Building Embedded Hardware and Software

Instructor: Dr. Ankur Beohar
School of Electrical and Electronics Engineering
VIT Bhopal University

Embedded Systems

- Suggested Textbooks:
  - Raj Kamal, “Embedded systems Architecture, Programming and Design”, Third Edition, Tata McGraw Hill, 2017.
  - Rob Toulson and Tim Wilmshurst, “Fast and Effective Embedded Systems Design – Applying the ARM mbed”, Elsevier, 2017.
  - Arnold S. Berger, “Embedded Systems Design: An Introduction to Processes, Tools, and Techniques”, CRC Press, 2002.
- Other sources
  - Lecture notes
  - Handouts
  - Blogs
  - MOOC courses

Syllabus Overview

- Hardware: ADC, DAC, Memory devices, Sensors and Actuators, PCB design process
- Software: Cross assemblers/compilers, Linker, Runtime Library, Preprocessor Workflow, make files, Compiler Tool chains – gcc & ARM, Device Driver, Firmware, Middleware
- Debugging tools: Emulators, Simulators, In-Circuit Debuggers, Logic Analyser, Integrated Development Environment (IDE)

Activity 1

- Research two microcontrollers and provide information about them from their datasheets. There are several microcontroller manufacturers which you can investigate including Atmel, Microchip, Freescale, TI, etc. For each microcontroller, report the following information.
  - Clock frequency
  - Bitwidth of the datapath
  - Size of Flash memory
  - Number of pins
  - Does the microcontroller contain an Analog-to-Digital Converter? If so, how many bits of precision does it have?
- Research the Arduino and Raspberry Pi platforms to determine if there are operating systems which can be used on each platform. If there are, list those operating systems and state whether they are open source or not.

Activity 2

- Identify two embedded systems which are sold on the market today and analyze their interfaces.
- Describe all inputs to each system and outputs from each system.
- Classify the inputs and outputs based on their mode of interaction:
    - Visual - describing data carried by visible light
    - Audio - describing data carried by sound
    - Tactile - describing data carried by touch
    - Electronic - describing data encoded in electrical signals

Sensors

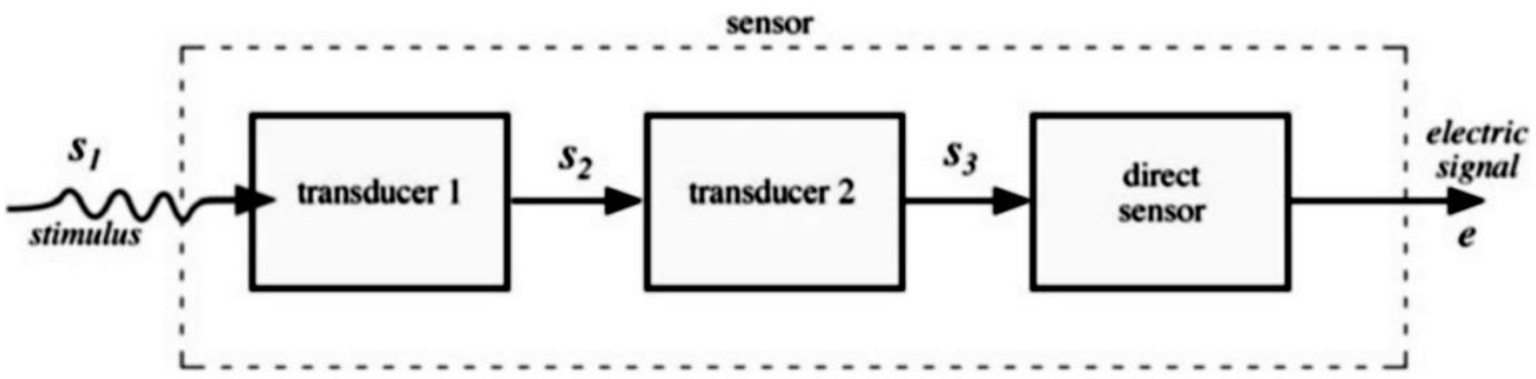
- A **sensor** is a device that receives a stimulus and responds with an electrical signal
- The terms **sensor** and **detector** are synonyms
- Different sensors are IR sensors, proximity sensors, temperature sensors, tilt sensors, accelerometers, ultrasonic sensors, RADAR, SONAR, etc.

Sensors

- The **stimulus** is the quantity, property, or condition that is received and converted into electrical signal.
- Examples of stimuli are light intensity and wavelength, sound, force, acceleration, distance, rate of motion, and chemical composition.
- When we say "**electrical**" we mean a signal which can be channeled, amplified, and modified by electronic devices.
- The term **sensor** should be distinguished from **transducer**.
- The **transducer** is a converter of any one type of energy or property into another type of energy or property, whereas the **sensor** converts it into electrical signal.

Sensors

- Transducers may be parts of a hybrid or complex sensor
- There are two types of sensors, direct and hybrid.
- A direct sensor converts a stimulus into an electrical signal or modifies an externally supplied electrical signal, whereas a hybrid sensor in addition needs one or more transducers before a direct sensor can be employed to generate an electrical output.



Sensor Characteristics

- Range
- Span
- Accuracy
- Precision
- Sensitivity
- Linearity
- Hysteresis
- Resolution
- Repeatability

Sensor Types

- Temperature Sensors
- Pressure Sensors
- Flow Sensors
- Level Sensors
- Imaging sensors
- Noise Sensors
- Air Pollution Sensors
- Proximity and displacement Sensors
- Infrared Sensors
- Moisture and Humidity Sensors
- Speed Sensors

Actuators

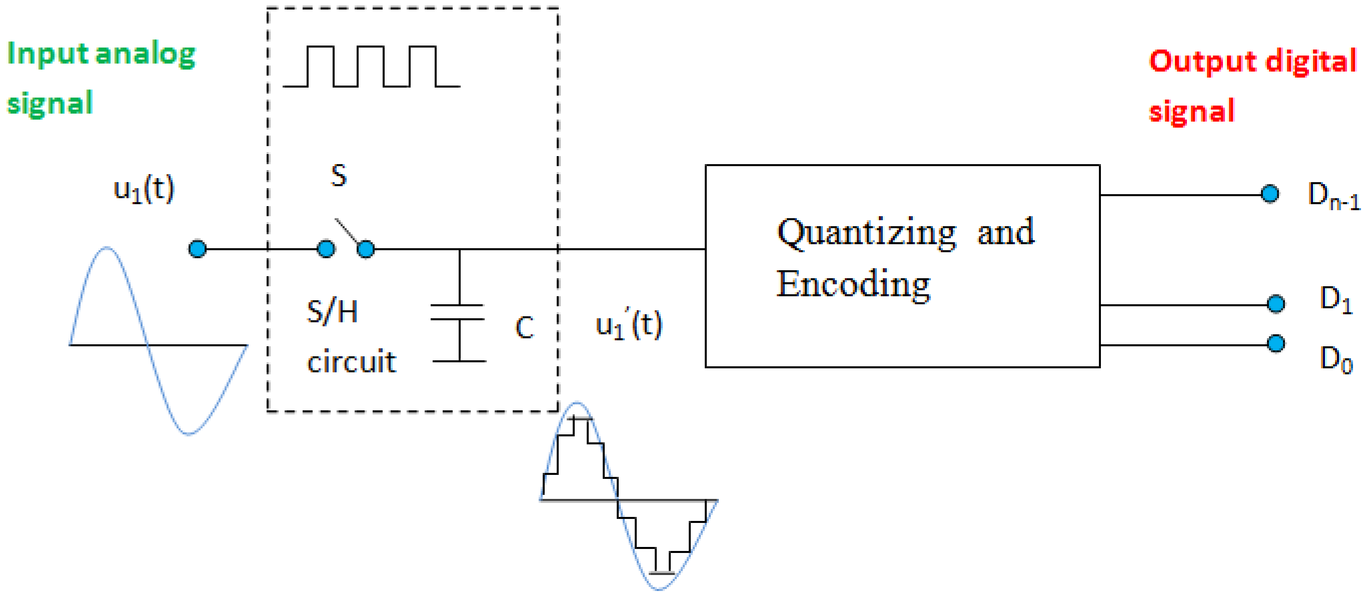
- The actuator is actually a device that transforms a certain form of energy into motion.

- As their mechanism converts energy into motion, we can categorize them based on energy sources:

  - **Pneumatic**: use compressed air for generating motion.
  - **Hydraulic**: use the liquid for generating motion.
  - **Thermal**: use a heat source for generating motion.
  - **Electric**: use external energy sources such as batteries or other types of electric energy to generate motion.

Analog to Digital Conversion

- 3 steps involved
  - Sampling
  - Quantization
  - Encoding



Analog to Digital Conversion

Sampling and Holding

- In the process of Sample and hold (S/H), the continuous signal will gets sampled and freeze (hold) the value at a steady level for a particular least period of time.

- It is done to remove variations in input signal which can alter the conversion process and thereby increases the accuracy.

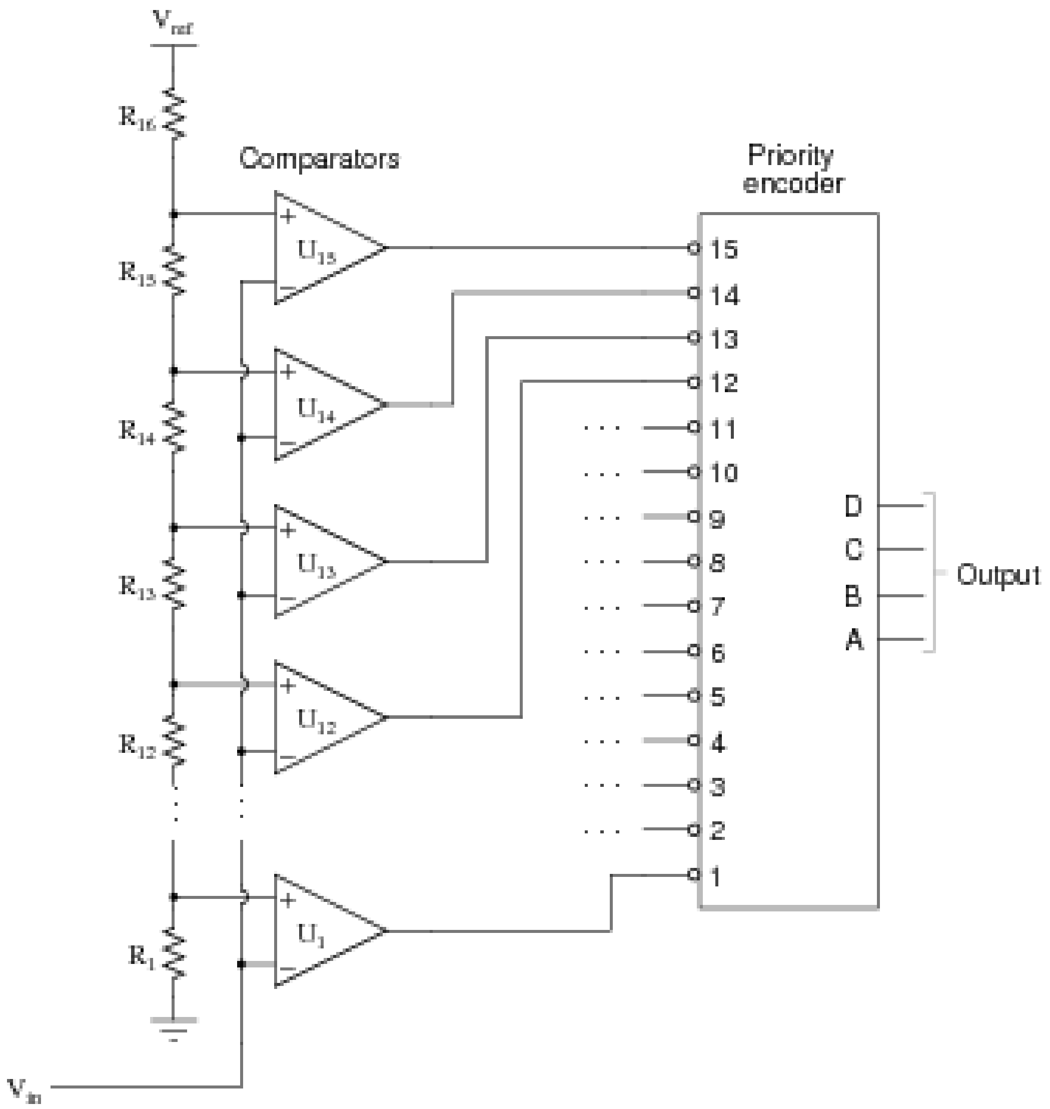
- The minimum sampling rate has to be two times the maximum data frequency of the input signal

Analog to Digital Conversion

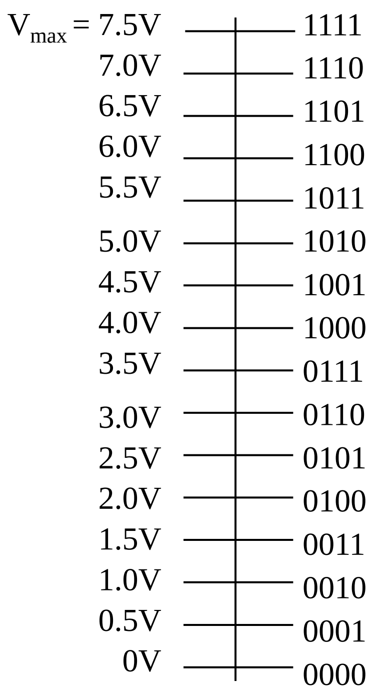
Quantizing and Encoding

- Quantizing: It is the process in which the reference signal is partitioned into several discrete quanta and then the input signal is matched with the correct quantum.
- Encoding: A unique digital code is assigned for each quantum and after that the input signal is allocated with this digital code.

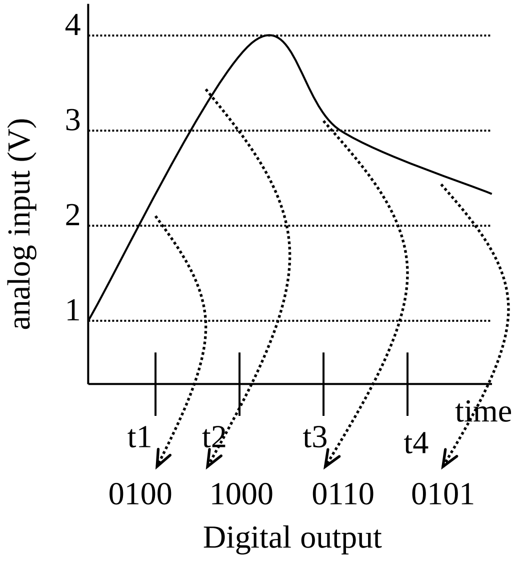
Analog to Digital Conversion



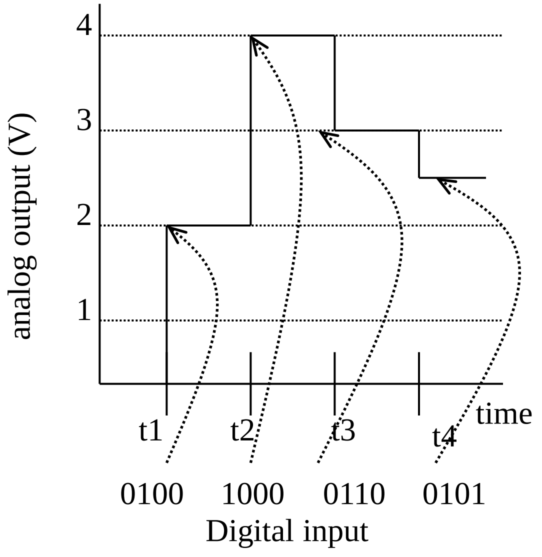
Analog-to-Digital Converter



proportionality



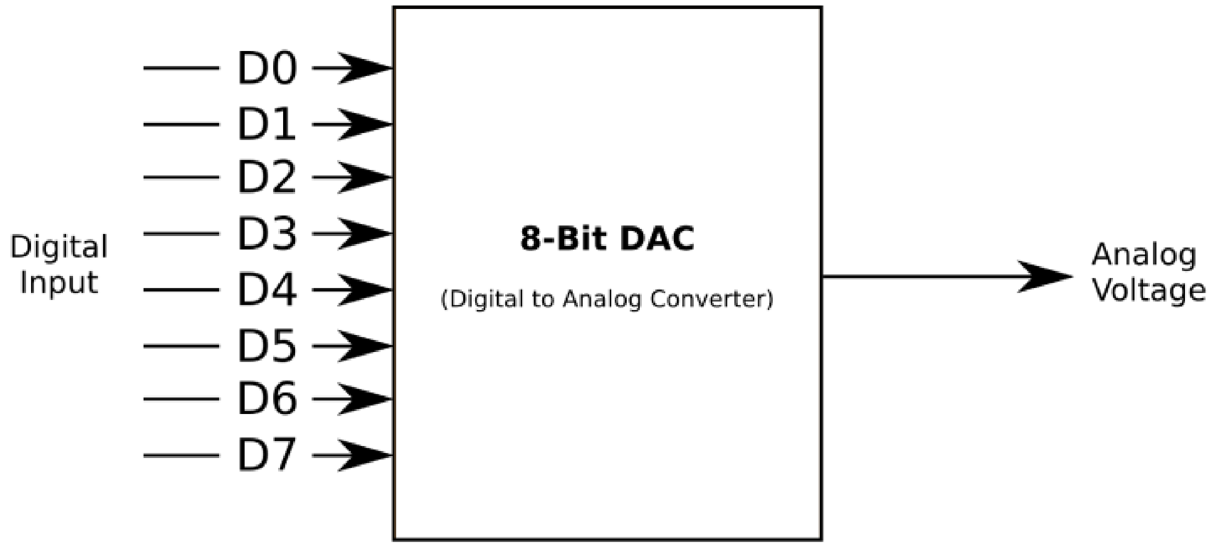
Analog to Digital
(A/D)



Digital to Analog
(D/A)

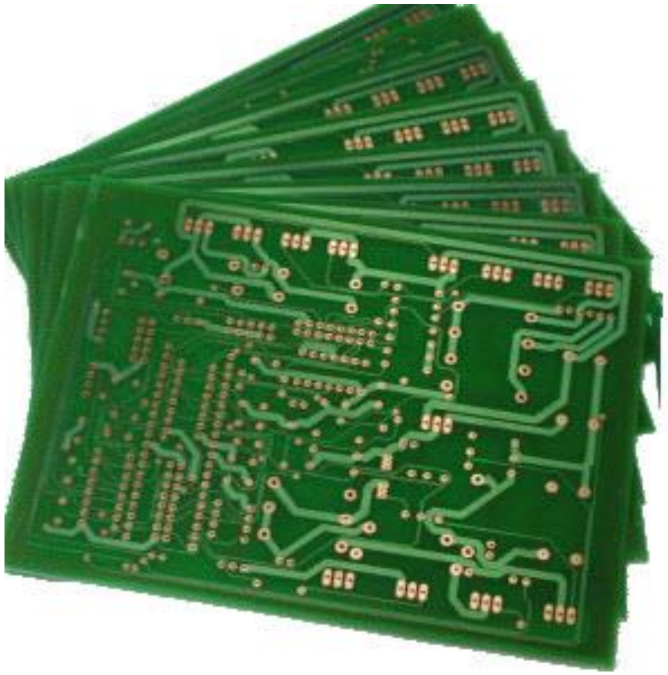
Digital to Analog Converter

- Digital input to analog output
- Mainly used in the output stage
- DAC can reconstruct sampled data into an analog signal with precision
- 2 types
  - Binary Weighted resistor and R-2R ladder



PCB: Printed Circuit Board

- These boards are made up of special materials that do not conduct electricity such as fiber and glass.
- The circuits are designed on the boards with copper tracks instead of wires for the conduction of electricity between the electronic components.
- The printed circuit boards used in all electronic products such as automotives, wireless devices, Robotic applications, etc.
- Offers quick functioning, access, control, monitoring and precise and exact results when compared to other wiring methods based devices



What Are The Parts of a PCB?

Substrate

- The first, and most important, is the substrate, usually made of fiberglass.
- Fiberglass is used because it provides a core strength to the PCB and helps resist breakage.
- Think of the substrate as the PCB's "skeleton".

Copper Layer

- Depending on the board type, this layer can either be copper foil or a full-on copper coating.
- Regardless of which approach is used, the point of the copper is still the same — to carry electrical signals to and from the PCB, much like your nervous system carries signals between your brain and your muscles.

What Are The Parts of a PCB?

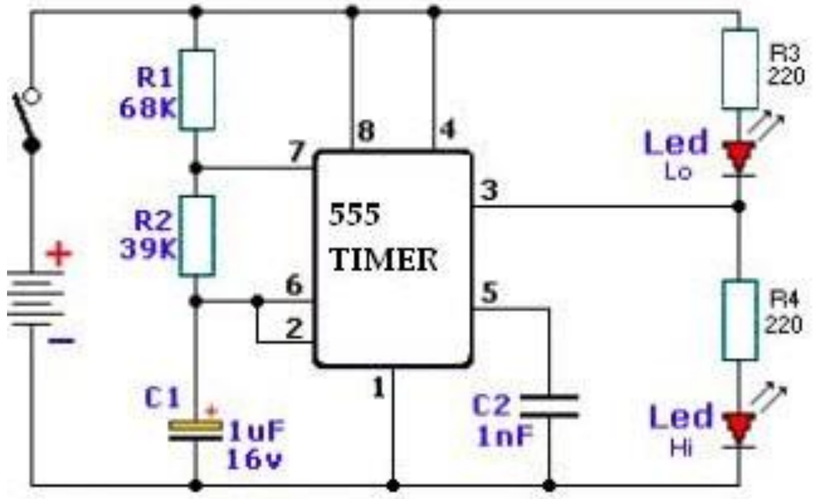
Solder Mask

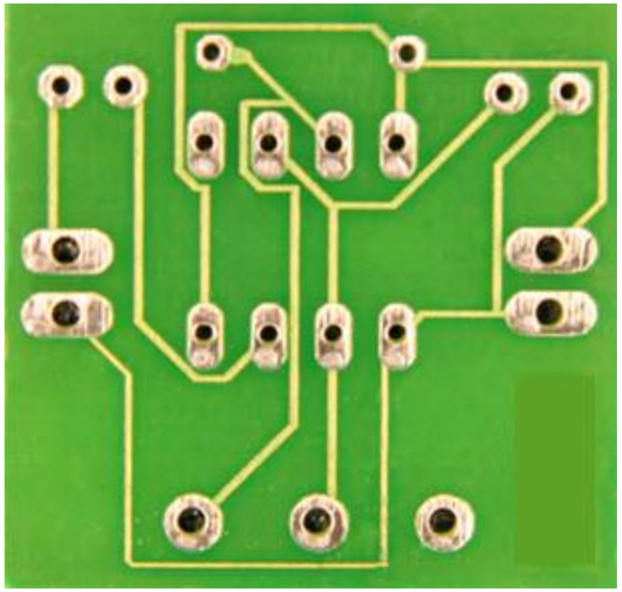
- The third piece of the PCB is the solder mask, which is a layer of polymer that helps protect the copper so that it doesn't short-circuit from coming into contact with the environment.
- In this way, the solder mask acts as the PCB's "skin".

Silkscreen

- The final part of the circuit board is the silkscreen.
- The silkscreen is usually on the component side of the board used to show part numbers, logos, symbols switch settings, component reference and test points.
- The silkscreen can also be known as legend or nomenclature

PCB Design Process

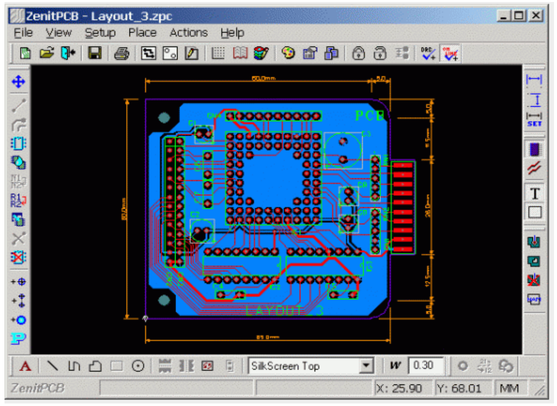


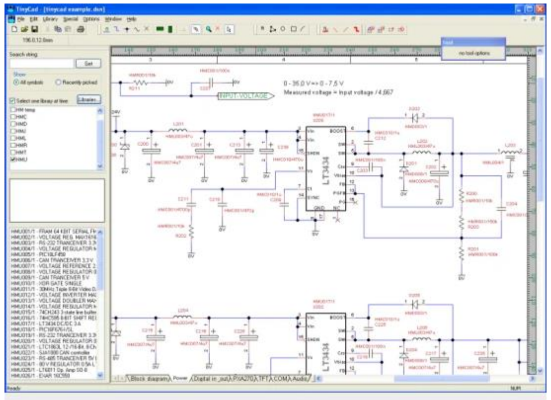


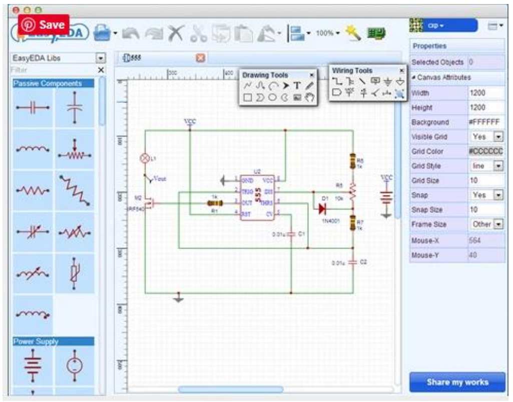
PCB Design Process

Step 1: Design the PCB Circuit with a Software

- Draw the schematic circuit diagram with the PCB layout software such as CAD software, Eagle and Multisim software.
- It is also possible to change the circuit design’s position and then to modify it according to your convenience and requirement.







PCB Design Process

Step 2: Film generation

- The film is generated from the finalized circuit board diagram of the PCB layout software which is sent to the manufacturing unit where the negative image or mask is printed out on a plastic sheet.

Step 3: Select Raw Material

- The bulk of the printed circuit board is made with an unbreakable glass or fiberglass having copper foil bonded unto one or both the sides of the board.
- The PCBs made from unbreakable paper phenolic with a bonded copper foil are less expensive and are often used in household electrical devices.

PCB Design Process

Step 4: Preparing Drill Holes

- Machines and carbide drills are used to put holes on the printed circuit board.
- There are two types of machines available to drill the PCBs;
  - Hand machines
  - CNC machines.
- The hand machines require human intervention or effort to drill the holes, whereas CNC machines are computer-based machines that work-based on the machine timetables or programs that run both automatic as well as manually.

PCB Design Process

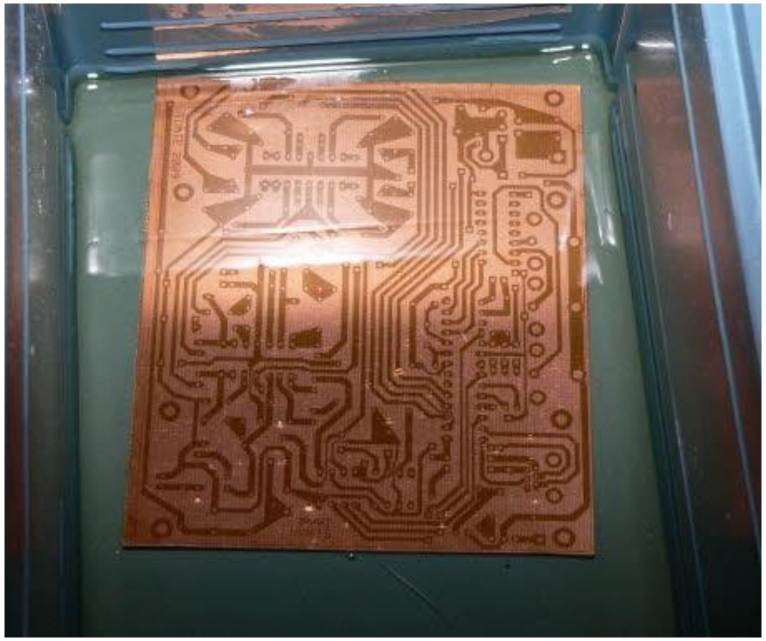
Step 5: Apply Image

- The printed circuit layout can be printed in different ways on PCBs like a manual pen, dry transfers, pen plotters, and printers. The laser printers are a better way to print the layouts on printed circuit boards.

PCB Design Process

Step 6: Stripping and Etching

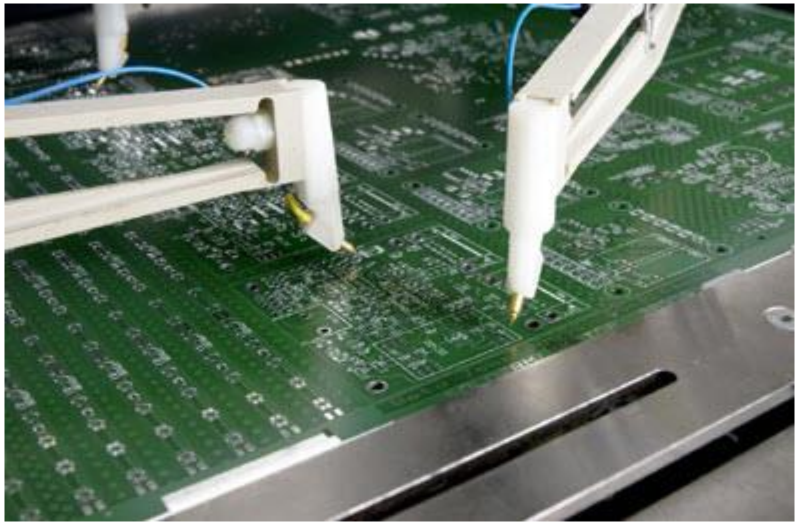
- This process involves removing the unwired copper on the PCBs by using different types of chemicals like ferric chloride, ammonium per-sulfate, etc.
- Make the solvent by mixing 1% of sodium hydroxide and 10 grams of sodium hydroxide pellets to one liter of water and mix it until everything is dissolved.



PCB Design Process

Step 7: Testing

- After finishing the manufacturing process of the Printed Circuit Board, the Board undergoes a testing process to check whether the PCB is working properly. Nowadays many automatic testing equipments are available for the high volume testing of the PCBs.



Memory Technologies

- A memory whose contents are lost when the power is cut off is called a volatile memory
- A microcomputer typically includes some amount of **RAM** (random access memory), which is a memory where individual items (bytes or words) can be written and read one at a time relatively quickly.
- **SRAM** (static RAM) is faster than **DRAM** (dynamic RAM), but it is also larger (each bit takes up more silicon area). DRAM holds data for only a short time
- **DRAM** is more volatile than **SRAM** because it loses its contents even if power is maintained

Non-Volatile Memory

- Embedded systems invariably need to store data even when the power is turned off
- Contents of **ROM** (Read Only Memory) is fixed during manufacture
- Program never changes is called firmware
- There are several variants of ROM that can be programmed in the field
- Flash memory is a form of non-volatile memory with reasonably fast read times but not fast as SRAM and DRAM

Storage

- What is a memory?
  - Artifact that stores bits
  - Storage fabric and access logic
- Write-ability
  - Manner and speed a memory can be written
- Storage-permanence
  - Ability of memory to hold stored bits after they are written
- Many different types of memories
  - Flash, SRAM, DRAM, etc.
- Common to compose memories

Write-ability

- Ranges of write ability
  - High end
    - Processor writes to memory simply and quickly
    - E.g., RAM
  - Middle range
    - Processor writes to memory, but slower
    - E.g., FLASH, EEPROM
  - Lower range
    - Special equipment, “programmer”, must be used to write to memory
    - E.g., EPROM, OTP ROM
  - Low end
    - Bits stored only during fabrication
    - E.g., Mask-programmed ROM

Storage-permanence

- Range of storage permanence
  - High end
    - Essentially never loses bits
    - E.g., mask-programmed ROM
  - Middle range
    - Holds bits days/months/years after memory’s power source turned off
    - E.g., NVRAM
  - Lower range
    - Holds bits as long as power supplied to memory
    - E.g., SRAM
  - Low end
    - Begins to lose bits almost immediately after written
    - E.g., DRAM

Activity – Familiarizing Arduino

- How many controllers are there in Arduino UNO board? Name them?
- Total Number of Analog and Digital Pins available?
- Is it possible to access both the controllers for writing codes?

Software Components

Software Engineer Tools

- There are five main development tools that allows software engineers to get started on development and testing of their applications.
- These are first **simulators**, software that imitates or intended to hardware's behavior without the actual hardware.
- **Emulators**, the hardboard platform that imitates the operation of your intended system.
- **Compilers**, it's the software that allows developers to pre-executable code for their intended architecture.
- **Installers**, a software-hardware combination that allows compiled executable programs to be installed onto a platform.
- And **debuggers**, a hardware-software solution that allows programmers to test and validate their executable programs.
- These tools are important because developing hardware takes time and it's expensive.

Software Engineer Tools

- The software engineer's role includes the design of the software product, along with its validation of the hardware platform.
- A complete hardware design and documentation usually finishes first, with software following.
- Simulators and emulators will allow validation of design to occur before the arrival of hardware.
- Compilers, Installers, and debuggers will provide a quick development of features for your embedded target. And an easy way to fix off our bugs.
- Prototyping software fast is important, but not at the cost of quality.

High quality code

- Write a code that is maintainable, testable, portable, robust, efficient and consistent.
- A common misconception is that the advanced platform a general programmer might use. That the advanced hardware will make up for poorly written software. This is not true, especially from an embedded perspective. because even the best hardware cannot make up for buggy, slow, or inefficient embedded code.
- By putting time into architecting your software, writing code effectively. Making it modular, you can extend the lifetime and re-usability of this pieces, of your various projects and platforms.

Embedded Software

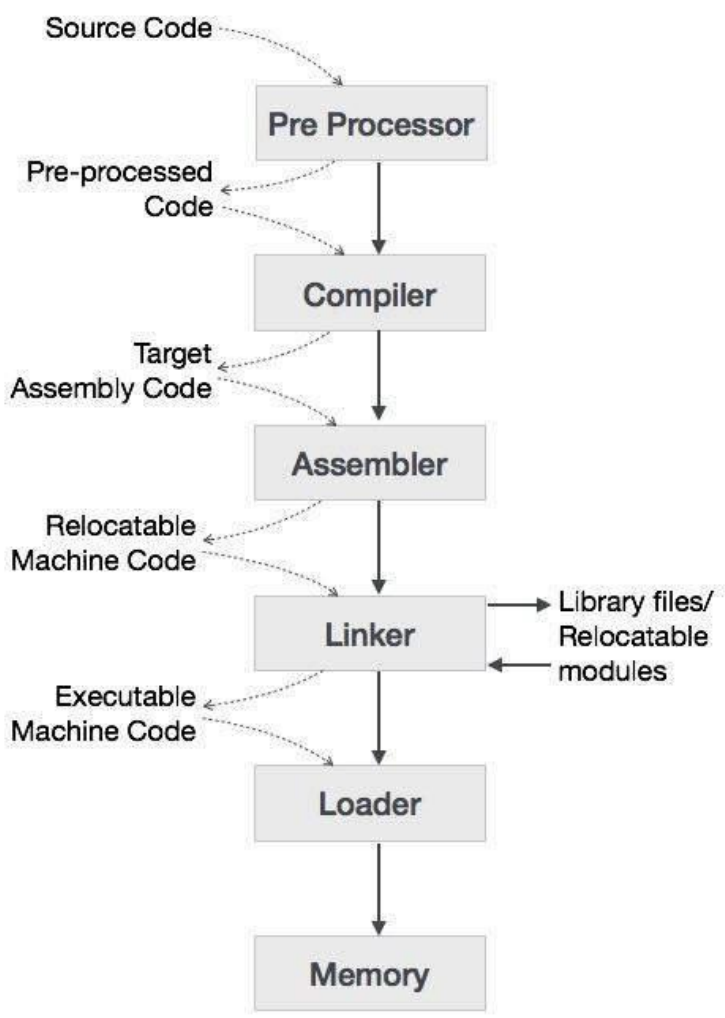
- There are many embedded software languages use in the industry such as C, C++, Java and Ada. But C-Programming is the most widely used language for embedded software design.
- C-Programming has benefits for both low level hardware interactions and high level software language features.
- This provides portability across different embedded platforms.
- Typical embedded engineers actually write a form of C called Embedded C.

Embedded C

- Embedded C differs from C because it puts a focus on the following features.
  - Efficient memory management
  - Timing centric operations
  - Direct hardware/IO control
  - Code size constraints
  - Optimized execution
- You can think of Embedded C as optimum features in minimum space and time.

Language Processing System

- The hardware understands a language, which humans cannot understand.
- We write programs in high-level language, which is easier for us to understand and remember.
- These programs are then fed into a series of tools and OS components to get the desired code that can be used by the machine. This is known as Language Processing System.



Language Processing System

How a program using C compiler is executed on a host machine.

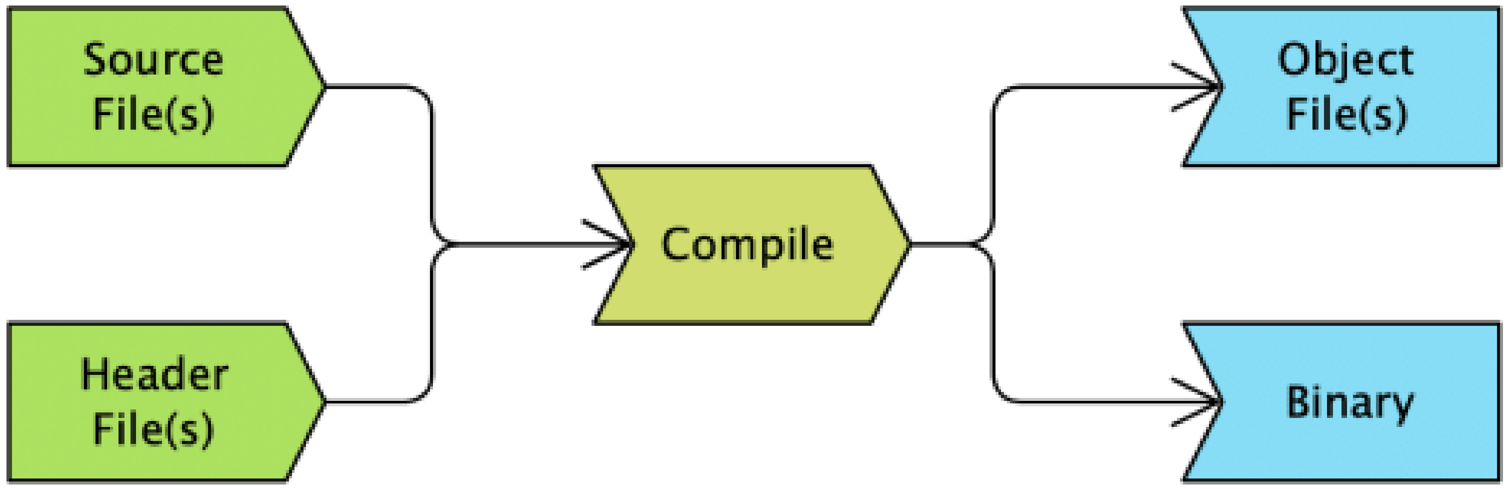
- User writes a program in C language (high-level language).
- The C compiler, compiles the program and translates it to assembly program (low-level language).
- An assembler then translates the assembly program into machine code (object).
- A linker tool is used to link all the parts of the program together for execution (executable machine code).
- A loader loads all of them into memory and then the program is executed.

Compilers

- A compiler is a program that translates code from one language (such as C, C++, or Rust) and outputs it into another language (such as assembly, object code, or machine code).
- A compiler can be used to create an executable binary directly, or can generate intermediate files that can be used during other software construction steps, such as linking.
- The name "compiler" is primarily used for programs that translate the source code from a highlevel programming language to a low-level language (e.g., assembly language or machine code).
- The most common compilers used for embedded systems are GCC, Clang, Keil, and IAR
- A compiler transforms those input files into object files (usually ending in .o)

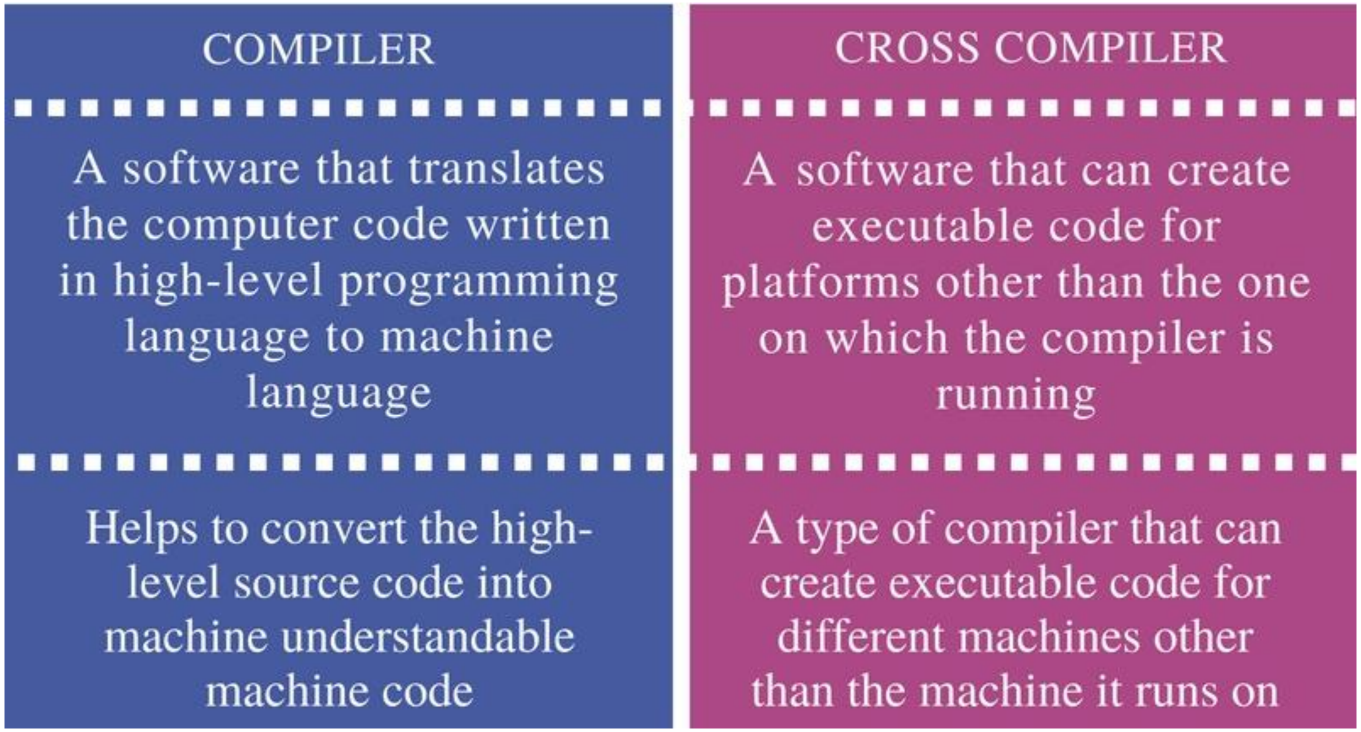
Compilation Process

- The inputs to a compiler are the source files and header files that define our program, as well as an array of optional flags that can be used to control the behavior of the compiler.
- A compiler transforms those input files into object files (usually ending in .o) or an executable program (usually ending in .bin, .hex, .out, .exe, .elf; sometimes there is no ending).



Cross Compiler

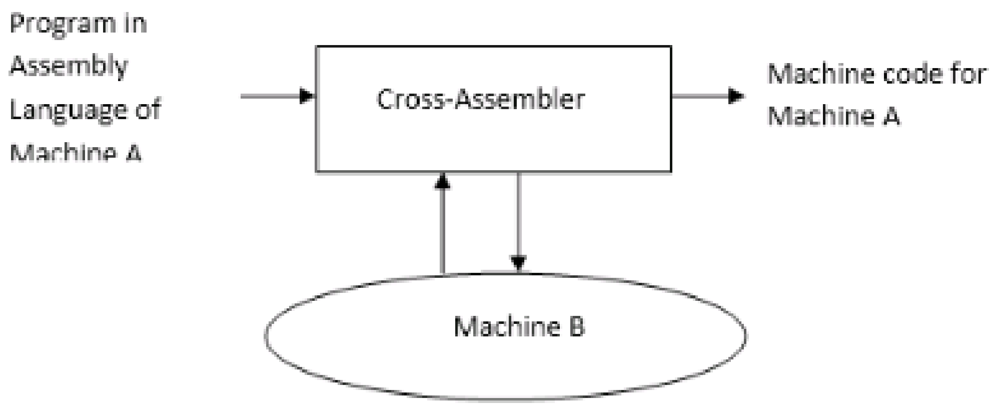
- If the compiled program can run on a system having different CPU or operating system than the system on which the compiler compiled the program, then that compiler is known as a cross-compiler.



COMPILER	CROSS COMPILER
A software that translates the computer code written in high-level programming language to machine language	A software that can create executable code for platforms other than the one on which the compiler is running
Helps to convert the high-level source code into machine understandable machine code	A type of compiler that can create executable code for different machines other than the machine it runs on

Cross Assembler

- A cross assembler takes the conversion process a step further by allowing you to generate machine code for a different processor than the one the compiler is run on.
- Cross assemblers are generally used to develop programs which are supposed to run on game consoles, appliances which are not able to run a development environment.


Program in Assembly Language of Machine A
Cross-Assembler
Machine code for Machine A
Machine B

Cross-Assembler

Linker

- Linker is a program in a system which helps to link a object modules of program into a single object file.
- It takes object modules from assembler as input and forms an executable file as output for loader.
- Linking is performed at both compile time, when the source code is translated into machine code and load time, when the program is loaded into memory by the loader.
- Linking is performed at the last step in compiling a program.

Run Time Library

- When the source code of a computer program is translated into the respective target language by a compiler, it would cause an extreme enlargement of program code if each command in the program and every call to a built-in function would cause the in-place generation of the complete respective program code in the target language every time.
- When the library files remain out of compilation of the main executable and comes into use only when required at runtime, it is called dynamic linking.
- These library files can be custom programmed or provided by the language compiler for standard functions.

  e.g. C compilers provide C-Standard runtime libraries
- For this reason, some programming bugs are not discovered until the program is tested in a "live" environment – **Runtime error**

PreProcessor

- The preprocessor is a part of the compiler which performs preliminary operations (conditionally compiling code, including files etc...) to your code before the compiler sees it.
- Preprocessor directives change the text of the source code and the result is a new source code without these directives.
- Before the actual compilation of every C program it is passed through a Preprocessor. The Preprocessor looks through the program trying to find out specific instructions called Preprocessor directives that it can understand. All Preprocessor directives begin with the # (hash) symbol.
- e.g. #include <stdio.h> , #define

Make Files

- A Makefile is the set of instructions that you use to tell compiler how to build your program and are a simple way to organize code compilation.
- This is present with other files in the parent folder of source code.
- The make utility requires a file, **makefile**, which defines set of tasks to be executed.
- You may have used make to compile a program from source code. Most open source projects use make to compile a final executable binary, which can then be installed using **make install**.

Compiler Tool chains

- A tool chain is a set of tools (chain of tools - compiler, linker, debugger, standard-libraries etc.) that are used to create an executable
- Tool chains are used in the embedded world for cross-compiling, which means creating a program on a host which will eventually run on a different kind of target platform - therefore there is a need to create it with a specific compiler, linker, debugger etc.

Popular tool chains

- GCC - GNU Compiler Collection - a free software collection of compilers, can be set up to cross compile.
- ARMCC - ARM Compiler tool chain – ARM processors are widely used in smartphones e.g. Qualcomm snapdragon processors
- Intel tool chain – proprietary high performance tool chain for Intel processors

Hardware, Software, Firmware, Middleware, Device Drivers, OS & Applications, The Difference?

Hardware

- What is Hardware? **Hardware is a system consisting of electronic devices, designed to work together as a single unit.**
- It is the only tangible "ware" out of our list that you can physically interact with. Examples include PCBs like motherboard, RAM, Integrated Circuits, Processor, Microcontroller, etc.
- For embedded engineers, the assembled system is the hardware.
- Hardware is named so since it is the hardest part to change in a given product's life cycle. If you need to change it, you need to design the entire PCB.

Firmware

- What is Firmware? **Firmware is a program that is specifically designed to work with particular hardware and it lives in non-volatile memory such a flash and it is executed directly from it.**
- Originally Firmware is written on Masked ROMs, on which the data cannot be changed once written and they run for ages till the device goes out of use.
- Nowadays these kinds of firmware can be found in devices like TVs, washing machines and microwaves. On your computer, it can be found in the BIOS of your motherboard.

Firmware

- The first replacement of Masked ROMs came in the form of EPROM which can be erased by exposure to UV light and then reprogrammed as required. Then came EEPROMs which used electricity to change the contents.
- Firmware is named so because it is not “hard” to change, at the same time it is not “easy/soft” to change.
- To change/update the firmware you need to download it, verify the integrity of the data, reboot the device and go into boot mode, then rewrite the flash memory and reboot it again in normal mode.
- It is the job of embedded firmware developers is to write this program.

Software

- What is Software? **Software is a program that can work on a wide variety of hardware and they are usually copied from non-volatile memory (like hard-disk or SSD) onto volatile memory (like SRAM and DRAM) before they can start executing.**
- Software is named so since they are very easy to change. The end-user can do it with no trouble at all.
- Examples of software can include Operating systems, all the applications that run on them, drivers and middleware.

Middleware

- What is Middleware? **Middleware is a computer program that connects 2 software together. The 2 software that needs to connect can be in the same machine or in 2 machines in the same room or it can be in 2 corners of the world.**
- The job of middleware is to combine the 2 programs and make a bigger one.
- The middleware in embedded systems, which is basically a layer of computer programs that sit between the Hardware Abstraction Layer and the application layer to help them communicate with each other.
- Named as this “ware” sits between 2 other “wares”

Device Drivers

- A program that controls a device attached to a computer.
- Device driver provides a software interface to hardware devices, enabling operating systems and other computer programs to access hardware functions without needing to know precise details about the hardware being used.
- An interface is nothing but a group of functions. All operating systems talk to hardware via some predefined software interfaces.
- e.g. NVIDIA graphics drivers, audio drivers in PCs

OS & Applications

- What is an operating system? **An operating system is a program that abstracts the underlying software with the aim of improving the efficiency and ease of use both for the end-users and application programmers.**
- An **OS** does its job by abstracting the memory, processing power and I/O such that multiple processes can run simultaneously by sharing the available hardware resources without the knowledge of each others' existence.
- **Applications** run on top of device drivers and operating systems. Example of applications include Browsers, Word processors, Multimedia players, etc.
- These applications cannot talk to the hardware directly. Instead, it does it through the operating system and its system calls.

Some more “wares”!

- **What are Adwares?**
- **What is Bloatware / Fatware?**
- **What is Vaporware?**

System security-related “wares” or computer programs.

- **What is Malware?**
- **What is Spyware?**
- **What is Ransomware?**

Debugging Tools

Debugging Tools

- Debugging/Testing is an important part of the overall development process which involves finding and solving bugs.
- A Virtual Testing Device is a software program on the computer, unlike a real device, that provides simulation for most of the important features of an actual device.
- It mimics the nature of the real device, which helps the testers to run the software application on it to get an idea about how it would run on the designated real device.
- These are of two types: Simulators and Emulators

Emulators

- An emulator is a software that mimics the hardware and software of the target device on your computer.
- They do this by translating the ISA (Instruction Set Architecture) of the target device to the one used by the computer you are using to conduct testing using binary translation.
- E.g. Android Emulators are used by Android app developers

Simulators

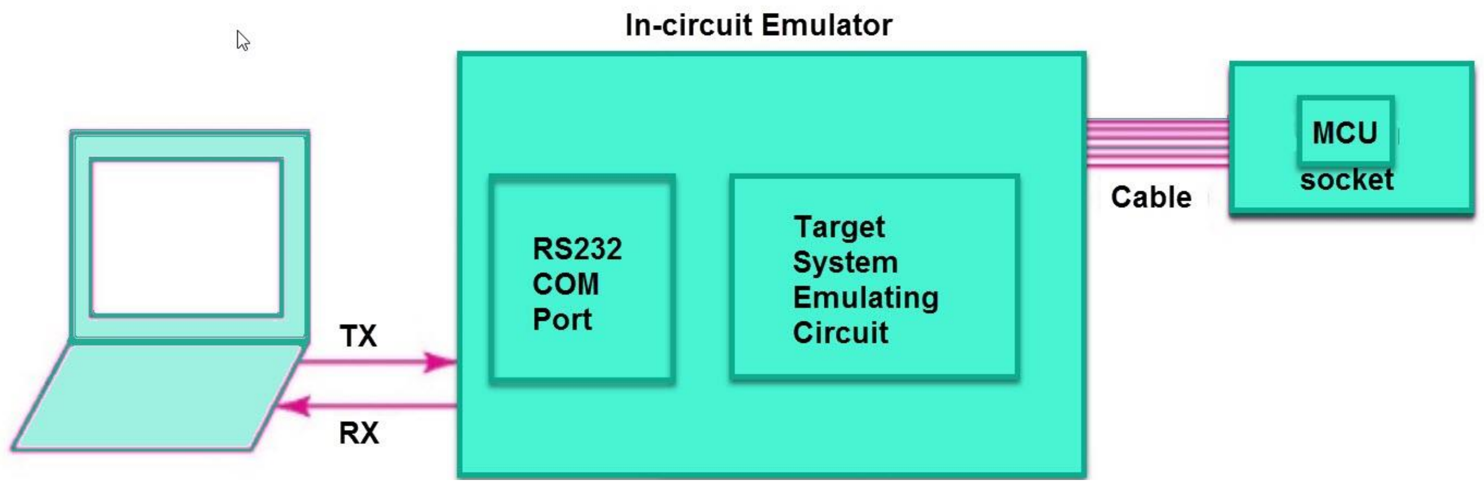
- A simulator is designed to create an environment that contains all of the software variables and configurations that will exist in an application's actual production environment.
- Simulators do not attempt to emulate the actual hardware that will host the application in production.
- Simulators create only software environments, they can be implemented using high-level programming languages.
  - SPICE simulator for circuit simulation

In Circuit Debuggers

- An in-circuit emulator (ICE) provides a window into the embedded system.
- The programmer uses the emulator to load programs into the embedded system, run them, step through them slowly, and view and change data used by the system's software.
- An emulator gets its name because it emulates (imitates) the central processing unit (CPU) of the embedded system's computer.
- Traditionally it had a plug that inserts into the socket where the CPU integrated circuit chip would normally be placed.

In Circuit Debuggers

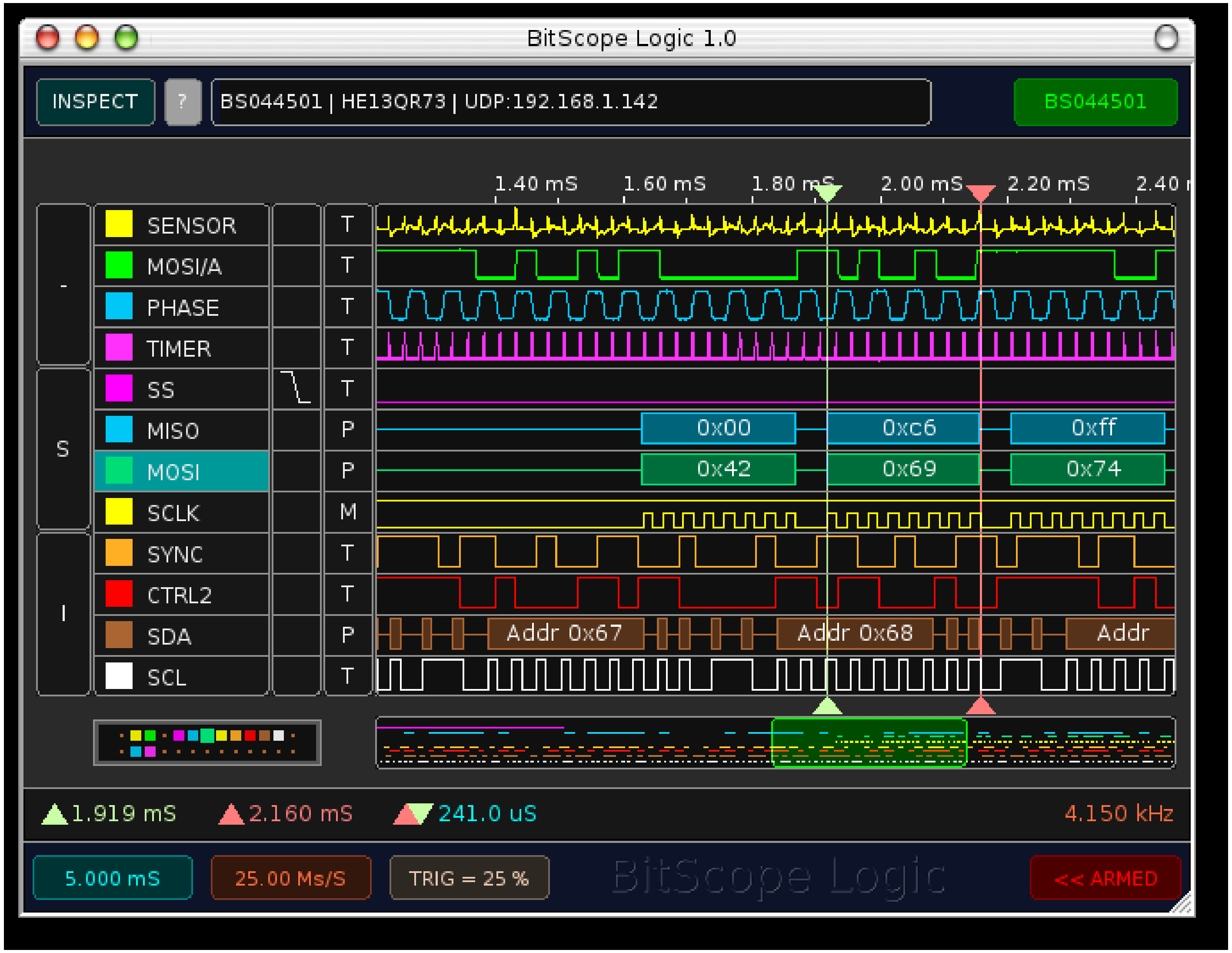
- ICEs attach a computer terminal or personal computer (PC) to the embedded system.
- The terminal or PC provides an interactive user interface for the programmer to investigate and control the embedded system.



Logic Analyser

- Developed out of the need to be able debug and undertake fault-finding on microprocessor based systems, need for simultaneous monitoring of many lines and test points.
- Widely used to develop and debug complex digital electronic logic circuits
- Displays traces of multiple logic channels and reveal the circuit operation.
- Displays relative timing of a large number of signals, enabling traces of logic signals to be seen in such a way that the operation of several lines in a digital circuit can be monitored and investigated.

Logic Analyser



Logic Analyser

- Multiple channels
  - 32 - 200+ channels, each channel monitoring one digital line
- Provide a time display of logic states
  - Horizontal time axis and a vertical axis to indicate a logic high or low states
- Display logic states
  - Signals entering various channels are converted into a high or low state for further processing within the analyzer. It provides a logic timing diagram of the various lines being monitored.
- Does NOT display analog information
  - Monitors the logic operation of the system.

Logic Analyser Applications

- Verifying and debugging the operation of digital designs looking a logic states and timings.
- Correlate a large number of digital signals
- Investigate the system operation.
- Detect timing violations
- Trace embedded software operation.

Integrated Development Environment

- An IDE, or Integrated Development Environment, enables programmers to consolidate the different aspects of writing a computer program.
- IDEs increase programmer productivity by combining common activities of writing software into a single application: editing source code, building executables, and debugging.
  - PyCharm for Python language;
  - Microsoft visual studio for C, C++, C# etc. languages

Thank You