

UNIT-4

1 Explain in detail about Super loop based approach for Embedded firmware design.

A:- THE SUPER LOOP BASED APPROACH

- * The Super loop based firmware development approach is suitable for applications that are not time critical and where the response time is not so important.
- * It is very similar to a conventional procedural programming where the code is executed task by task.
- * The tasks listed on top on the program code is executed first and the tasks just below the top are executed after completing the first task.
- * In a multiple task based system, each task is executed in Serial in this approach.

A typical Super loop implementation will look like:

- * Configure the common parameters and perform initialization for various hardware components memory, register etc
- * Start the first task and execute it
- * Execute the next task
- * 5.:
- * 6.:
- * Execute the last defined task
- * Jump back to the first task and follow the same flow.
- * The order in which the tasks to be executed are fixed and they are hard coded in the code itself.
- * There

- * Since the tasks are running inside an infinite loop, the only way to come out is either a hardware reset or an interrupt assertion.
- * A hardware reset brings the program execution back to the main loop.
- * An interrupt request suspends execution temporarily and completes the corresponding interrupt routine and then restarts task execution from the point where it got interrupted.
- * In this approach the priorities and task are fixed hence there is no need of an OS and the code for ~~Performance~~ performing these tasks resides in code memory.
- * Examples of Super loop based products are:
 - Reading/Writing to and from a card using card reader
 - An electronic video game toy containing key pad and display unit

Pros:

- * Doesn't require an operating system for task scheduling and monitoring and free from OS related overheads
- * Simple and straight forward design Reduced memory footprint.

Cons:

- * Non Real time in execution behaviour
- * Any issues in any task execution may affect the functioning of the product. If the program hangs up at ultimately the product stops functioning.

Enhancements:

- * Use of Hardware and Software Watch Dog Timers (WDT) helps in coming out from loop when an unexpected failure occurs or when the processor hangs up.
- * Combine Super loop based technique with interrupts and execute the tasks which require Real time attention as Interrupt Service routines.
- * Use of advanced processors/controllers greatly reduces the time required to service different tasks and thereby capable of providing a nearby real time attention to external events.

2 Explain in detail about Embedded OS based Approach for Embedded Firmware design

★ The embedded device contains an Embedded Operating System which can be one of:

- ★ A Real Time Operating System (RTOS)
- ★ A Customized General purpose Operating System (GPOS)
- ★ The Embedded OS is responsible for scheduling the execution of user tasks and the allocation of system resources among multiple tasks
- ★ It involves lot of OS related overheads apart from managing and executing user defined tasks.
- ★ The GPOS based design is very similar to a conventional PC based application development where the device contains an OS and user applications run on top of it.
- ★ Microsoft Windows XP Embedded is an example of GPOS for embedded devices.
- ★ Point of Sale (POS) terminals, Gaming Stations, Tablet PCs etc are examples of embedded devices running on embedded GPOS.
- ★ For developing applications on top of OS, OS supported APIs are used and Driver Software is required for different hardware present on board to communicate.

- * Real time Operating System (RTOS) based design approach is employed in embedded products demanding Real-time-Response.
- * RTOS contains a Real Time kernel responsible for performing pre-emptive Scheduling, multitasking, multi-threading etc
- * A RTOS allows flexible Scheduling of System resources like CPU and memory and offers some way to Communicate between tasks.
- * "Windows CE", "Windows Mobile", "QNX", "VxWorks", 'Thread x', 'Micro C/OS-II', 'Embedded Linux', 'Symbian' etc are examples of RTOSs employed in Embedded Product development.
- * Mobile Phones, PDA's, Flight Control Systems etc are examples of embedded devices that runs on RTOSs.

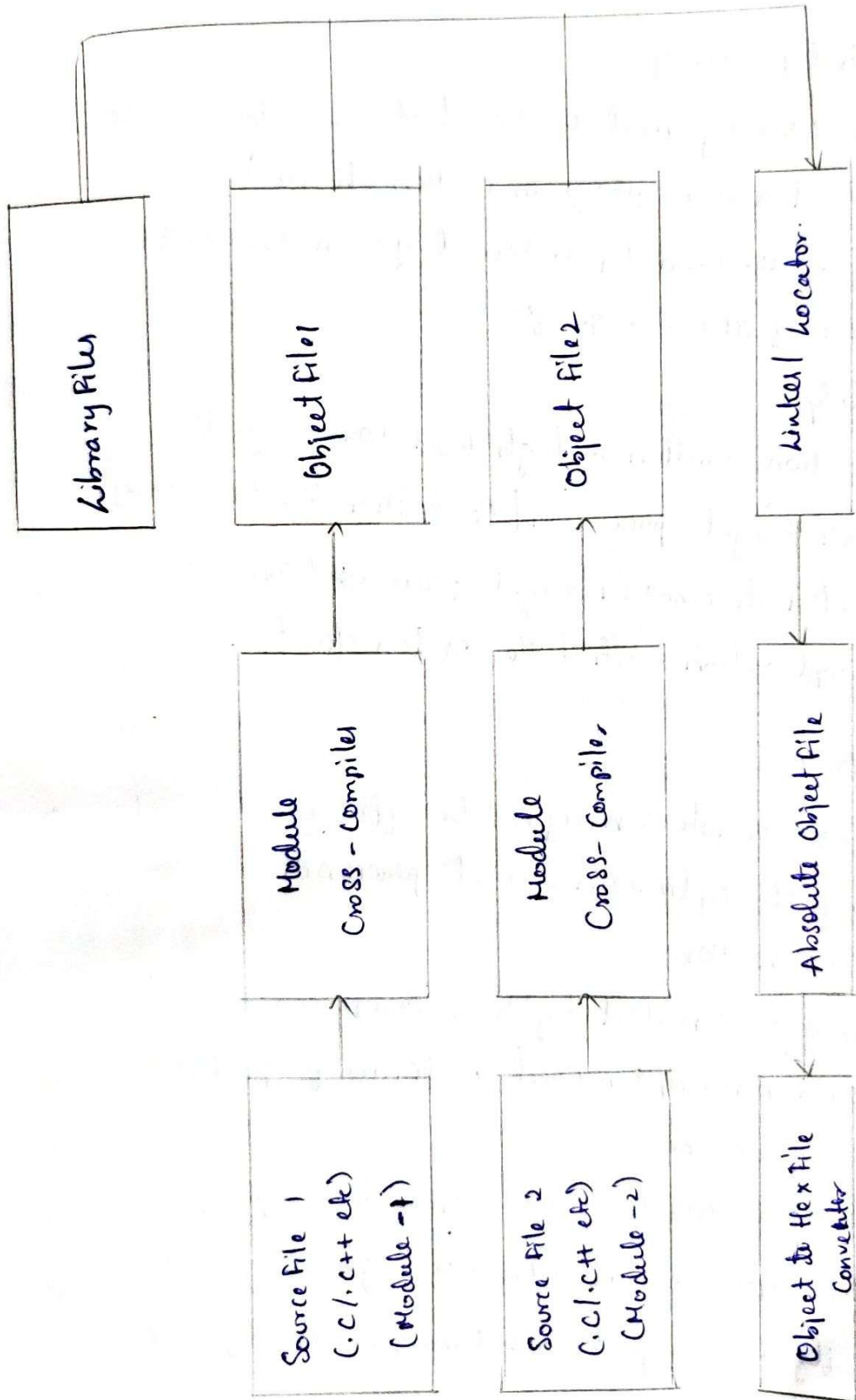
3(a) Explain the High-level language to machine language conversion process with neat sketch.

(b) Explain the advantages and limitations of the high-level language-based Development of Embedded Firmware

3a) HIGH LEVEL LANGUAGE:

- * The embedded firmware is written in any high level language like C, C++
- * A Software utility called 'Cross-compiler' converts the high level language to target processor specific machine code
- * The Cross-compilation of each module generates a corresponding Object file. The Object file does not contain ~~the corresponding object file~~ the absolute address of where the generated code needs to be placed ~~as re-locatable~~ on the program memory
- * The software program called linker/locator is responsible for assigning absolute address to object files during the linking process.
- * The Absolute Object file created from the object files corresponding to different source code modules contain information about the address where each instruction needs to be placed in code memory.

- * A Software Utility Called 'Object to Hex file Converter' translates the absolute object file to corresponding hex file



High level language to machine language conversion process.

Advantages:-

★ Reduced Development time:

Developer requires less or little knowledge on internal hardware details and architecture of the target processor/ Controller.

★ Developer independency:

The Syntax used by most of the high level languages are Universal and a program written high level can easily understand by a second person knowing the Syntax of the language.

★ Portability:-

An Application written in high level language for particular target processor/ Controller can be easily be converted to another target processor/ Controller specific application with little or less effort.

Drawbacks:-

★ The Cross Compilers may not be efficient in generating the optimized target processor specific instructions.

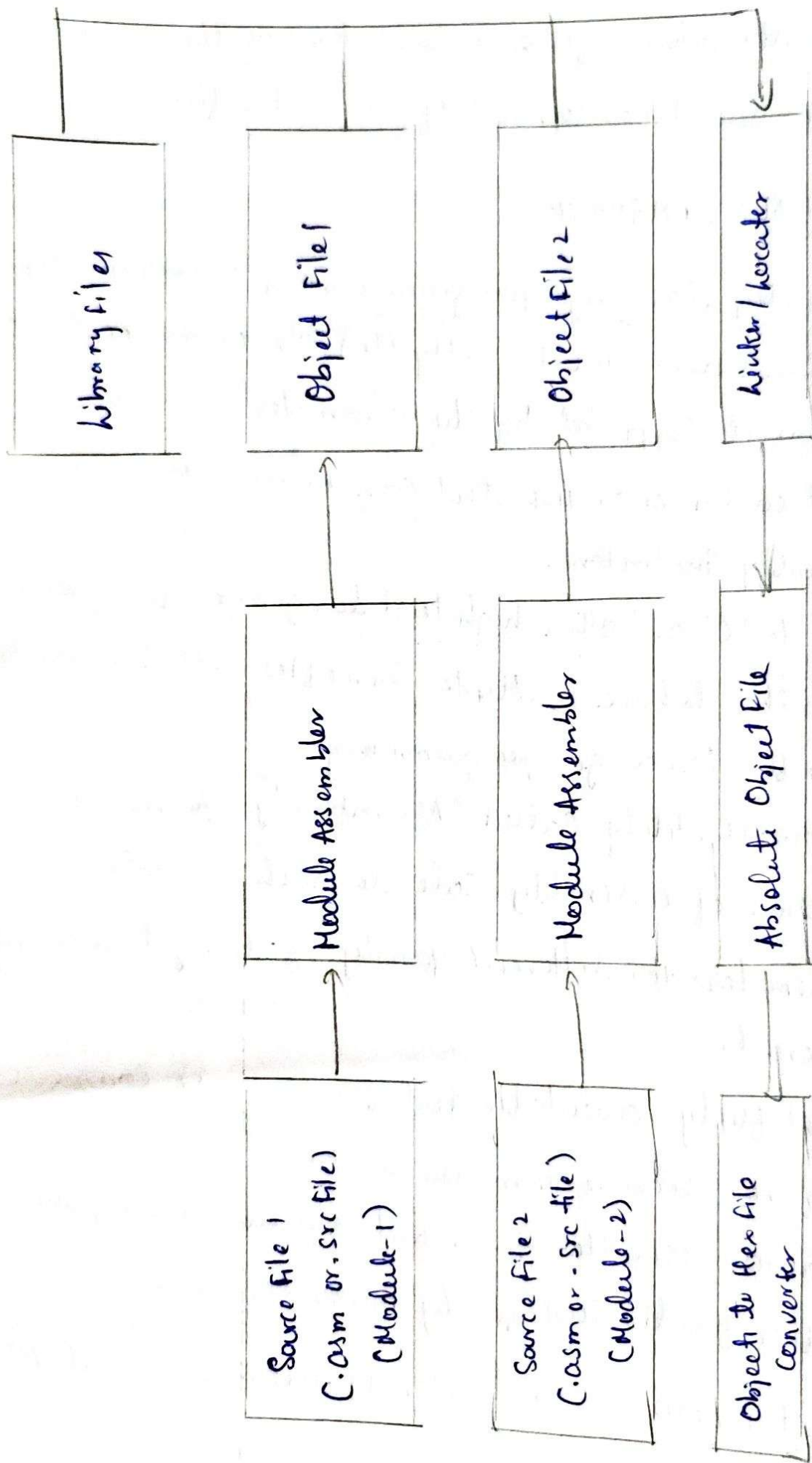
★ Target images created by such compilers may be messy and non-optimized in terms of performance as well as code size.

★ The investment required for high level language based development tools (IDE) is high compared to Assembly Language based firmware development tools.

- 4a) Explain the Assembly language to machine language Conversion process with neat Sketch.
- b) Explain the advantages and limitations of the Assembly language-based Development of embedded firmware.

Ans:- ASSEMBLY LANGUAGE

- * The Assembly language program written in assembly code is saved as .asm (Assembly file) file or a .src (Source) file or a format supported by the assembler.
- * Any text editor or an IDE tool can be used for writing the assembly instructions.
- * Similar to 'C' and other high level language programming, it is possible to have multiple source files called modules in assembly language programming.
- * The software utility called "Assembler" performs the translation of assembly code to machine code.
- * The assemblers for different family of target machines are different.
- * Some are freely available and some are commercial and requires license from the vendor.
- * ASI Macro Assembler from Keil Software is a popular assembler for the 8051 family micro controller.
- * Microsoft macro assembler, MASM Software is used for 8086 processor.



Assembly language to machine language
Conversion process -

- * Each source file can be assembled separately to examine the syntax errors and incorrect assembly instructions.
- * Assembling of each source file generates a corresponding object file.
- * The object file does not contain the absolute address of where the generated code needs to be placed on the program memory.
- * The software program called linker/loader is responsible for linking various object modules in a multi-module project and assigning absolute address to object files during the linking process.
- * The Absolute object file created from the object files corresponding to different source code modules contain information about the address where each instruction needs to be placed in code memory.
- * A software utility called 'Object to Hex file Converter' translates the absolute object file to corresponding hex file (binary file).
- * Hex files are ASCII files that contain a hexadecimal representation of target applications.

b) Advantages:

- * Efficient code memory & data memory usage (Memory Optimization).
- * The developer is well aware of the target processor architecture and memory organization, so optimized

Code can be written for performing operations.

- * This leads to less utilization of code memory and efficient utilization of data memory.

- * High performance

- Optimised code not only improves the code memory usage but also improves the total system performance.

- Through effective assembly coding, optimum performance can be achieved for target processor.

- * Low level Hardware Access.

- Most of the code for low level programming like accessing external device specific registers from OS kernel, device drivers, and low level interrupt routines, etc are making use of direct assembly coding.

- * Code Reverse Engineering:-

- It is the process of understanding the technology behind a product by extracting the information from the finished product.

- It can easily be converted into assembly code using a dis-assembler program for the target machine.

Drawbacks:

- High Development Time
 - The developer takes lot of time to study about architecture, memory organization, addressing modes and instruction set of assembly code is required for performing a simple action.
- Developer dependency:
 - There is no common written rule for developing assembly language based applications.
- Non portable:
 - Target applications written in assembly instructions are valid only for that particular family of processors and ~~valid~~ cannot be re-used for another target processors/controllers.
 - If the target processor/controller changes a complete re-writing of the application using assembly language for new target processor/controller is required.