UNIT-I

Introduction to Embedded Systems

Embedded System Introduction:

We live in an era where pervasive (spreading) computing exists everywhere, right from a small handheld device such as a mobile phone to the electronic control units within automobiles or avionics. Today, large volumes of information is getting processed and communicated over the Internet every microsecond. Buzz words such as Cloud Computing, Big Data Mining and Internet of Things are everywhere.

There are two broad classifications of computing systems - general purpose computing system and embedded computing systems. If we define these in simple words, general purpose computing systems are those used in desktop or laptop computers, which can process several different applications. An embedded system refers to any device that has some computational intelligence in it. It is generally used as a standalone system that repeatedly performs a specific task or as part of a large system to perform multiple tasks with the required hardware and software embedded within. Systems used in printers, washing machines, mp3 players; CT scan machines etc. are great examples of embedded systems.

An embedded system is a constrained system and its design goals vary from a general purpose system. The constraints are: high performance, low power consumption, small size and low cost of the system.

The basic components of an embedded system include hardware, software and some mechanical parts. Embedded hardware includes a processing unit, block of memory and I/O sub-unit which are called as the system resources. The embedded software can be thought of as the application software in a small computing system or both the system and the application software in case of a large complex system. The system software mentioned here is the real time operating system (RTOS) used to manage the usage of system resources by application software.

What is an Embedded System?

- An embedded system is a system that has software embedded into computer-hardware, which makes a system dedicated for an application or specific part of an application or product or part of a larger system.
- An embedded system is designed to do a specific/few task only.

- **Examples:**
    - A Washing machine can only wash clothes.
    - A Digital camera can only capture the images.
    - An Air conditioner can control the temperature in the room.

Host and Target concept:

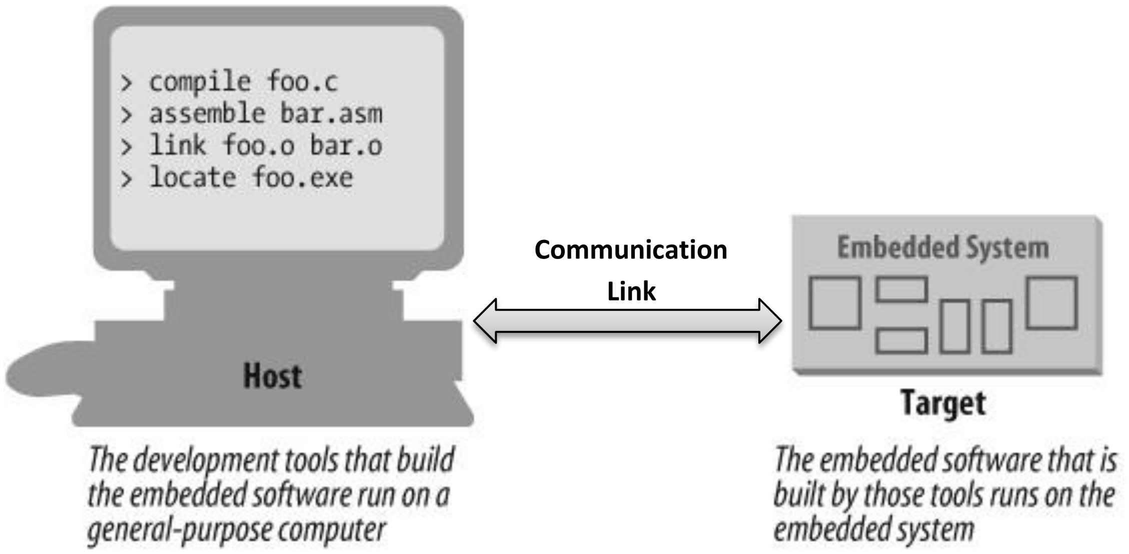

> compile foo.c
> assemble bar.asm
> link foo.o bar.o
> locate foo.exe
Host
Communication Link
Embedded System
Target
The development tools that build the embedded software run on a general-purpose computer
The embedded software that is built by those tools runs on the embedded system

Figure: Embedded System using Host and Target Machine

Performance of Host machine:

The application program developed runs on the host computer. The host computer is also called as Development Platform. It is a general purpose computer. It has a higher capability processor and more memory. It has different input and output devices. The compiler, assembler, linker, and locator run on a host computer rather than on the embedded system itself. These tools are extremely popular with embedded software developers because they are freely available (even the source code is free) and support many of the most popular embedded processors. It contains many development tools to create the output binary image. Once a program has been written, compiled, assembled and linked, it is moved to the target platform.

Performance of Target machine:

The output binary image is executed on the target hardware platform. It consists of two entities - the target hardware (processor) and runtime environment (OS). It is needed only for final output. It is different from the development platform and it does not contain any development tools.

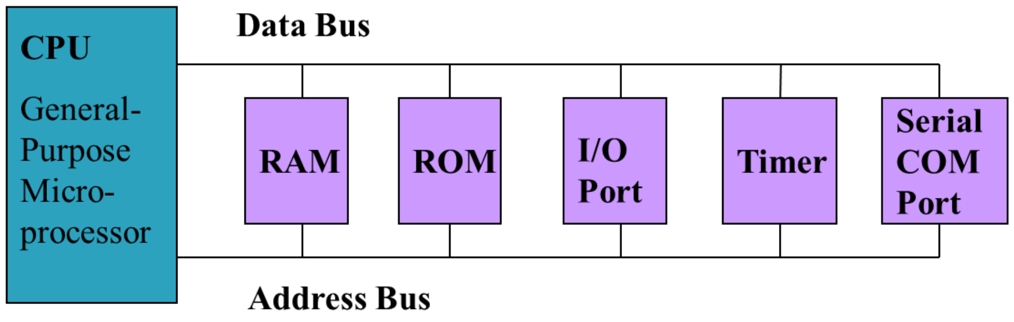
Embedded Applications:

Where Embedded Systems?

- Avionics: Navigation systems, Inertial Guidance Systems, GPS Receivers etc
- Automotives: Automatic Breaking System, Air Bag System, Ignition Systems etc.

- Consumer Electronics: MP3 Players, Mobile Phones, Video Game Consoles, Digital Cameras, Set top Boxes, Camcorders etc.
- Telecommunication: Cell Phones, Telephone Switches etc.
- Medical Systems: ECG, EEG, MRI, CT scan, BP Monitors, etc.
- Military Applications
- Industries
- Home Appliances: AC, TV, DVD Players, Washing machine, Refrigerators, Microwave Oven, etc.
- Computer Peripherals: Printers, Scanners, Fax Machines.
- Computer Networking Systems: Network Routers, Switches, Hubs etc.
- Security Systems: Intruder Detection Alarms, CC Cameras, Fire Alarms
- Measurement & Instrumentation: Digital Multimeters, Digital CRO's, Logic Analyzers, PLC Systems.
- Banking and Retail: ATM, Currency Counters.
- Card readers: Barcode, Smart Card Readers, Hand Held Devices.
- And Many More….

Features and Architecture Considerations for Embedded Systems:



CPU/Processor: It act as a brain to the system, it is used to execute the program and to sending and receiving signals. The processing unit could be a microprocessor, a microcontroller, embedded processor, DSP, ASIC or FPGA selected for an embedded system based on the application requirements.

ROM for the program: Nonvolatile (read-only memory, ROM); meaning that it retains its contents even when power is off. It also called Program memory. In embedded systems, the application program after being compiled is saved in the ROM. The processing unit accesses the ROM to fetch instructions sequentially and executes them within the CPU. There are different categories of ROM such as: programmable read only memory (PROM), erasable programmable read only memory (EPROM), electrically erasable programmable read only memory (EEPROM) etc. There is also flash memory which is the updated version of EEPROM and extensively used in embedded systems.

RAM for data: Random access memory (RAM) is volatile i.e. it does not retain the contents after the power goes off. It is used as the data memory in an embedded system. It holds the variables declared in the program, the stack and intermediate data or results during program run time. The Processing unit accesses the RAM for instruction execution to save or retrieve data. There are different variations of RAM such as: static RAM (SRAM), dynamic RAM (DRAM), pseudo static RAM (PSRAM), non-volatile RAM (NVRAM), synchronous DRAM, (SDRAM) etc.

I/O PORTS: To provide digital communication with the outside world. These Ports are two types Parallel Port and Serial Port, Parallel are used to communicate with Parallel I/O devices like LEDs, LCDs, 7-Segment Displays, Keypads etc., Serial Port is used to communicate with Serial devices like Bluetooth module, Wi-Fi, Zigbee, GSM Modules, PCs etc.

Address and Data buses: To link these subsystems to transfer data and instructions. The system bus consists of three different bus systems: address bus, data bus and control bus. Processor sends the address of the destination through the address bus. So address bus is unidirectional from processor to the external end. Data can be sent or received from any unit to any other unit in the diagram. So data bus is bidirectional. Control bus is basically a group of control signals from the processing unit to the external units

Timers: Timers are a fundamental concept in embedded systems and they have many use cases such as creating accurate delays, executing a periodic task, implementing a PWM (Pulse With Modulation) output or capturing the elapsed time between two events, to vary the speed of data transfer rate i.e. Baud rate (bps-bits per second) in case serial communication etc.

Clock: To keep the whole system synchronized. It may be generated internally or obtained from a crystal or external source; modern MCUs offer considerable choice of clocks.

Watchdog timer: This is a safety feature, which resets the processor if the program becomes stuck in an infinite loop/hang.

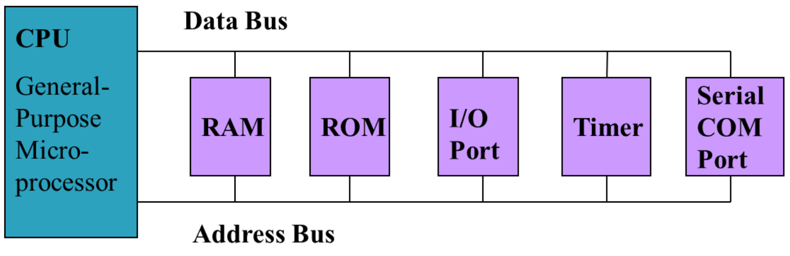
Communication interfaces: A wide choice of interfaces is available to exchange information with another IC or System. They include Universal Asynchronous Receiver/Transmitter (UART), Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I^2C or IIC), Universal Serial Bus (USB), Controller Area Network (CAN), Ethernet, and many others.

Non-volatile Memory for data: This is used to store data whose value must be retained even when power is removed. Serial numbers for identification and network addresses are two obvious candidates.

Microprocessor vs. Microcontroller

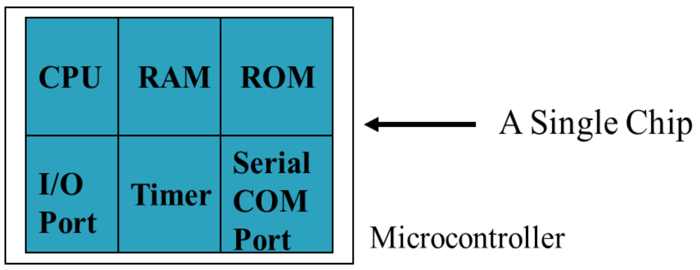
Microprocessor Based Embedded System:

- CPU for Computers
- No RAM, ROM, I/O on CPU chip itself
- Example: Intel's 8085,8086, Motorola's 680xx



Microcontroller Based Embedded System:

- A Small Computer or System on Chip (SoC)
- On-chip RAM, ROM, I/O ports...
- Example: TI's MSP430, Motorola's 6811, Intel's 8051, Zilog's Z8 and PIC 16X



S.No.	**Microprocessor**	**Microcontroller**
1	Contains Only CPU (**µP**); RAM, ROM, I/O Ports, Serial Port, Timers are Separately Interfaced	CPU (**µP**), RAM, ROM, I/O Ports, Serial Port, Timers are all on a Single Chip.
2	It has Many Instructions to Move data between Memory and **µP**	It has One or Two Instructions to Move data between Memory and **µC**
3	It has Few Bit Manipulation Instructions	It has Many Bit Manipulation Instructions
4	It has Less Number of Multifunctional Pins	It has More Number of Multifunctional Pins
5	General Purpose	Single (Specific) Purpose
6	High Speed	Low Speed
7	High Power Consumption	Low Power Consumption
8	**µP** Based System Requires More Hardware & High Cost	**µC** Based System Requires Less Hardware & Less Cost
9	Designer can decide on the amount of ROM, RAM and I/O ports.	Fixed amount of on-chip ROM, RAM, I/O ports

Embedded Processor and their types:

Microprocessor

Microprocessor is a programmable digital device which has high computational capability to run a number of applications in general purpose systems. It does not have memory or I/O ports built within its architecture. So, these devices need to be added externally to make a system functional. In embedded systems, the design is constrained with limited memory and I/O features. So microprocessors are used where system capability needs to be expanded by adding external memory and I/O.

Microcontroller

A microcontroller has a specific amount of program and data memory, as well as I/O ports built within the architecture along with the CPU core, making it a complete system. As a result, most embedded systems are microcontroller based, where are used to run one or limited number of applications.

Embedded Processor

Embedded processors are specifically designed for embedded systems to meet design constraints. They have the potential to handle multitasking applications. The performance and power efficiency requirements of embedded systems are satisfied by the use of embedded processors.

DSP

Digital signal processors (DSP) are used for signal processing applications such as voice or video compression, data acquisition, image processing or noise and echo cancellation.

ASIC

Application specific integrated circuit (ASIC) is basically a proprietary device designed and used by a company for a specific line of products (for example Samsung cell phones or Cisco routers etc.). It is specifically an algorithm called intellectual property core implemented on a chip.

FPGA

Field programmable gate arrays (FPGA) have programmable macro cells and their interconnects are configured based on the design. They are used in embedded systems when it is required to enhance

Memory Types:

The semiconductor memory can be classified into two types

- Volatile Memory
- Non-volatile memory

Volatile memory

- RAM
  - Static RAM
  - Dynamic RAM

Non-volatile memory

- ROM
  - Masked ROM
  - OTPROM
  - EPROM
  - EEPROM
  - FLASH
    - NOR FLASH
    - NAND FLASH
- NVRAM (Battery Backup RAM)

Volatile Memory:

RAM (Random Access Memory):

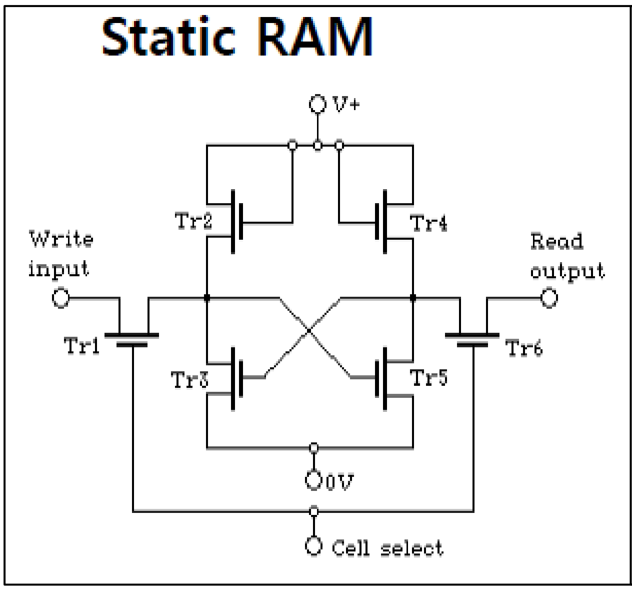
Loses its contents when power is removed, it is usually called Random-Access Memory-RAM. The vital feature is that data can be read or written with equal ease. It has ability to access any memory cell directly. RAM is much faster than ROM. It used to write and read data values while program running. Local variables, pointers, functions, recursive functions results in using large amounts of RAM. Volatile memory is used for data, and small microcontrollers often have very little RAM.

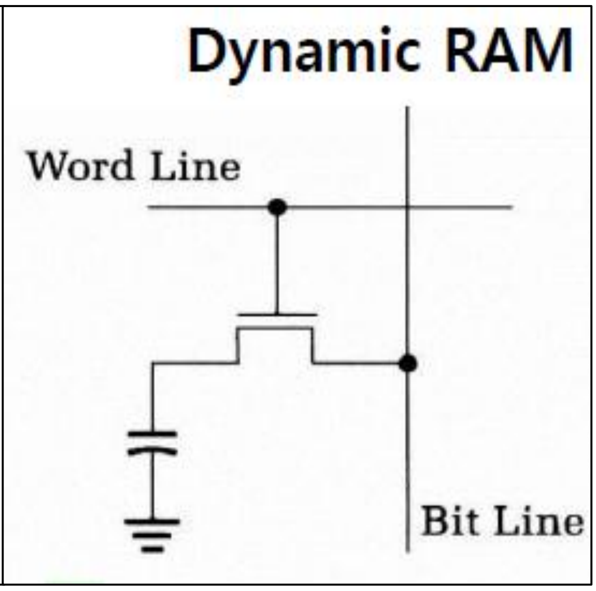
Static RAM:

Means that it retains its data even if the clock is stopped (provided that power is maintained, of course). A single cell of static RAM needs six transistors. RAM therefore takes up a large area of silicon, which makes it expensive.

Dynamic RAM:

This needs only one transistor per cell but must be refreshed regularly to maintain its contents, so it is not used in small microcontrollers. Most memory in a desktop computer is dynamic RAM.





Static RAM Vs Dynamic RAM:

Static RAM (SRAM)	Dynamic RAM (DRAM)
Made from Flip-Flops.	Made from Capacitors
High cost (per bit)	Low cost (per bit)
High using Power	Low using Power
Fast	Slow
Used in Cache Memory	Used in Main Memory
Large in Size	Low in Size
Will Retain State Forever	Automatically Discharges after sometime, Need Refreshing

Nonvolatile Memory:

ROM (Read-Only Memory):

Retains its contents when power is removed and is therefore used for the program and constant data. It is usually called Read-Only Memory-ROM. It is used as Program Memory in Microcontroller. It can't be written or modified at run time. There are many types of nonvolatile memory in use:

Masked ROM:

The data are encoded into one of the masks used for photolithography and written into the IC during manufacture. This memory really is read-only. It is used for the high-volume production of stable products, because any change to the data requires a new mask to be produced at great expense.

OTP ROM (One-Time Programmable ROM):

This is just PROM (Programmable ROM) in a normal package without a window, which means that it cannot be erased. It can be programmed one time only. Used when the firmware is stable and the product is shipping in bulk to customers. Devices with OTP ROM are still widely used and the first family of the MSP430 used this technology.

EPROM (Erasable Programmable ROM):

As its name implies, it can be programmed electrically but not erased. Devices must be exposed to ultraviolet (UV) light for about ten to twenty minutes to erase them. Erasable devices need special packages with quartz windows, which are expensive.

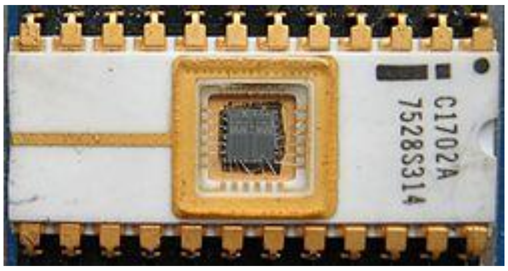


Figure: Example of EPROM IC with quartz window

EEPROM:

EEPROM (also E^{2}PROM) stands for **Electrically Erasable Programmable ROM**. EEPROMs can be programmed and erased in-circuit i.e. without removing from hardware kit, by applying electrical signals. The contents of this memory may be changed during run time (similar to RAM), but remains permanently saved even if the power supply is off (similar to ROM). EEPROM is often used to read and store values, created during operation, which must be permanently saved.

FLASH Memory:

This can be both programmed and erased electrically and is now by far the most common type of memory. Flash Memory is designed for high speed and high density, at the expense of large erase blocks (typically 512 bytes or larger). The practical difference is that individual bytes of EEPROM can be erased but flash can be erased only in blocks. Most MSP430 devices use flash memory, shown by an **F** in the part number. Microcontrollers use NOR flash, which is slower to write but permits random access. NAND flash is used in bulk storage devices and can be accessed only serially in rows.

Many microcontrollers include both: Flash Memory for the firmware (embedded program), and a small EEPROM for parameters and history.

Overview of Design Process of Embedded Systems:

The below figure shows the Design Process of Embedded Systems, In the first stage, according to user requirements the designer chooses the electronic chip which is suitable. In the next stage the designer will work on S/W side and H/W side separately. After that they will integrate the both S/W and H/W to design the Embedded System.

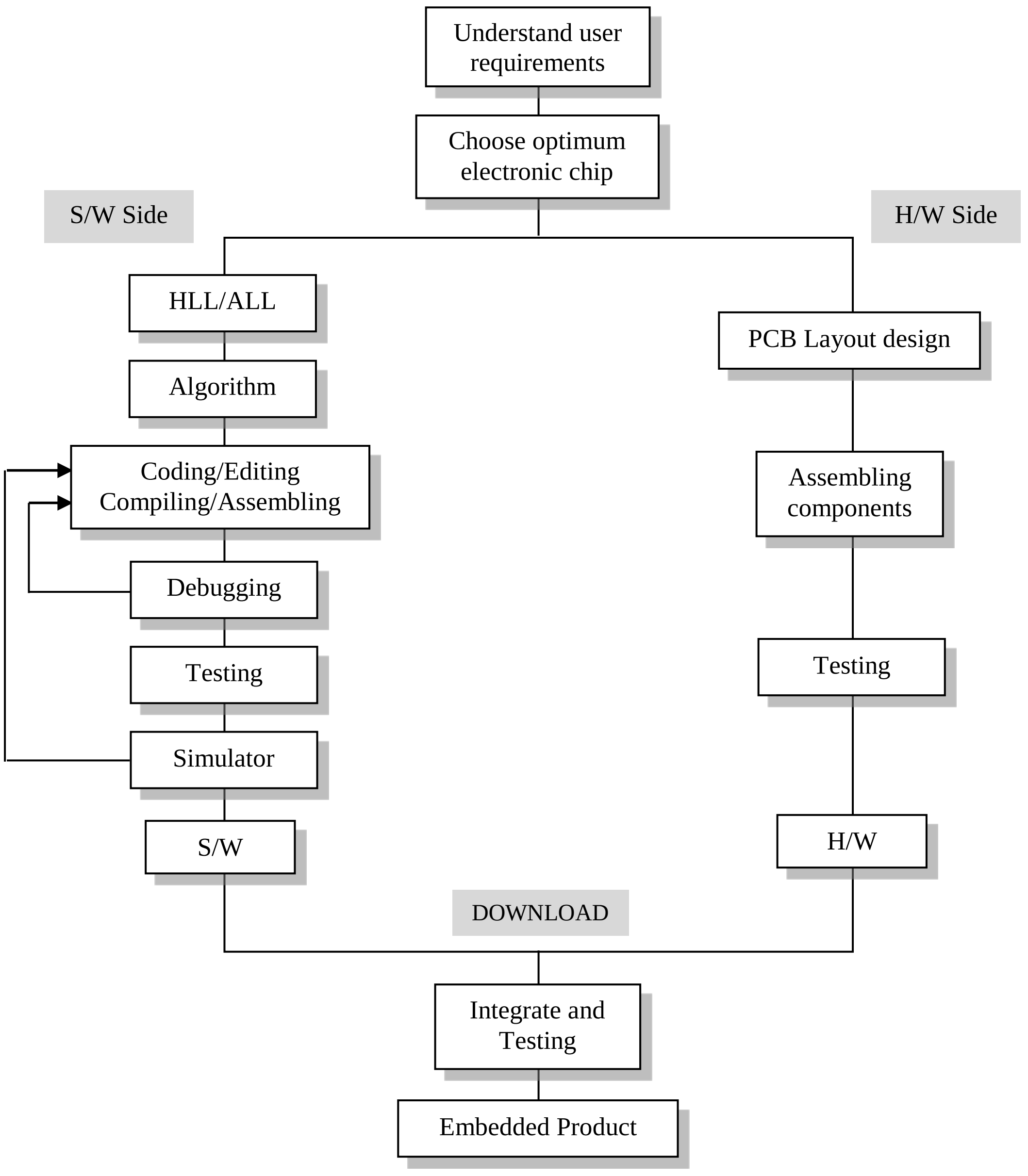


Figure: Design Process of Embedded Systems

An example of an embedded system:

Here is a simple application to introduce a small embedded system – a stepper motor controller for a robotics system. The stepper motor mentioned here is an electromechanical device that rotates in discrete step angles when electrical pulses are applied to it. Suppose in an industrial environment, a robot arm is employed to pick-up components from one container and deposit to another container. The robot arm is operated by three stepper motors, one to move the arm from one container to the other and other two to make a grip to tightly hold the component. To control these stepper motors, a small embedded system can be designed as shown in below figure. The hardware components are a microcontroller, three stepper motors and a robot arm. To make this system functional, three program modules are required.

- ⇨ Module1 for the stepper motor1 to rotate counter clockwise with definite angle so that robot arm can move from container1 to container 2 also to rotate clockwise to move back to container1 when the component is picked.
- ⇨ Module 2 for the stepper motors 2 & 3 to make the motor 2 to rotate in counter clockwise and motor 3 to rotate in clockwise both simultaneously so that the robot arm can pick and make a grip on the component.

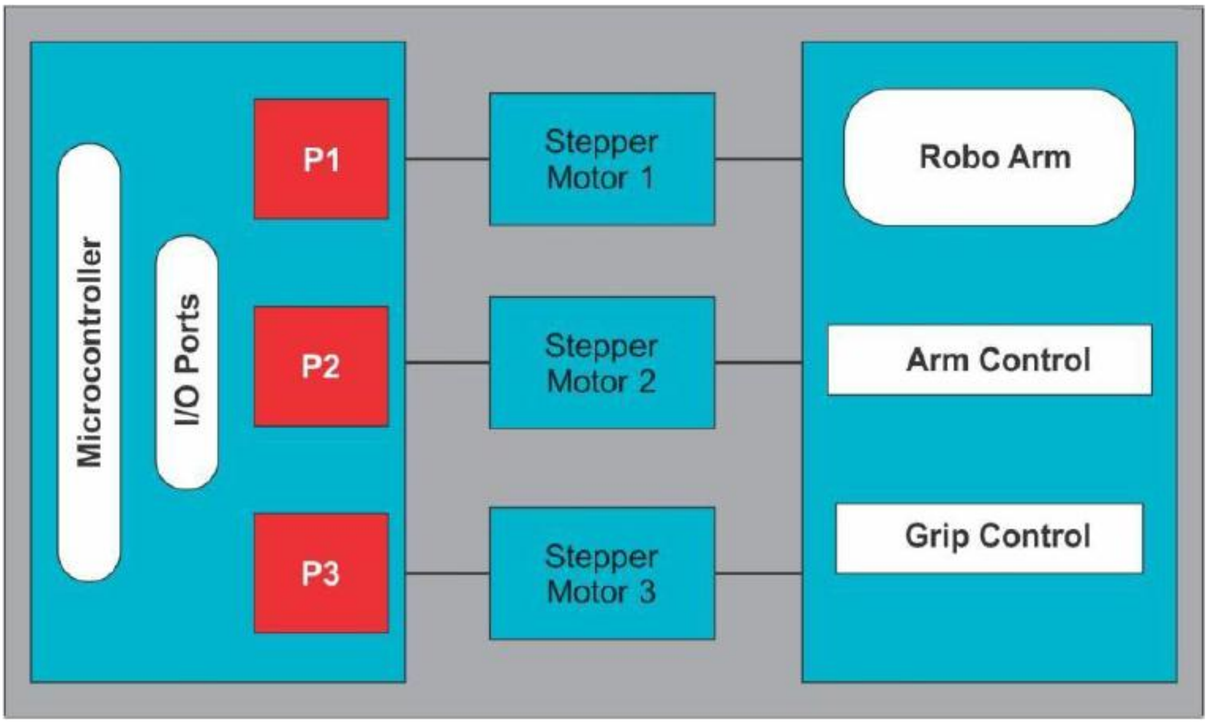


Figure: Stepper Motor Control Embedded System

Programming Languages and Tools for Embedded Design:

Programming Languages:

- ⇨ Code is typically written in C or C++, but various high-level programming languages, such as Python and JavaScript, are now also in common use to target microcontrollers and embedded systems. Ada is used in some military and aviation projects.

Machine code: The binary data that the processor itself understands. Each instruction has a binary value called an opcode. It is unrecognizable to humans, unless you spent a very long time on low-level debugging. Some very early computers had to be programmed in machine code, but that was long ago, thank goodness. You will see it, however, because the contents of memory are shown in the debugger and machine code is included in the “disassembly”.

Assembly language: Little more than machine code translated into English. The instructions are written as words called mnemonics rather than binary values and a program called an assembler translates the mnemonics into machine code. It does a little more than direct translation, but not a lot, nothing like a compiler for a high-level language.

A major disadvantage of assembly language is that it is intimately tied to a processor and is therefore different for each architecture. Even worse, the detailed usage varies between development environments for the same processor. Most programming of small microcontrollers was done in assembly language until recently, despite these problems, mainly because compilers for C produced less-efficient code. Now the compilers are better and modern processors are designed with compilers in mind, so assembly language has been pushed to the fringes. A few operations, notably bitwise rotations, cannot be written directly in C, and for these assembly language may be much more efficient. However, the main argument for learning assembly language is for debugging. There is no escape if you need to check the operation of the processor, one instruction at a time. Disassembly is the opposite process to assembly, the translation of machine code to assembly language.

C: The most common choice for small microcontrollers nowadays. A compiler translates C into machine code that the CPU can process. This brings all the power of a high-level language—data structures, functions, type checking and so on—but C can usually be compiled into efficient code. Compilation used to go through assembly language but this is now less common and the compiler produces machine code directly. A disassembler must then be used if you wish to review the assembly language.

C++: An object-oriented language that is widely used for larger devices. A restricted set can be used for small microcontrollers but some features of C++ are notorious for producing highly inefficient code. Embedded C++ is a subset of the language intended for embedded systems. Java is another object-oriented language, but it is interpreted rather than compiled and needs a much more powerful processor.

BASIC: Available for a few processors, of which the Parallax Stamp is a well-known example. The usual BASIC language is extended with special instructions to drive the peripherals. This enables programs to be developed very rapidly, without detailed understanding of the peripherals. Disadvantages are that the code often runs very slowly and the hardware is expensive if it includes an interpreter.

Programming Tools:

The microcontrollers can be programmed using various Programming/Development tools like Embedded Software from Mentor Graphics, IAR Systems, Keil MicroVision from ARM Ltd., and Code Composer Studio (CCS) from Texas Instruments.

Introduction to Code Composer Studio:

At this juncture of the book, it is important to concentrate on a single development tool for programming the hardware. In this book, Code Composer Studio is used because it is free and is supported by a large TI community commonly known as E2E. This forum is a congregation of engineers and hobbyist working on TI products across the world.

Code Composer Studio (CCS):

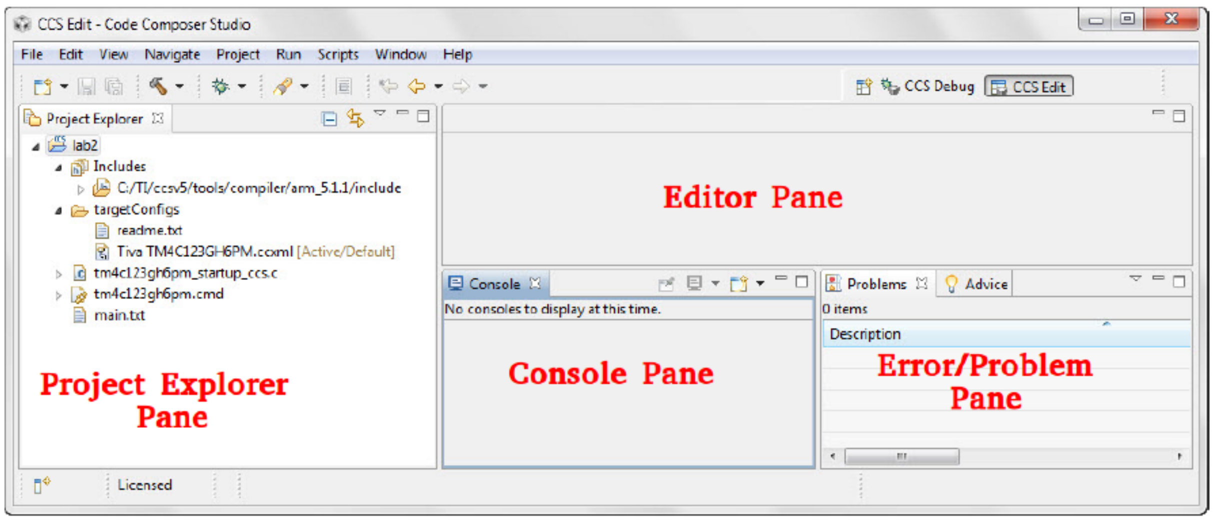
CCS is a TI proprietary cross platform IDE used to convert C and assembly language code to executable for TI processors, involving Digital Signal Processors, ARM, and other microcontrollers. CCS combines editing, debugging and analysis tools into a single IDE based on Eclipse open source development tool.

Compilation (Build Process):

CCS uses a Code Generation Tool (often called CGT) for compilation process also called as build process. It is used to generate executables for a target device. This process takes source codes written in C, C++ and/or Assembly Language and produces executable at the output called binaries. This process undergoes many intermediated stages. A normal development flow of a compilation process takes the source files written in C/C++ and compiles them to create assembly language files using a compiler. For projects where source code is written in assembly language, this process is bypassed. In many projects, some of the logic functions are written in assembly languages along with other C/C++ source codes. After the assembly language files are generated by the compiler, an assembler converts them to re-locatable object files. These object files are linked to the runtime libraries included in the project by a linker. The linker produces executables from these files using protocols mentioned in the command file. A command file typically consists of a list of all memories of the part and the type of software compatible to them.

CCS uses C/C++ compiler and other compilation tools which are developed by engineers at TI over the years and packaged into TI‟s code generation tools. They are commonly amalgamation (combination) of various tools used in compilation processes, namely, Compiler, Assembler, Linker, Optimizer, Code Generator, Parser, Linear Assembler, Archiver, Disassembler and many more.

Review the CCS Editing GUI – Figure below shows various panes of CCS which appear after project is created. Understanding their nomenclature and functionality with help in process of application development


CCS Edit - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
CCS Debug
CCS Edit
Project Explorer
lab2
Includes
C:/TI/ccsv5/tools/compiler/arm_5.1.1/include
targetConfigs
readme.txt
Tiva TM4C123GH6PM.ccxml [Active/Default]
tm4c123gh6pm_startup_ccs.c
tm4c123gh6pm.cmd
main.txt
Project Explorer Pane
Editor Pane
Console
No consoles to display at this time.
Console Pane
Problems
Advice
0 items
Description
Error/Problem Pane
Licensed

Debugger:

The CCS debugger depends on a configuration file and a general extension language (GEL) file. The debugger initializes and loads the software on a target device using information provided by these files. A target configuration file specifies

(i) Connection type to the target device,

(ii) Target device, and

(iii) About a startup script

Table: Programming/Development Tools for Tiva C Series

Product	License	Compiler	IDE	Debugger	JTAG
Embedded Software, Mentor Graphics	30-day full function	GNU C/C++	Gdb	Eclipse	
KEIL MicroVision, ARM Ltd.	Full function. Onboard emulation limited	Real View C/C++	µVision	µVision	U-Link, 199 USD
CCS, Texas Instruments	32KB code size limited. Upgradeable	TI C/C++	Eclipse	CCS	XDS100 79 USD
IAR Systems	32KB code size limited. Upgradeable	IAR C/C++	Embedded Workbench	C-SPY	J-Link, 299 USD

In CCS, startup scripts are specified to setup the memory map for debugger. It is also used to setup any initial target state that is necessary for connection to the debugger using memory or register writes. These scripts are known as GEL script file. „OnStartup()‟ function in the GEL file runs when the debugger is launched. After the target is connected, „OnTargetConnect()‟defined in the GEL file is executed.

Organization and Building a CCS Project:

In CCS, designs are organized in workspaces and projects which are merely folders in the file system. When CCS is launched, it prompts the users to provide a folder path to the workspace. This folder consists of individual project folders and a folder named as '.metadata'. The .metadata folder consists of CCS settings and preferences for the particular workspace. Along with source files, header files and library files, each project folder contains the build and the tool settings for the project. It also contains the target configuration file and the command file required by the debugger. CCS has two predefined build configurations, namely, debug and release. It also provides custom build configurations.

Debug – It is generally used when it is required to operate in debugging mode. It includes the symbol tables and executes compilation without any optimization.

Release – Building project in this mode is suited when the user requires performance. It discards all symbol tables and implements the full code optimization. It is therefore a noted convention to use this mode only when the final version of a project is to be deployed on the hardware. For all other intermediate versions, debug mode is rather preferred.

Custom Configuration – CCS also provide its users to add custom build configurations for a particular project. It can be done by going to processor options under 'properties -> build -> ARM Compilers‟.