Microcontroller and Embedded Systems

18CS44

Module - 4

MODULE 4

Embedded System Design Concepts: Characteristics and Quality Attributes of Embedded Systems, Operational quality attributes, non-operational quality attributes, Embedded Systems-Application and Domain specific, Hardware Software Co-Design and Program Modelling, embedded firmware design and development

Text book 2: Chapter-3, Chapter-4, Chapter-7 (Sections 7.1, 7.2 only), Chapter-9 (Sections 9.1, 9.2, 9.3.1, 9.3.2 only)

CHARACTERISTICS OF AN EMBEDDED SYSTEM

1. Application and domain specific
2. Reactive and real time
3. Operates in harsh environments
4. Distributed
5. Small size and weight
6. Power concerns

Application and domain specific

- Each embedded system is having certain function to perform and they are developed in such a manner to do the intended functions only.
- For example, we cannot replace the embedded control unit of your micro oven with air conditioner’s embedded control unit, because the embedded control units of micro oven and air conditioner are specifically designed to perform certain specific tasks.

Reactive and real time

- Any changes happening in the real world is captured by the sensor input devices in real time and control algorithm running inside the unit reacts in a designed manner to bring the controlled output variables to the desired level.
- Embedded systems produces changes in the output as changes in the input. So they are generally referred as Reactive systems.
- Real time system operation means the timing behavior of the system should be deterministic; meaning the system should respond to request or tasks in a known amount of time.

Operates in harsh environment

- The environment in which the embedded system deployed may be a dusty one or a high temperature zone or an area subject to vibrations and shock.
- System placed in such areas should be capable to withstand all these adverse operating conditions. The design should take care of the operating conditions of the area where the system is going to implement.

Distributed

- The term distributed means that embedded system may be a part of larger system. Many numbers of such distributed embedded system form a single large embedded control unit.
- An automatic vending machine is a typical example for this. The vending machine contains a card reader, a vending unit, etc. Each of them are independent embedded units but they work together to perform the overall Vending function.

Small size and weight

Power concerns

- Embedded systems should be designed in such a way as to minimize the heat dissipation by the system.
- The product of high amount of heat demands cooling requirements like cooling fans which in turn occupies additional space and make the system bulky.

Quality attributes of embedded systems

- Operational quality attributes
- Non - Operational quality attributes

Operational quality attributes

- the quality attributes related to the embedded systems when it is in the operational mode or ‘online’ mode.

1. Response
2. Throughput
3. Reliability
4. Maintainability
5. Security
6. Safety

Response

- Response is a measure of quickness of the system. It gives an idea about how fast your system is tracking the changes in input variables.

Throughput

- it can be defined as the rate of production or operation of a defined process over a started period of time.

Reliability

- Reliability is a measure of how much % you can rely upon the proper functioning of the system or what is the % susceptibility of the system to failures.
- Mean time between failures (MTBF) and mean time to repair (MTTR) are the terms used in defining system reliability.
- MTBF gives the frequency of failures in hours/weeks/months.
- MTTR specifies how long the system is allowed to be out of order following a failure.

Maintainability

- maintainability deals with support and maintenance to the end user or client in case of technical issues and product failure or on the basis of a routine system check-up.
- A more reliable system means a system with less corrective maintainability requirements and vice versa.

Security

- deals with the protection of data and application from unauthorised disclosure, unauthorized modification, unauthorized users

Safety

- Safety deals with the possible damage that can happen to the operators, public and the environment due to the breakdown of an embedded system or due to the emission of radioactive or hazardous material from the embedded products.

Non-Operational Quality Attributes

1. Testability & Debug-ability
2. Evolvability
3. Portability
4. Time to prototype and market
5. Per unit and total cost.

Testability & Debug-ability

- Application and by which means he/she can test it for an embedded products testability is applicable to both the embedded hardware and firmware
- debug ability is a means of debugging the product as such for figuring out the probable sources that creating the unexpected behaviour in the total system.

Evolvability

- Evolvability referred as the non-heritable variation. (Capacity of a system for adaptive evolution)

Portability

- Portability is a measure of system independence.
- An embedded product is said to be portable if the product is capable of functioning; as such in various environments, target processors/controllers and embedded operating system
- A standard embedded product should always be flexible and portable.

Time-to-Prototype and Market

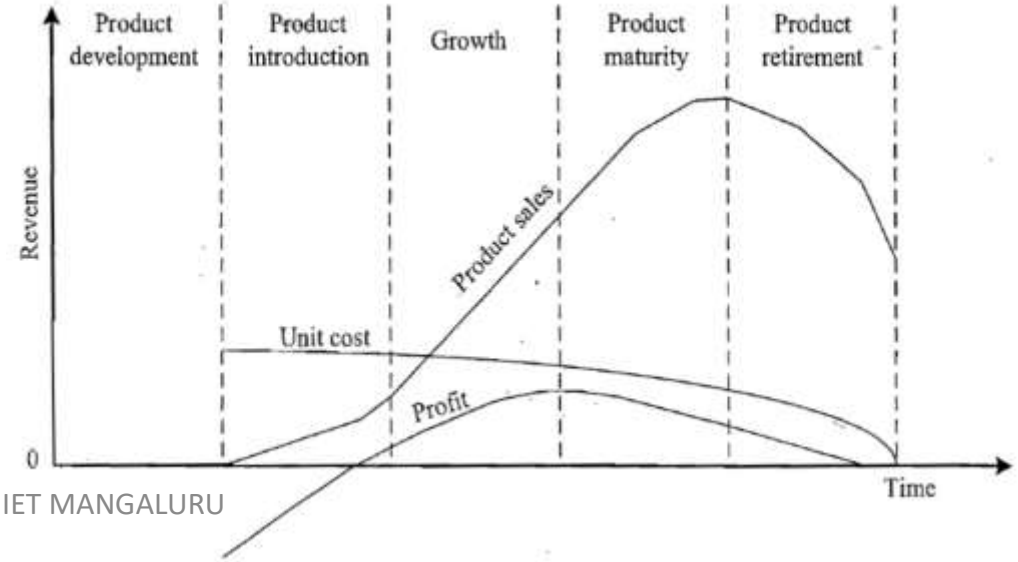
- Time-to-market is the time elapsed between the conceptualization of a product and the time at which the product is ready for selling or use.

Per Unit Cost and Revenue

- Cost is a factor which is closely monitored by both end user (those who buy the product) and product manufacturer (those who build the product).
- Cost is a highly sensitive factor for commercial products.

Product Life-cycle (PLC)

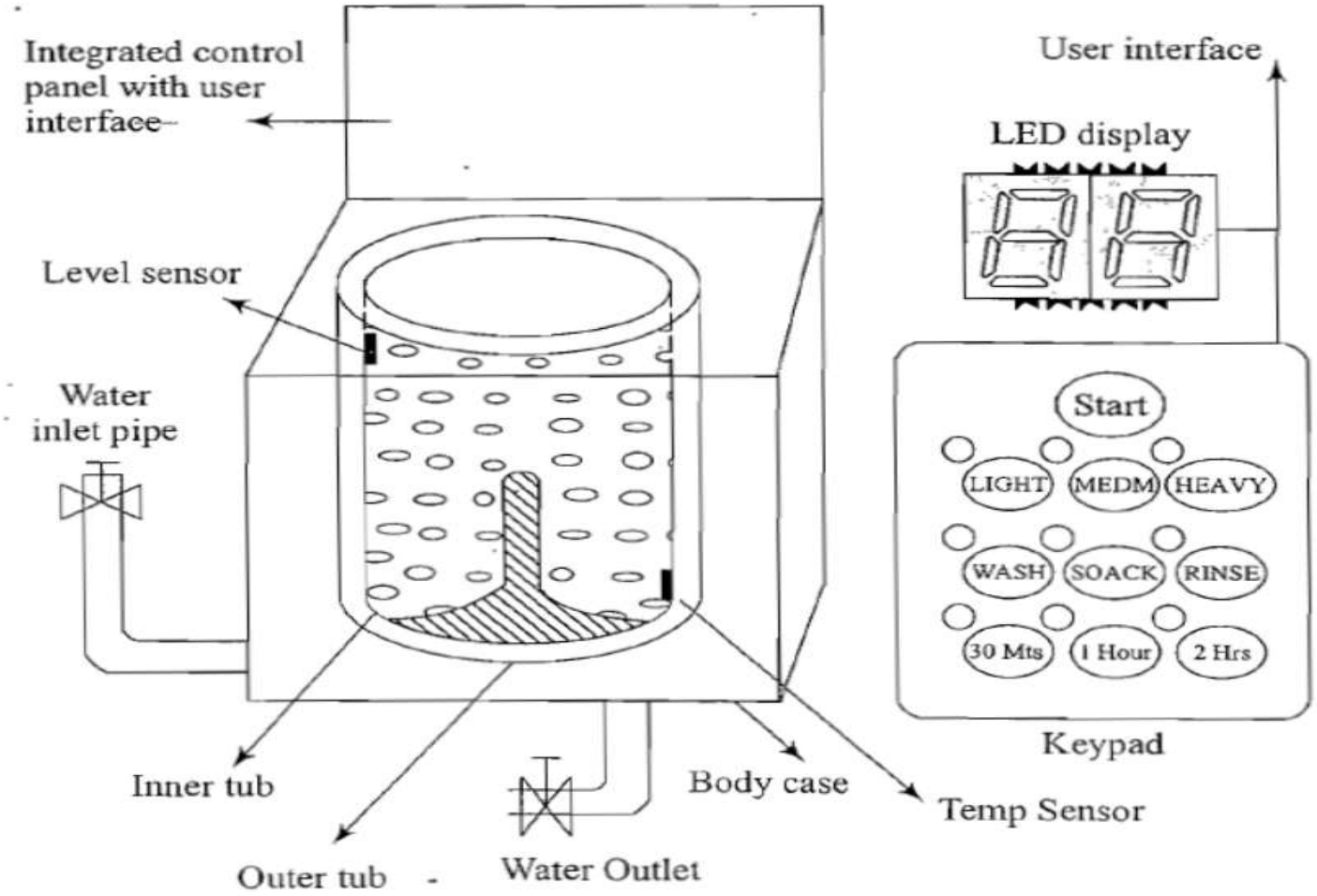
- The product idea generation, prototyping, definition, actual product design and development. Exponent are the activities carried out during this phase.
- During the design and development phase there is only investment and no returns once the product is ready to sell, it is introduced to the market. This stage is known as the Product Iteration stage.
- The Product Retirement/Decline phase starts with the drop in sales volume, market share and revenue the/decline happens due to various reasons like competition from similar product with enhanced features or technology changes, etc.



EMBEDDED SYSTEMS -APPLICATION –AND DOMAIN-SPECIFIC

- Embedded systems are application and domain specific, meaning; they are specifically built for certain applications in certain domains like consumer electronics, telecom, automotive, industrial control, etc.
- In general purpose computing, it is possible to replace a system with another system which is closely matching with the existing system, whereas it is not the case with embedded systems, hence it is not possible to replace an embedded system developed for a specific application.

washing machine-application-specific embedded system



Automotive–domain-specific examples of embedded system

- The presence of automotive embedded system in a vehicle varies from simple mirror and wiper controls to complex air bag controller and antilock brake systems (ABS).
- Automotive embedded systems are normally built around microcontroller or DSPs or a hybrid of the two and are generally known as Electronic Control Units (ECUs).
- The various types of electronic control units (ECUs) used in the automotive embedded industry can be broadly classified into two-
  - High-speed embedded control units
  - Low-speed embedded control unit

High-Speed Electronic Control Units (HECUs)

- High-speed electronic control units (HECUs) are deployed in critical control units requiring fast response.
- They include fuel injection systems, antilock brake systems, engine control, electronic throttle, steering controls, transmission control unit and central control unit.

Low-speed Electronic Control Unit (LECUs)

- Low-Speed Electronic Control Units (LECUs) are deployed in application where response time is not so critical.
- Audio controllers, passenger and driver door locks, door glace controls (power windows), wiper control are examples of LECUs

Automotive Communication Buses

- Automotive applications make use of serial buses for communication.
- Which greatly reduces the amount of wiring required inside a vehicle.
- The following are the different types of serial interface buses deployed in automotive embedded applications.
    - CAN
    - LIN
    - MOST

Controller Area Network (CAN)

- It is generally employed in safety system like airbag control; power train systems like engine control and Antilock Brake System (ABS); and navigation systems like GPS.

Local Interconnect Network (LIN)

- LIN is a low speed.
- LIN bus is employed in applications like mirror controls, fan controls, seat positioning controls, window controls, and position controls where response time is not a critical issue.

Media-Oriented System Transport (MOST)

- Bus the Media-oriented system transport (MOST) is targeted for automotive audio/video equipment interfacing

Key Players of the Automotive Embedded Market

The key players of the automotive embedded market can be visualized in three verticals namely,

silicon providers,

solution providers

tools and platform providers.

Silicon Providers

- Silicon providers are responsible for providing the necessary chips which are used in the control application development.
- **Analog Device：** Provider of world class digital signal processing chips, precision analog microcontrollers. programmable inclinometer/accelerometer, LED drivers, etc. for automotive signal processing applications, driver assistance system, audio system, GPS/Navigation system etc.
- **Xilins：** Supplier of high-performance FPGAs, CPLDs and automotive specific IP cores for GPS navigation systems. driver information systems, distance control, collision avoidance, rear seat entertainment. adaptive cruise control, voice receptionist. etc.
- **Atmel** : Supplier of cost-effective high-density Flash controllers and memories. Atmel provides a series of high-performance microcontrollers, namely, ARM1, ARM2, and 80C51. A wide range of Application Specific Standard Products (ASSPs) for chassis, body electronics, security, safety and car infotainment and automotive networking products for CAN, LIN and FlexRay are also supplied by Atmel.
- **NXP semiconductor** : Supplier of 8/16/32 Flash microcontrollers**.**
- **Texas Instruments：** Supplier of microcontrollers, digital signal processors and automotive communication control chips for Local Inter Connect (LIN) bus products.

Solution Providers

Solution providers supply complete solution for automotive applications making use of the chips, platforms and different development tools.

- **Bosch Automotive :** body electronics, diesel engine control, gasoline engine control, powertrain systems, safety systems, in-car navigation systems and infotainment systems.
- **DENSO Automotive :** engine management, climate control, body electronics, driving control & safety, hybrid vehicles, embedded infotainment and communications.
- **Infosys Technologies** (WWW. infosys. com): Infosys is a solution provider for automotive embedded hardware and software
- **Delphi** (www.delphi.com): Delphi is the complete solution provider for engine control, safety, infotainment, etc., and OEM for spark plugs, bearings, etc

Tools and Platform Providers

- Tools and platform providers are manufacturers and suppliers of various kinds of development tools and Real Time Embedded Operating Systems for developing and debugging different control unit related applications.
- Tools fall into two categories, namely embedded software application development tools and embedded hardware development tools.

  - **The Math Works**
  - **Keil Software**
  - **Microsoft**

Hardware Software Co-Design and Program Modelling

- In the traditional embedded system development approach, the hardware software partitioning is done at an early stage and engineers from the software group take care of the software architecture development and implementation, whereas engineers from the hardware group are responsible for building the hardware required for the product.
- There is less interaction between the two teams and the development happens either serially or in parallel. Once the hardware and software are ready, the integration is performed. The increasing competition in the commercial market and need for reduced ‘time-to-market’ the product calls for a novel approach for embedded system design in which the hardware and software are co-developed instead of independently developing both.

- During the co-design process, the product requirements captured from the customer are converted into system level needs or processing requirements. At this point of time it is not segregated as either hardware requirement or software requirement, instead it is specified as functional requirement.
- The system level processing requirements are then transferred into functions which can be simulated a verified against performance and functionality.
- The Architecture design follows the system design.

Fundamental issues in hardware software co-design

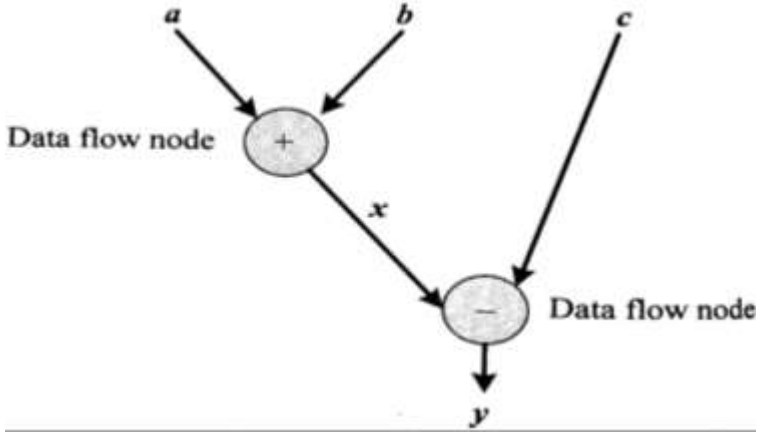
- Selecting the model (Model is a formal system consisting of objects and composition rules)
- Selecting the Architecture(The architecture specifies how a system is going to implement in terms of the number and types of different components and the interconnection among them)
  - **controller architecture** (state register holds the present state and the combinational circuits implement the logic for next state and output.)
  - **data path architecture** (output is generated as a result of a set of predefined computations on the input data.)
  - **Finite State Machine Data path** :combines the controller architecture with data path architecture.
  - **Complex Instruction Set Computing**
  - **Reduced Instruction Set Computing**
  - **Parallel processing architecture** (SIMD, MIMD)
- Selecting the language
- Partitioning System Requirements into hardware and software

Computational models in embedded design

- Data Flow Graph (DFG) model
- State Machine model
- Concurrent Process model
- Sequential Program model
- Object Oriented model

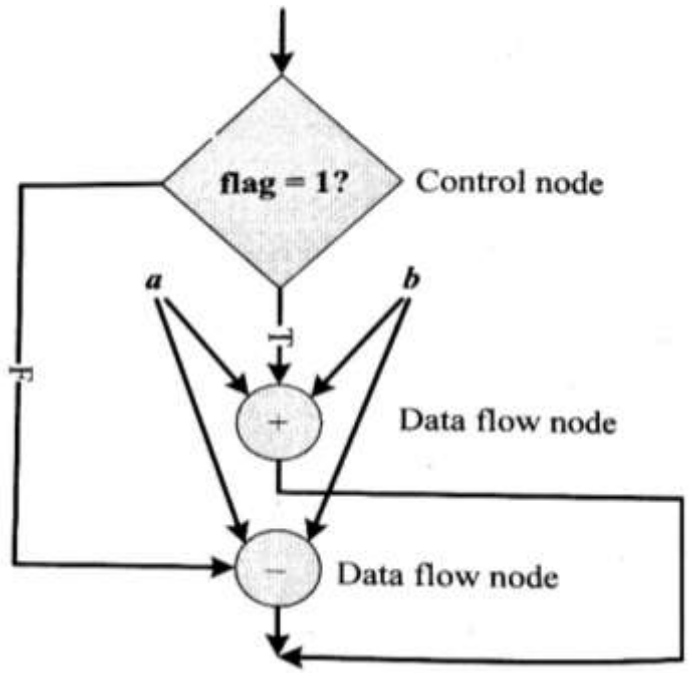
Data Flow Graph/Diagram (DFG) Model

- The Data Flow Graph (DFG) model translates the data processing requirements into a data flow graph.
- This model emphasis on the data and operations on the data which transforms the input data to output data.
- Indeed, Data Flow Graph (DFG) is a visual model in which the operation on the data (process) is represented using a block (circle) and data flow is represented using arrows.
- An inward arrow to the process (circle) represents input data and Data flow node an outward arrow from the process (circle) represents output data in DFG notation.
- x = a + b; and y=x - c.



Control Data Flow Graph/Diagram (CDFG)

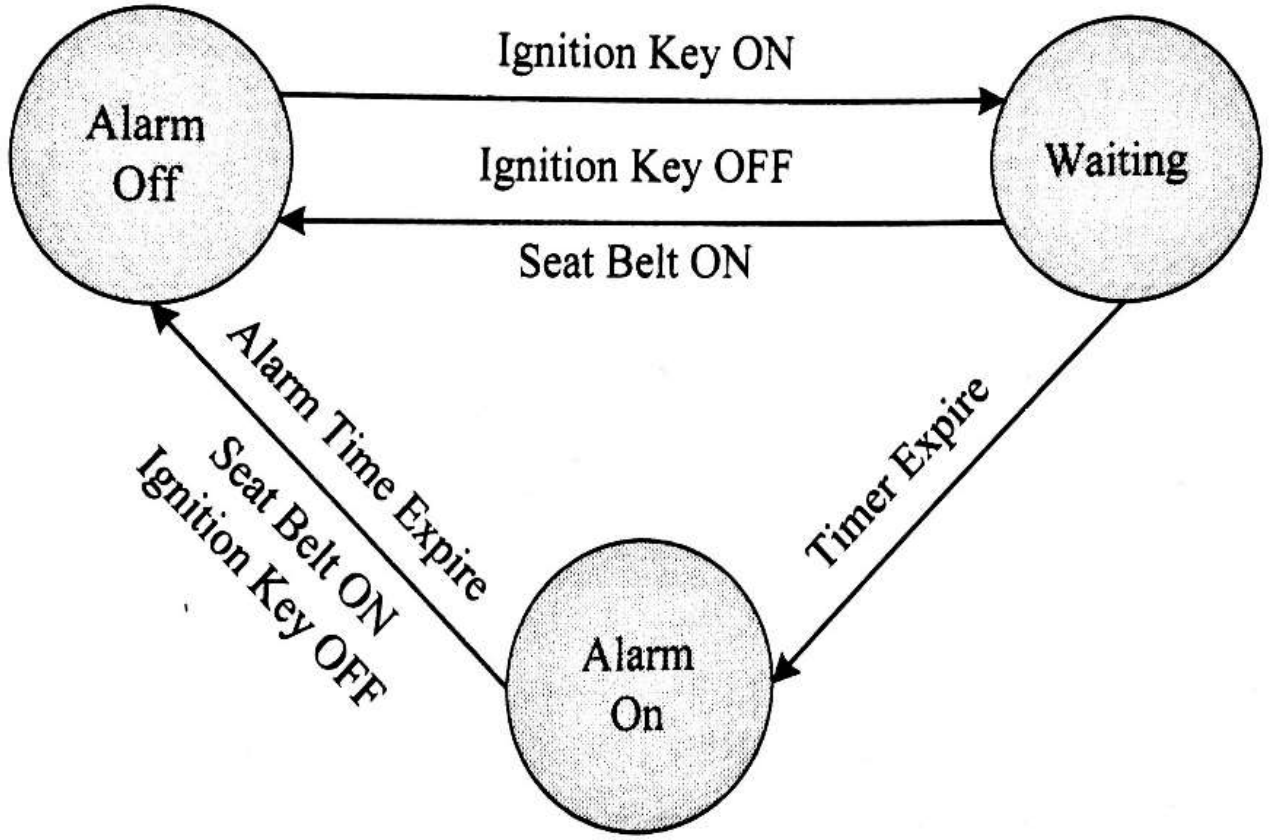
- CDFG contains both data flow nodes and decision nodes, whereas DFG contains only data flow nodes
- if flag=1, x=a +b; else y=a-b;
- The control node is represented by a ‘Diamond’ block which is the decision making element in formal flow chart based design.

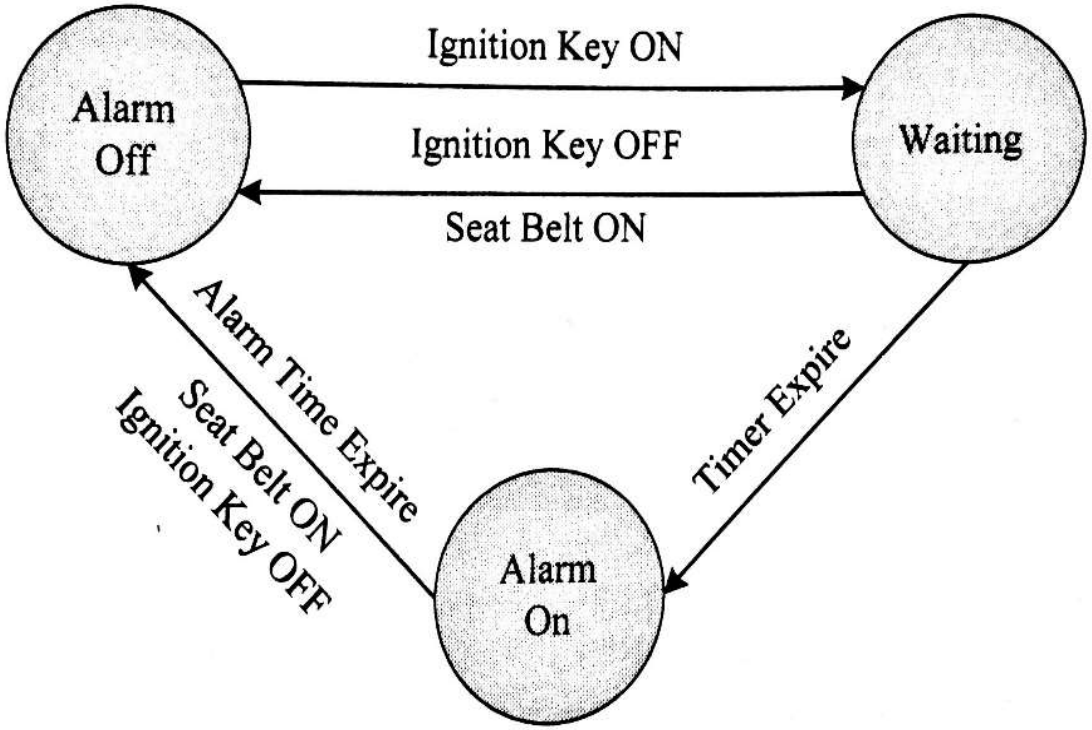


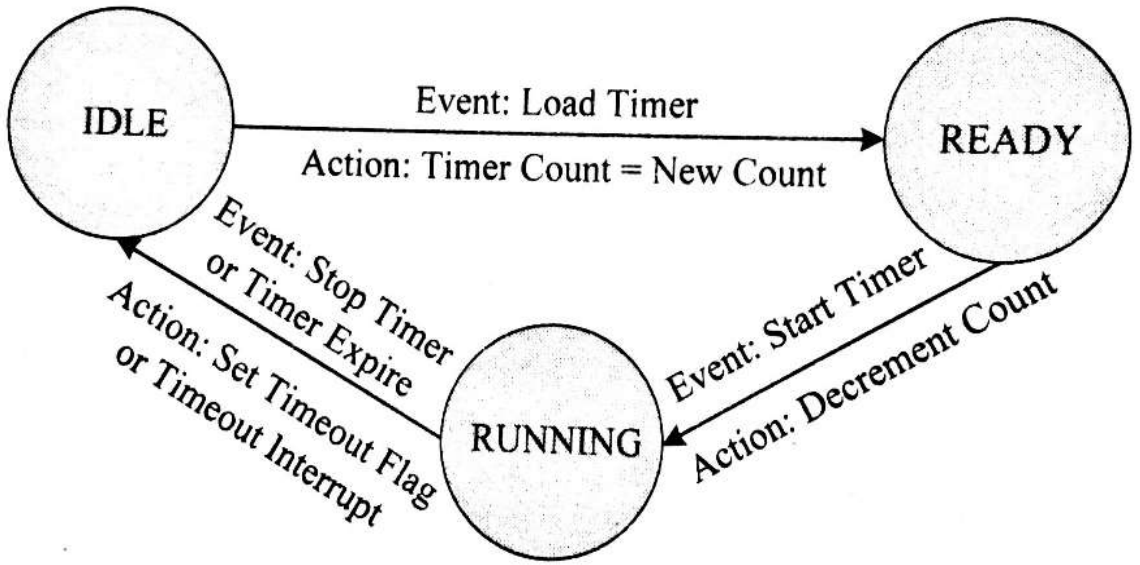
State Machine Model

- The State Machine model is used for modelling reactive or event-driven embedded systems whose processing behaviour are dependent on state transitions.
- The State Machine model describes the system behaviour with 'States', 'Events', 'Actions' and 'Transitions'.
- State is a representation of a current situation.
- An event is an input to the state. The event acts as stimuli for state transition.
- Transition is the movement from one state to another.
- Action is an activity to be performed by the state machine.
- A Finite State Machine (FSM) model is one in which the number of states are finite.

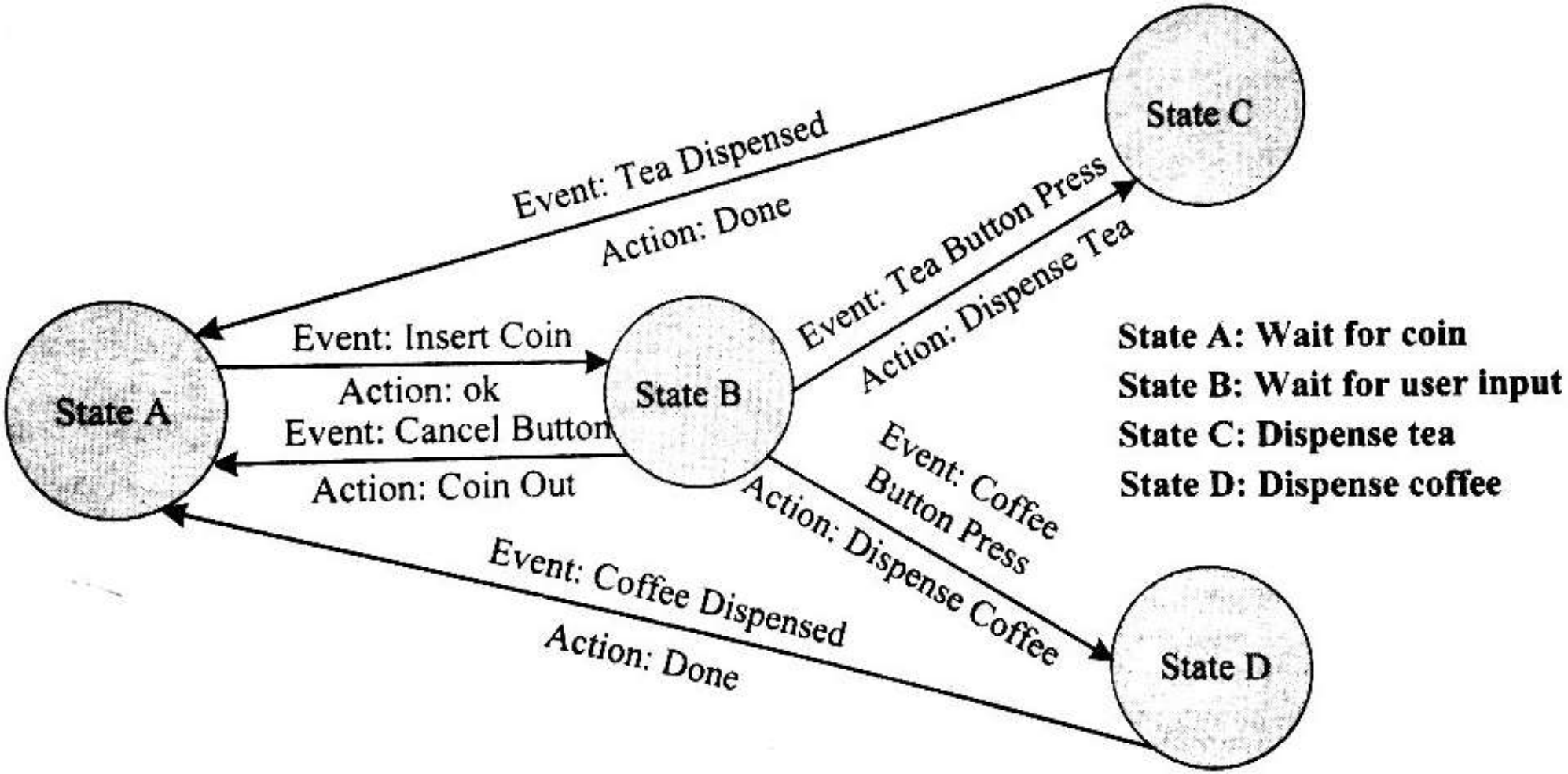
Design of an embedded system for driver/passenger 'Seat Belt Warning' in an automotive using the FSM model.



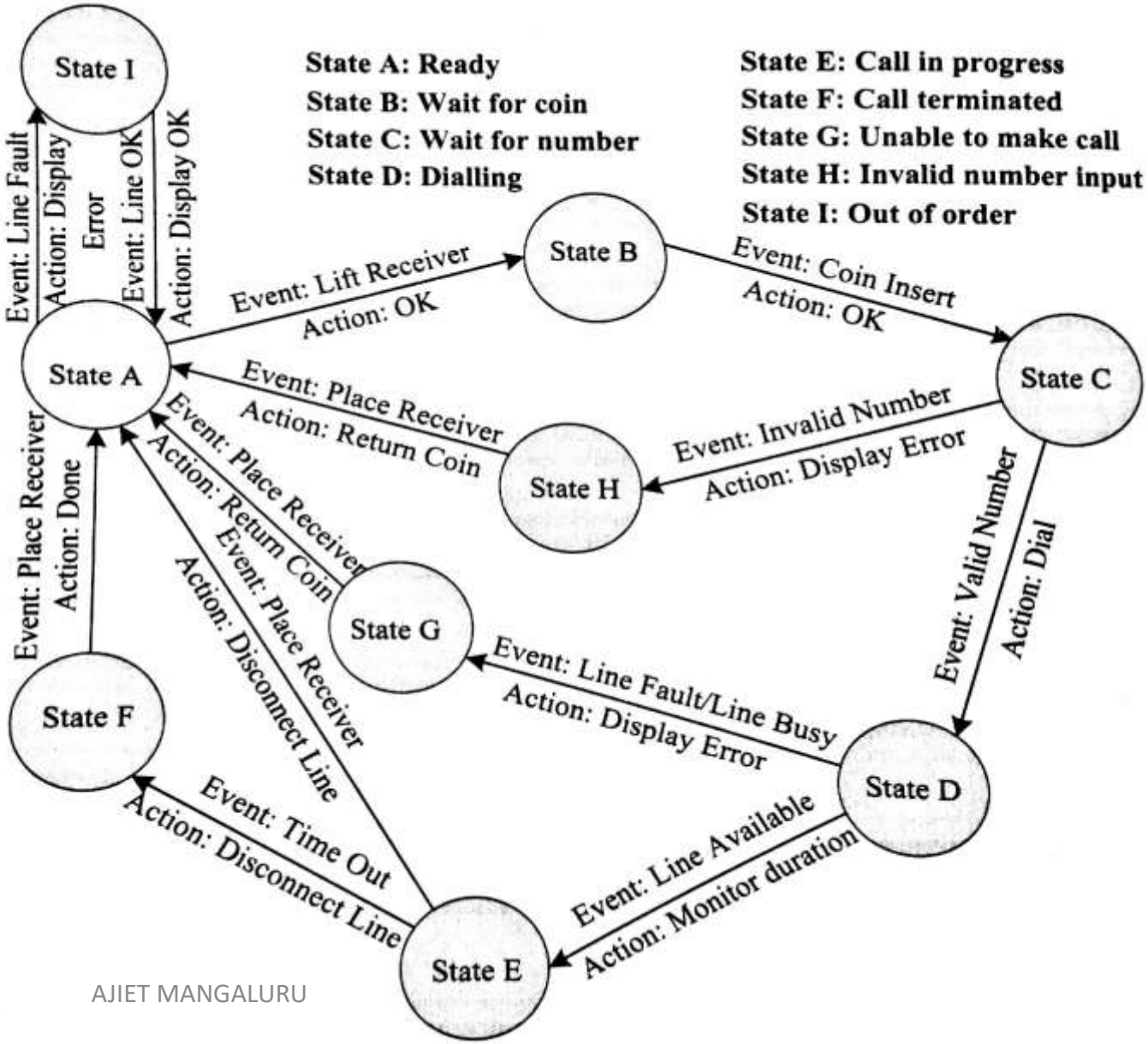

Alarm Off
Waiting
Alarm On
Ignition Key ON
Ignition Key OFF
Seat Belt ON
Alarm Time Expire
Seat Belt ON
Ignition Key OFF
Timer Expire


IDLE
READY
RUNNING
Event: Load Timer
Action: Timer Count = New Count
Event: Stop Timer
or Timer Expire
Action: Set Timeout Flag
or Timeout Interrupt
Event: Start Timer
Action: Decrement Count

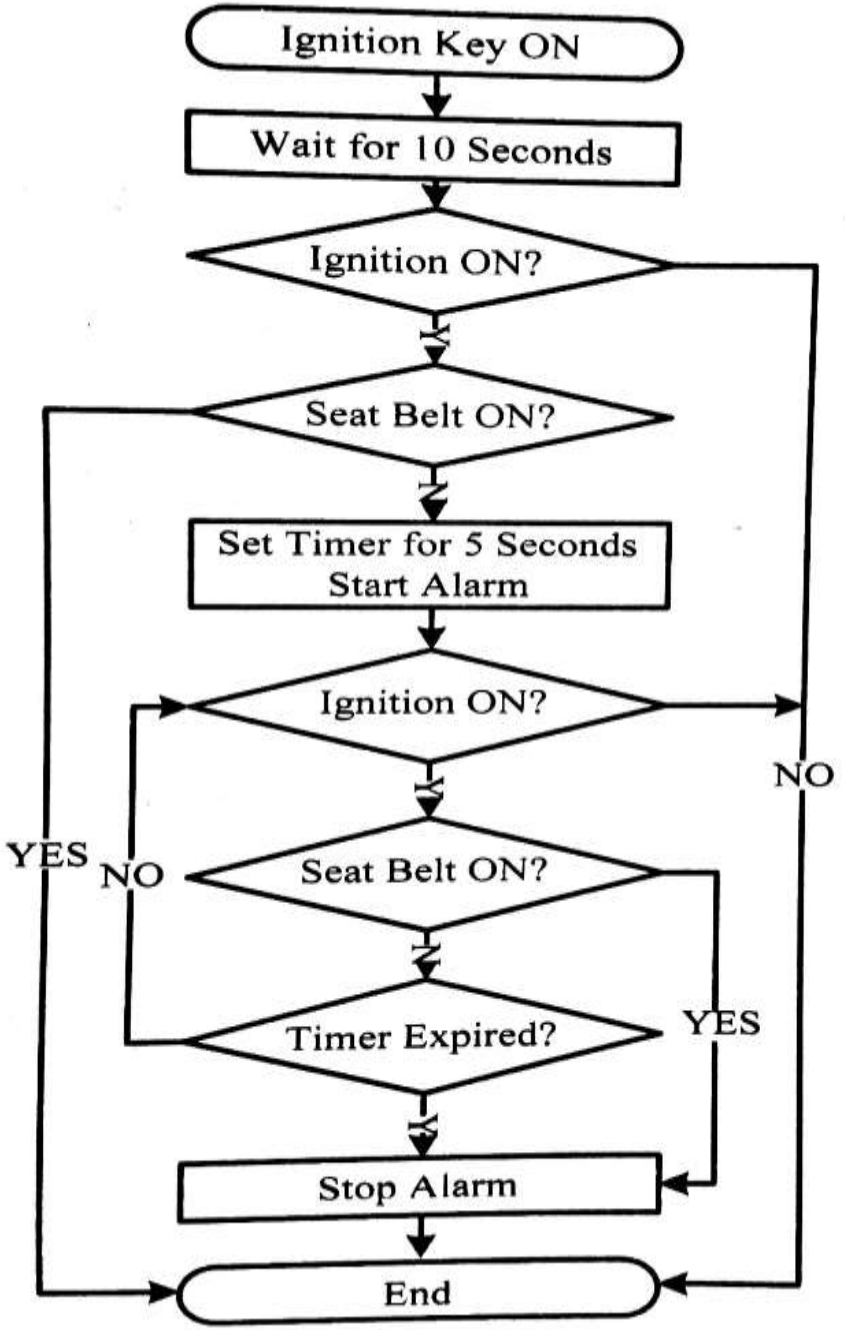
Design an automatic tea/coffee vending machine based on FSM model for the following requirement.



- Design a coin operated public telephone unit based on FSM model for the following requirements.



Sequential Program Model



Embedded Firmware Design and Development

- The embedded firmware is responsible for controlling the various peripherals of the embedded hardware

- In case of hardware breakdown, the damaged component may need to be replaced by a new component and for firmware corruptions the firmware should be reloaded, to bring back the embedded product to the normal functioning.

- For most of the embedded products the embedded firmware is stored at a permanent memory (ROM) and they are no alterable by end users.

- Designing embedded firmware requires understanding of the particular embedded product hardware

Embedded firmware design approaches

- The firmware design approaches for embedded product is purely dependent on the complexity of the functions to be performed, the speed of operation required, etc.
- Two basic approaches are used for embedded firmware design.
    - **‘conventional procedural Based Firmware Design’(‘Super loop Model’)**
    - **‘Embedded Operating (OS) based design’.**

The Super Loop Based Approach

- The Super Loop based firmware development approach is adopted 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.
- Each task is executed in serial in this approach.
- the tasks 1 to n are performed one after another and when the last task (nth task) is executed, the firmware execution is again re-directed to Task 1 and it is repeated forever in the loop.
- Since the tasks are running inside an infinite loop, the only way to come out of the loop is either a hardware reset or an interrupt assertion.

- The ‘Super loop based design' doesn’t ‘ require an operating system
- This type of design is deployed in low-cost embedded products and products where response time is not time critical.
- A typical example of a ‘Super loop based’ product is an electronic video game toy containing keypad and display unit

Drawback

- any failure in any part of a single task will affect the total system.
- If the program hangs up at some point while executing a task, it will remain there forever and ultimately the product stops functioning. (Use of Hardware and software Watch Dog Timers)
- lack of real timeliness.

The Embedded Operating System (OS) Based Approach

- General Purpose Operating System (GPOS) or a Real Time Operating System (RTOS).
- **The General Purpose OS (GPOS)** based design is very similar to a conventional PC based application development where the device contains an operating system (Windows/Unix/ Linux, etc. for Desktop PCs) and you will be creating and naming user applications on top of it
- **Real Time Operating System (RTOS)** based design approach is employed in embedded products demanding Real-time response.
- 'Windows CE', 'pSOS', 'VxWorks', 'ThreadX', 'MicroC/OS-II', 'Embedded Linux', 'Symbian' etc. are examples of RTOS employed in embedded product development.
- Mobile phones, PDAs (Based on Windows CE/Windows Mobile Platforms), handheld devices, etc. are examples of 'Embedded Products' based on RTOS.

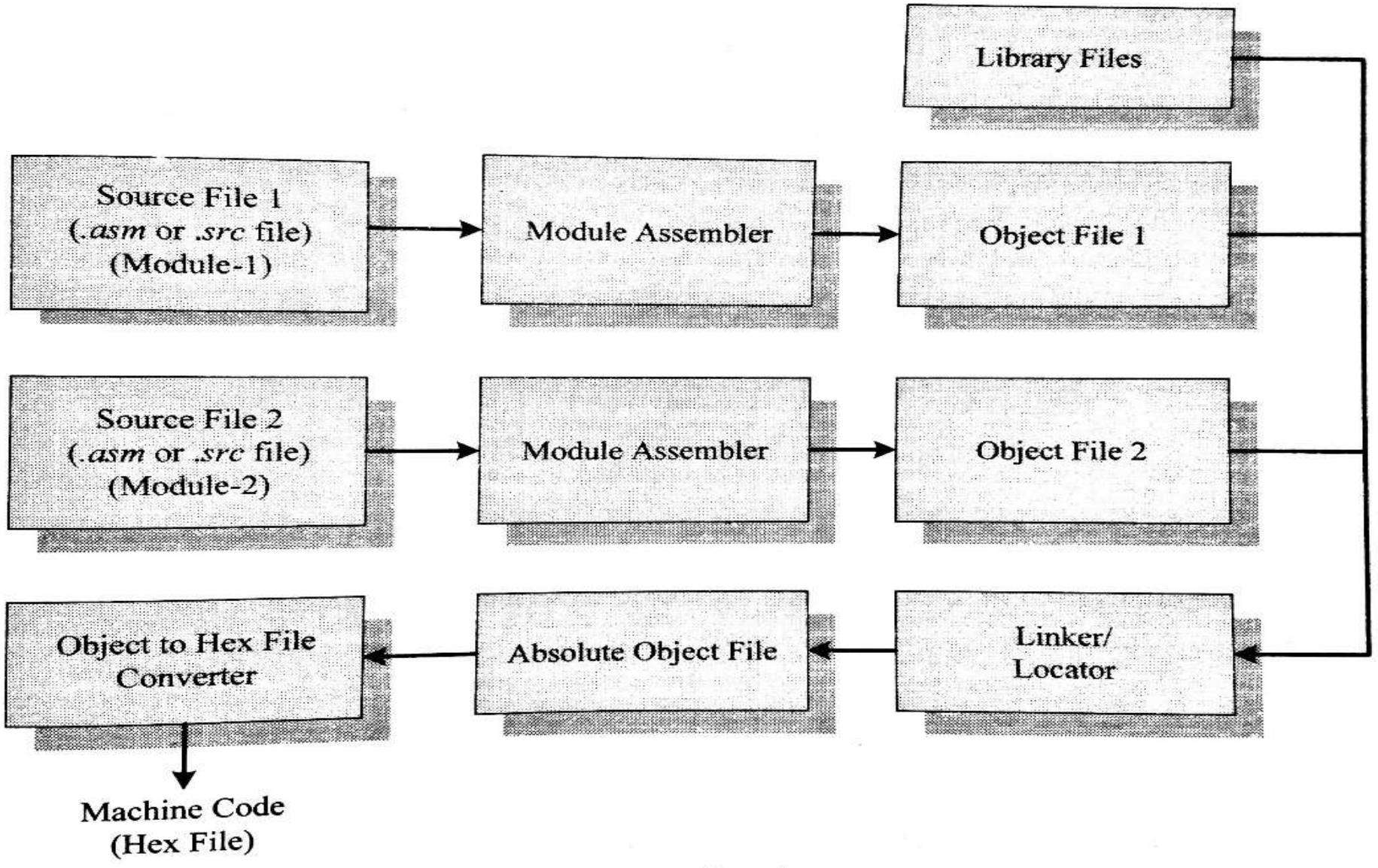
Embedded firmware development languages

- Assembly language or low level language
- High Level Language
- combination of Assembly and High level Language.

Assembly Language based Development

- Assembly language' is the human readable notation of 'machine language', whereas 'machine language' is a processor understandable language.
- Each line of an assembly language program is split into four fields as given below

LABEL OPCODE OPERAND COMMENTS


Library Files
Source File 1
(.asm or .src file)
(Module-1)
Module Assembler
Object File 1
Source File 2
(.asm or .src file)
(Module-2)
Module Assembler
Object File 2
Object to Hex File
Converter
Absolute Object File
Linker/
Locator
Machine Code
(Hex File)

Advantages

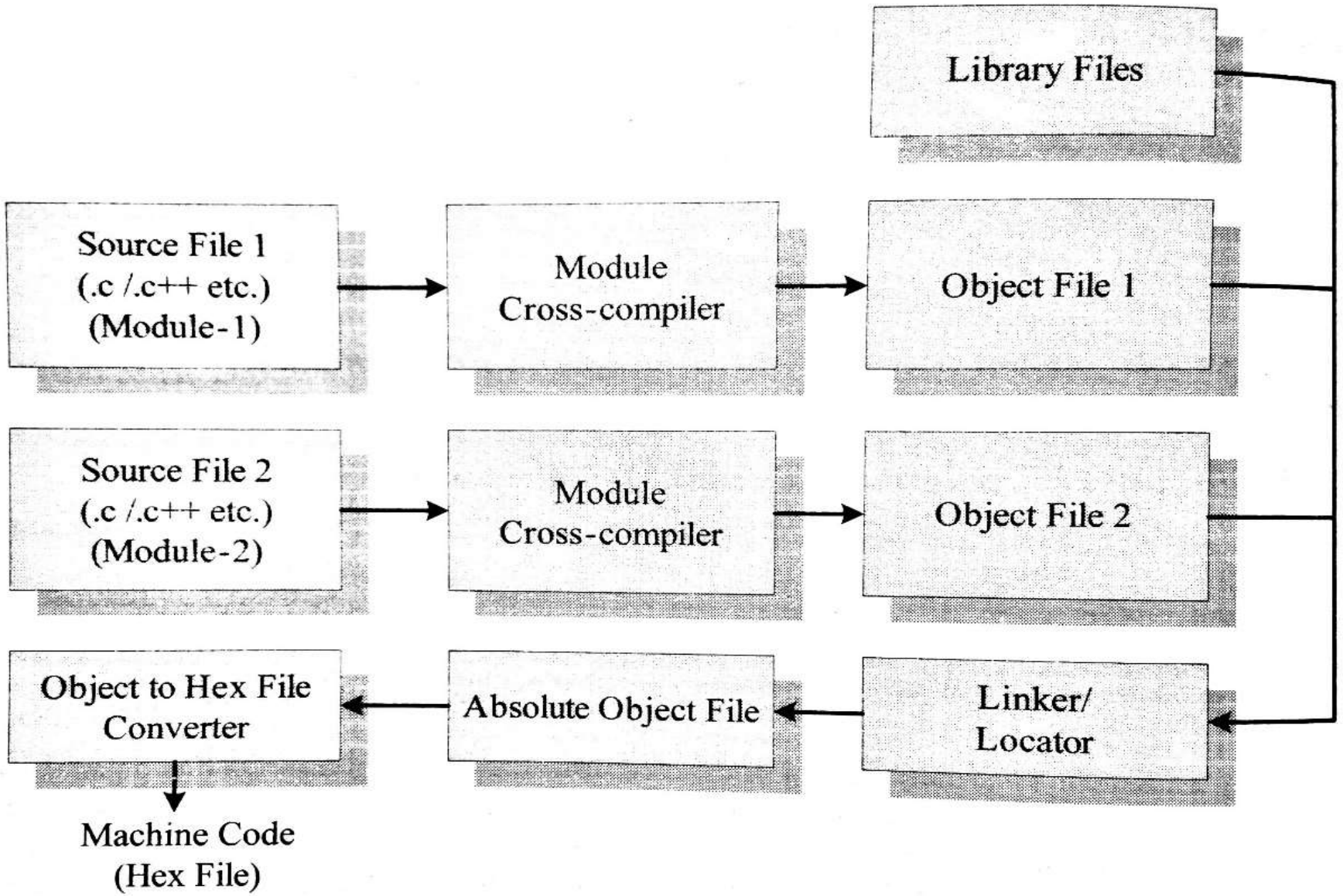
- Efficient Code Memory and Data Memory Usage (Memory Optimisation)
- High Performance
- Low Level Hardware Access

Drawbacks

- High Development Time
- Developer Dependency
- Non-Portable

- Assembly language based programing is highly time consuming, tedious and requires skilled programmers with sound knowledge of the target processor architecture. Also applications developed in Assembly language are non-portable

High Level Language Based Development


Library Files
Source File 1
(.c /.c++ etc.)
(Module-1)
Module
Cross-compiler
Object File 1
Source File 2
(.c /.c++ etc.)
(Module-2)
Module
Cross-compiler
Object File 2
Object to Hex File
Converter
Absolute Object File
Linker/
Locator
Machine Code
(Hex File)

Advantages

- Reduced Development Time
- Developer Independency
- Portability

Drawbacks

- The time required to execute a task also increases with the number of instructions
- hardware access timing is critical
- The investment required for high level language based development is high
- Some compilers available for high level languages may not be so efficient in generating optimised target processor specific instructions.

Mixing Assembly and High Level Language

- mixing assembly language with high level language
- mixing higher level language with assembly
- in-line assembly programming.

Mixing Assembly with High level language (e.g. Assembly Language with 'C')

- Assembly routines are mixed with ‘C’ in situations where the entire program is written in ‘C’ and the cross compiler in use do not have a built in support for implementing certain features like **Interrupt Service Routine functions** (ISR) or if the programmer wants to take advantage of the **speed** and **optimised code** offered by machine code generated by hand written assembly rather than cross compiler generated machine code.
- When accessing certain low level hardware, the timing specifications may be very critical and a cross compiler generated binary may not be able to offer the required time specifications accurately.
- Passing parameter to the assembly routine and returning values from the assembly routine to the caller ‘C’ function and the method of invoking the assembly routine from ‘C’ code is cross compiler dependent.

Mixing High level language with Assembly (e.g. ‘C’ with Assembly Language)

- The source code is already available in Assembly language and a routine written in a high level language like ‘C’ needs to be included to the existing code.
- The entire source code is planned in Assembly code for various reasons like optimised code, optimal performance, efficient code memory utilisation and proven expertise in handling the Assembly, etc.
- But some portions of the code may be very difficult and tedious to code in Assembly.
- To include **built in library functions** written in ‘C’ language provided by the cross compiler. For example, **Built in Graphics library functions** and **String operations** Supported by ‘C’.

lnline Assembly

- lnline assembly is another technique for inserting target processor/controller specific Assembly instructions at any location of a source code written in high level language ‘C’.

- Compiler vs cross compiler
- Compiler vs assembler
- C vs Embedded C
- High level language vs low level language