is a part of the operating system and performs multiple operations on host machines.

Features of Locator

1. The locator is a program whose code is written by the programmer. It includes the definition of available addresses present in RAM and ROM. These defined addresses are used by the locator to create and load files for permanently positioning the codes by making uses of device programmer.
2. The locator makes use of the cross-assembler output (which is nothing but a memory allocation map) and generates locator program output file.

3. It makes use of cross-compiled code which comprises of different segments such as instruction, initialized values and addresses, constant strings and un-initialized data.
4. It is also responsible for searching out the I/O tasks and hardware device driver code at their respective addresses without any need of reallocation. Since, these addresses are already predetermined for any particular system.
5. It generates the locator program output which can either be in motorola S-record format or Intel hex file format. It also reallocates the linked file and generate a file for permanently locating the code in standard format.

Q37. Explain the various elements of an embedded system development environment.

Ans: (Model Paper-II, Q6(b) | April/May-17, Set-4, Q6(a))

The various elements of a embedded system development environment are,

Integrated Development Environment (IDE)

Integrated Development Environment (IDE) is a software package consisting of text editor, cross-compiler, linker, loader simulator, emulator and a debugger. In embedded software development, it integrates environment to develop and debug the target processor specific embedded firmware.

Disassembler

Disassembler is a program designed for converting machine codes into target processor which is nothing but assembly instructions and codes. This implies that machine language codes are converted into assembly language code. Also the functionality of the disassembler is reciprocal to the functionality of assembling/cross assembling.

Decompiler

Decompiler is a program designed for converting machine codes into high level language instructions. The functions performed by it is reciprocal to that of compiler/cross compiler.

Simulator

Simulator is a software that cross-compiles the source code (written in C/assembly) and places it at the RAM (host system) along with the behaviour of the registers (of the target system).

Emulator

Hardware device that emulates the target CPU is known as emulator. It contains an emulation logic, in which one end of it is attached to the debugging application that runs on the development PC. The other end is linked to the target surface using an interface. Emulators are the hardware devices which generally emulates the functions of a processor and performs debugging operations like suspension of firmware execution, setting of breakpoints, getting or setting internal RAM/CPU register etc. There are some software applications which works as a hardware emulator such applications are known as "Emulators".

Debugger

Debugging is the process of examining firmware implementation and supervising the target processors registers and memory, when the firmware is executing and verifying the signals from different buses present in embedded applications. The software tool used for the process of debugging is known as debugger.

Target Hardware Debugging

Hardware debugging holds the responsibility of monitoring different bus signals as well as to check the status lines of the target hardware..

Q38. Explain about the embedded software development process and tools. (Model Paper-III, Q6 | Dec.-13, Set-2, Q7(a))

(or)

Write notes on embedded software development-process. April/May-17, Set-1, Q6(b)

(Refer Only Embedded Software Development Process)

Ans:

Embedded Software Development Process

The process of building software for an embedded system is as shown in figure below,

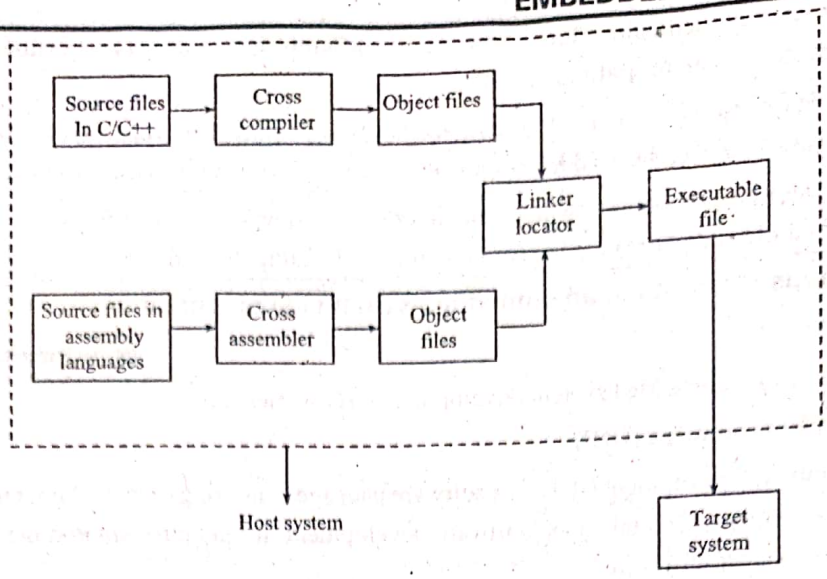


Figure: Embedded Software Building Process

The developmental process of embedded software requires two systems. Namely,

1. Host system
2. Target system.

1. Host System

Host system is basically a general-purpose computer, where the actual software is developed. A host system includes various software developmental tools such as compilers, assemblers, linkers and locators.

Procedure for Software Building within a Host System

(i) Initially, the software written in C/C++ languages is compiled into object code using a cross compiler. Similarly, the software written in assembly language is assembled into object code using cross assembler. Even though cross-assembler runs on the host system it produces object files that can suitably run on target system.

(ii) In the next step, the compiled and assembled object files are converted into a single binary executable file using linker/ locator

(iii) Finally, the executable file is loaded onto the memory of target system.

2. Target System

A target system is basically an embedded system which runs the software developed by the tools in the host system. The target system accepts the binary executable file and runs the file on its platform.

Embedded Software Tools

(i) Cross-Compiler

For answer refer Unit-V, Q34, Topic: Cross Compiler.

(ii) Cross-Assember

For answer refer Unit-V, Q34, Topic: Cross Assembler.

(iii) Program Development Tool Kit

For answer refer Unit-V, Q34, Topic: Program Development Tool Kit.

(iv) Linker

For answer refer Unit-V, Q36, Topic: Linker.

(v) Loader/Locator

For answer refer Unit-V, Q36, Topic:Loader.