# Module IV

## INTEGRATION AND TESTING OF EMBEDDED HARDWARE AND FIRMWARE

*Integration and Testing of Embedded Hardware and Firmware- Integration of Hardware and Firmware. Embedded System Development Environment – IDEs, Cross Compilers, Disassemblers, Decompilers, Simulators, Emulators and Debuggers.*

### Integration of Hardware and Firmware

Integration of hardware and firmware deals with the embedding of firmware into the target hardware board. It is the process of 'Embedding Intelligence' to the product. A variety of techniques are used for embedding the firmware into the target board. The commonly used firmware embedding techniques for a non-OS based embedded system are given below:

1. Out of circuit programming
2. In System Programming
3. In Application Programming

1. ***Out-of-Circuit Programming***

Out-of-circuit programming is performed outside the target board. The processor or memory chip into which the firmware needs to embedded is taken out of the target board and it is programmed with the help of programming device. The programming device is dedicated unit when contains the necessary hardware circuit to generate the programming signals. The programmer contains a ZIF socket with locking pin to hold the device to be programmed. The programming device will be under the control of a utility program running on a PC. The programmer is interfaced to the PC through RS-232C/USB/Parallel Port Interface. The commands to control the programmer are sent from the utility program to the programmer through the interface.

<u>Operations for embedding the firmware with a programmer.</u>

- Connect the programming device to the specified port of PC (USB/COM port/parallel port)
- Power up the device.(Ensure that the power indication LED is ON)
- Execute the programming utility on the PC and ensure proper connectivity is established between the PC and programmer. In case of error turn off device power and try connecting it again.
- Unlock the ZIF socket by turning the lock pin.

KTU NOTES the learning companion

For more visit www.ktunotes.in

- Insert the device to be programmed into the open socket as per the insert diagram shown on the programmer.
- Lock the ZIF socket.
- Select the device name from the list of supported devices.
- Load the hex file which is to be embedded into the device.
- Program the device by 'Program' option of utility program.
- Wait till the completion of programming operation.(Till busy LED of programmer is off)
- Ensure that programming is successful by checking the status LED on the programmer (Usually 'Green' for success and 'Red' for error condition) or by noticing the feedback from the utility program.
- Unlock the ZIF socket and take the device out of programmer.

Drawbacks:

- High development time.

Whenever the firmware is changed, the chip should be taken out of the board for reprogramming. This tedious and prone to chip damages due to frequent insertion and removal. Better use a socket on the board side to hold the chip till the firmware modifications are over.

- Not suitable for batch production.

2. ***In System Programming (ISP)***

In ISP, programming is done within the system. The firmware is embedded into the target device without removing it from the target board. It is the flexible and easy way of firmware embedding. The only pre-requisite is that the device must have an ISP support. A part from target board, PC, ISP utility and ISP cable, no other additional hardware is required for ISP. In order to perform ISP operations the target device should be powered up in a special 'ISP mode'. ISP mode allows the device to communicate with an external host through a serial interface, such as a PC or terminal. The device receives commands and data from the host, erases and reprograms code memory according to the received command. Once the operations are completed, the device is re-configured so that it will operate normally.

In System Programming with SPI Protocol

Devices with SPI In System Programming support contains a built-in interface and a on-chip EEPROM or FLASH memory is programmed through this interface.

KTU NOTES the learning campanion

For more visit www.ktunotes.in

The primary I/O lines involved in SPI-In System Programming are listed below,

a. MOSI – Master Out Slave In
b. MISO – Master In Slave Out
c. SCK – System Clock
d. RST – Reset of Target Device
e. GND – Ground of Target Device

The power up sequence for In System Programming for Atmel's AT89S series microcontroller:

- Apply supply voltage between VCC and GND pins of target chip.
- Set RST pin to "HIGH" state.
- If a crystal is not connected across pins XTAL1 and XTAL2, apply a 3 MHz to 24MHz clock to XTAL1 pin and wait for at least 10 milliseconds.
- Enable serial programming by sending the Programming Enable serial instruction to pin MOSI/P1.5. The frequency of the shift clock supplied at pin SCK/P1.7 needs to be less than the CPU clock at XTAL1 divided by 40.
- The Code or Data array is programmed one byte at a time supplying the address and data together with the appropriate Write instruction. The selected memory location is first erased before the new data is written. The write cycle is self – timed and typically takes less than 2.5 ms at 5V.
- Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO/P1.6.
- After successfully programming the device, set RST pin low or turn off the chip power supply and turn it ON to commence the normal operation.

***3. In Application Programming (IAP)***

In Application Programming (IAP) is a technique used by the firmware running on the target device for modifying a selected portion of the code memory. It is not a technique for first time embedding of user written firmware. It modifies the program code memory under the control of embedded application. Updating calibration data, look-up tables, etc., which are stored in the code memory, are typical examples of IAP. The Boot ROM resident API instructions which perform various functions such as programming, erasing and reading the Flash memory during ISP mode are made available to the end-user written firmware for IAP.

KTU NOTES the learning companion

For more visit www.ktunotes.in

**Firmware Loading for Operating System based Devices**

It is possible to embed the firmware into the target processor/controller memory at the time of chip fabrication itself. Such chips are known as 'Factory Programmed Chips'. The OS based embedded systems are programmed using the In System Programming (ISP) technique. OS based embedded systems contain a special piece of code called 'Boot loader' program which takes control of the OS and application firmware embedding and copying of the OS image to the RAM of the system for execution.

**EMBEDDED SYSTEM DEVELOPMENT ENVIRONMENT**

**Components of Embedded development environment**

- **Host Computer**
  Acts as the heart of development environment.
- **IDE Tools**
  Tools for firmware design and development
- **Electronic Design Automation Tools**
  Embedded Hardware Design
- **Emulator hardware**
  Debugging target board
- **Signal Sources(function generator)**
  Simulates inputs to target board
- **Target Hardware Debugging tools**
  CRO, Multimeter ,Logic Analyser
  For debugging hardware
- **Target Hardware**

**IDE**

In Embedded System, IDE stands for an integrated environment for developing and debugging the target processor specific embedded firmware. An IDE is also known as integrated design environment or integrated debugging environment. IDE is a software package which bundles a "Text Editor", "Cross-compiler", "Linker" and a "Debugger" IDE is a software application that provides facilities to computer programmers for software development. IDEs can either command line based or GUI based. IDE consists of

1. Text Editor or Source code editor
2. A compiler and an interpreter

KTU NOTES the learning campanion

For more visit www.ktunotes.in

3. Build automation tools
4. Debugger
5. Simulators
6. Emulators and logic analyzer

An example of IDE is Turbo C/C++ which provides platform on windows for development of application programs with command line interface. The other category of IDE is known as Visual IDE which provides the platform for visual development environment, ex- Microsoft Visual C++. IDEs used in Embedded firmware are slightly different from the generic IDE used for high level language based development for desktop applications. In Embedded applications, the IDE is either supplied by the target processor/controller manufacturer or by third party vendors or as Open source.

**Cross Compilation**

Cross compilation is the process of converting a source code written in high level language to a target processor/controller understandable machine code. The conversion of the code is done by software running on a processor/controller which is different from the target processor. The software performing this operation is referred as the Cross-compiler. In other words cross-compilation the process of cross platform software/firmware development. A cross complier is a compiler that runs on one type of processor architecture but produces object code for a different type of processor architecture.

**Need for Cross Compiler**

There are several advantages of using cross compiler. Some of them are as below:

-By using cross compliers we can not only develop complex Embedded System but reliability can be improved and maintenance is easy.

- Knowledge of the processor instruction set is not required.

-Register allocation and addressing mode details are managed by the compiler.

-The ability to combine variable selection with specific operations improves program readability.

-Keywords and operational functions that more nearly resemble the human thought process can be changed.

-Program development and debugging time will be dramatically reduced when compared to assembly language programming.

KTU NOTES
the learning campanion

For more visit www.ktunotes.in

-The library files that are supplied provide may standard routines that may be incorporated into our application.

-Existing routine can be reused in new programs by utilizing the modular programming techniques available with C.

-The C language is very portable and very popular.

**Types of Files generated on Cross compilation**

The various files generated during cross- compilation process are:

1. List File
2. Pre-processor Output file
3. Hex File (.hex)
4. Map File (File extension linker dependent)
5. Object File (.obj)

***1.List Files (.lst files)***

- Generated at the time of cross compilation
- Contain information about cross compilation process like
  - Cross compiler details
  - Formatted source text
  - Assembling code generated from the source file
  - Symbol table
  - Errors and warning detected by the cross compiler system

***2.Preprocessor output file***

- generated during cross compilation
- contain preprocessor output for the preprocessor instructions used in the source file
- This file is used for verifying the operation of macros and conditional preprocessor directives is a valid C file
- file extension is cross compiler dependent

***3.Hex file***

- The Hex file is an ASCII text file with lines of text that follow the Intel Hex file format.
- Intel Hex files are often used to transfer the program and data that would be stored in a Rom or EPROM.

KTU NOTES the learning companion

For more visit www.ktunotes.in

***4.Map Files***

- Object file created contains relocatable codes i.e their location in memory is not fixed.
- It is the responsibility of linker to link these object modules. The locator is responsible for locating the absolute address to each module in the code memory. Map files are generated by the linker and loader. These files are used to keep the information of linking and locating process. Map files use extensions .H,.HH,.HM depends on linker or loader

***5. Object files***

- It is the lowest level file format for any platform.
- Cross compiling each source module converts the various Embedded instructions and other directives present in the module to an object (.OBJ) file.
- The object file is specially formatted file with data records for symbolic information, object code, debugging information etc.
- OMF 1 & OMF2 are the 2 object files supported by C51 Cross compiler

List of details included in object file are

1. Reserved memory for global variables
2. Public symbol(variable or function)names
3. External symbol(variable or function)references
4. Library files with which to link
5. Debugging information to help synchronize source lines with object files

**DISASSEMBLER/DECOMPILER**

- Both are reverse engineering tools.
- Reverse engineering is a technology used to reveal the technology behind the working of a product

  -used to find out the secret behind popular proprietary product

  -helps the reverse engineering process by translating embedded firmware to assembly /high level instruction

  -Powerful tools for analyzing the presence of malicious contents

**DISASSEMBLER**

-utility program that convert machine code into assembly code.

-It is complementary to assembly or cross assembly

KTU NOTES the learning campanion

For more visit www.ktunotes.in

**DECOMPILER**

-is a utility program that convert machine language instruction to high level language instruction.

- Performs reverse operation of compiler or cross compiler.

**SIMULATORS, EMULATORS AND DEBUGGING**

**Simulators**

-Simulator is a software tool for simulating various functionality of the application software.IDE provides simulator support. Simulator simulates target hardware and firmware execution can be simulate using simulators

Features of simulator:

1. Purely software based
2. Doesn't require a real target system
3. Very primitive
4. Lack of real time behavior

Limitations:

1. Deviation from real behavior
2. Lack of real timeliness

Advantage of simulator based debugging

1. *No need of target board*

- Purely software oriented, IDE simulates the target board
- Since real hardware is not needed we can start immediately after the device interface and memory maps are finalized this saved development time

2. *Simulated I/O peripherals*

- It eliminates the need for connecting IO devices for debugging the firmware

3. *Simulates abnormal conditions*

- Can input any parameter as input during debugging hence can check for abnormal conditions easily

**EMULATOR**

- It is a piece of hardware that exactly behaves like the real microcontroller chip with all its integrated functionality.
- It is the most powerful debugging of all.
- A microcontroller's functions are emulated in real-time and non-intrusively.
- All emulators contain 3 essential function:

1. The emulator control logic, including emulation memory

KTU NOTES the learning companion

For more visit www.ktunotes.in

2. The actual emulation device

3. A pin adapter that gives the emulator's target connector the same "package" and pin out as the microcontroller to be emulated.

- An emulator is a piece of hardware that looks like a processor, has memory like a processor, and executes instructions like a processor but it is not a processor.
- The advantage is that we can probe points of the circuit that are not accessible inside a chip.
- It is a combination of hardware and software.

**DEBUGGERS**

***What is Debugging?***

- Debugging in embedded application is the process of diagnosing the firmware execution, monitoring the target processor's registers and memory while the firmware is running and checking the signals from various buses of embedded hardware.
- Classified as
    - Hardware Debugging
    - Firmware Debugging
- Hardware Debugging:
    - Deals with monitoring of various bus signals and checking the status lines of target hardware.
- Firmware Debugging:
    - Deals with examining the firmware execution, execution flow, changes to various CPU registers and status registers on execution of the firmware to ensure that the firmware is running as per the design

***Why is Debugging required?***

- Firmware Debugging is performed to figure out the bug or the error in the firmware which creates the unexpected behavior.

***Debugging Techniques***

***1. Incremental EEPROM Burning Technique:***

- Most primitive technique
- Code is separated into different functional code units.
- Code is burned into EEPROM in incremental order.
- LED or BUZZER provided to indicate correct functioning
- Time Consuming but is a onetime process.

KTU NOTES the learning campanion

For more visit www.ktunotes.in

***2. Inline Breakpoint Based Firmware Debugging***

- An inline debug code is inserted within the firmware at a point where we want to ensure correct execution
- Debug code is ***printf()*** function.
- View debug code generated data on the "Hyper Terminal"
    - Configure serial communication setting of the Hyper Terminal connection to the same as that of serial communication setting configured in the firmware (Baudrate, Polarity, Stop Bit, Flow Control).
    - Connect target board's serial port to development PC's COM port using RS232 cable.

***3. Monitor Program Based Firmware Debugging***

- Monitor Program which acts as supervisor is developed.
- It controls the downloading of user code into code memory, inspect and modifies register/memory locations, allow single stepping of source code etc.
- It always listen to serial port of target device and according to command received it performs command specific actions.
- The first step in any monitor program development is determining a set of commands for performing various operations like firmware downloading, memory register inspection/modification, single stepping, etc. Once the commands for each operation is fixed write the code for performing the actions corresponding to these commands.
- The commands may be received through any of the external interface of the target processor (e.g. RS-232C serial interface/parallel interface/USB, etc.).
- The monitor program should query this interface to get commands or should handle the command reception if the data reception is implemented through interrupts. On receiving a command, examine it and perform the action corresponding to it.
- The entire code handling the command reception and corresponding action implementation is known as the "monitor program".
- After the successful completion of the monitor program development, it is compiled and burned into the FLASH memory or ROM of the target board. The code memory containing the monitor program is known as the Monitor ROM

<u>The Monitor program contains the following set of minimal features:</u>

- Command set interface to establish communication with the dubbing application
- Firmware download option to code memory

KTU NOTES the learning companion

For more visit www.ktunotes.in

- Examine and modify processor registers and working memory(RAM)
- Single step program execution
- Set breakup points in firmware execution
- Send debug info to debug application running on host machine

- The monitor program usually resides at the reset vector (code memory 0000H) of the target processor
- The actual code memory is downloaded into a RAM chip which is interfaced to the processor in the Von Neumann architecture model.
- The von Neumann architecture model is achieved by ANDing the PSEN and RD signals of the target processor (In case of 8051) and connecting the output of AND Gate to the Output Enable (RD) pin of RAM chip. WR signal of the target processor is interfaced to the WR signal of the Von Neumann RAM.
- An address decoder circuit maps the address range allocated to the monitor ROM and activate the Chip Select (CS) of the ROM if the address is within the range specified for the Monitor ROM.
- A user program is normally loaded at location 0x4000 or 0x8000. The address decoder Circuit ensures the enabling of the RAM chip (CS) when the address range is outside that allocated to the ROM monitor.

KTU NOTES
the learning companion

The major drawbacks of monitor based debugging system are

- The entire memory map is converted into a Von Neumann model and it is shared between the monitor ROM, monitor program data memory, monitor program trace buffer, user written firmware and external user memory. For 8037, the original Harvard architecture supports 64K code memory and 64K external data memory (Total 128K memory map). Going for a monitor based debugging shrinks the total available memory to 64K Von Neumann memory and it needs to accommodate all kinds of memory requirement
- The communication link between the debug application running on Development PC and monitor program residing in the target system is achieved through a serial link and usually the controller's On-chip UART is used for establishing this link. Hence one serial port of the target processor becomes dedicated for the monitor application and it cannot be used for any other device interfacing. Wastage of serial port! It is a serious issue in controllers or processors with single UART

For more visit www.ktunotes.in

***4. In Circuit Emulator Based Firmware Debugging***

Functional Units

- Emulation Device
- Emulation Memory
- Emulator Control Logic
- Device Adaptors

*Emulation Device: replica of target CPU*

- Standard Chip
- Programmable Logic Device(PLD)

*Emulation Memory: Replacement for EEPROM of target device*

- It is the RAM incorporated in the emulator device
- Acts as Trace Buffer. Trace buffer is a memory pool holding the instructions executed/registers modified/related data by the processor while debugging. The common features of trace buffer memory and trace buffer data viewing are listed below. Trace buffer records cache bus cycle in frames. Trace data can be viewed in the debugger application as Assembly/Source code. Trace buffering can be done on the basis of a Trace trigger (Event). Trace buffer can also record signals from target board other than CPU signals. Trace data is a very useful information in firmware debugging

*Device adaptors*

- Act as an interface between the target board and emulator POD.
- Normally pin-to-pin compatible sockets which can be inserted/plugged into the target board for routing the various signals from the pins assigned for the target processor.
- The device adaptor is usually connected to the emulator POD using ribbon cables. The adaptor type varies depending on the target processor chip package.

- The above mentioned emulators are almost dedicated ones, meaning they are built for emulating a specific target processor and have little or less support for emulating the derivatives of the target processor for which the emulator is built. This type of emulators usually combines the entire emulation control logic and emulation device in a single board. They are known as Debug Board Modules (DBMs).
- An alternative method of emulator design supports emulation of a variety of target processors. Here the emulator hardware is partitioned into two
    - Base Terminal

KTU NOTES the learning campanion

For more visit www.ktunotes.in

- Probe Card.

- The Base terminal contains all the emulator hardware and emulation control logic except the emulation chip (Target board CPUs replica). The base terminal is connected to the Development PC for establishing communication with the debug application. The emulation chip is mounted on a separate PCB and it is connected to the base terminal through a ribbon cable.
- The Probe Card board contains the device adaptor sockets to plug the board into the target development board. The board containing the emulation chip is known as the Probe Card. For emulating different target CPUs the Probe Card will be different and the base terminal remains the same.

***5. On Chip Firmware Debugging***

- Today almost all processors incorporate built in debug modules called On Chip Debug (OCD) support.
- Though OCD adds silicon complexity and cost factor, from a developer perspective it is a very good feature supporting fast and efficient firmware debugging.
- The On Chip Debug facilities integrated to the processor/controller are chip vendor dependent and most of them are proprietary technologies like Background Debug Mode. Some vendors add 'on chip software debug support through JTAG (Joint Test Action Group) port.
- Processors/controllers with OCD support incorporate a dedicated debug module to the existing architecture
- Usually the on-chip debugger provides the means to set simple breakpoints, query the internal state of the chip and single step through code.
- OCD module implements dedicated registers for controlling debugging.
- An On Chip Debugger can be enabled by setting the OCD enable bit
- BDM and JTAG are the two commonly used interfaces to communicate between the Debug application running on Development PC and OCD module of target CPU.
- Background Debug Mode (BDM) interface is a proprietary On Chip Debug solution from Motorola
- BDM defines the communication interface between the chip resident debug core and host PC where the BDM compatible remote debugger is running.
- BDM makes use of 10 or 26 pin connector to connect to the target board.

KTU NOTES the learning companion

For more visit www.ktunotes.in

- Serial data in (DSI), Şerial data out (DS0) and Serial clock (DSLR) are the three major signal lines used in BDM.
- DSI sends debug commands serially to the target processor from the remote debugger application
- DSO sends the debug response to the debugger from the processor
- Synchronisation of serial transmission is done by the serial clock DSCLK generated by the debugger application. Debugging is controlled by BDM specific debug commands. The debug commands are usually 17-bit wide. 16 bits are used for representing the command and I bit for status/control.
- Chips with JTAG debug interface contain a built-in JTAG port for communicating with the remote debugger application.

  The signal lines of JTAG protocol are explained below

  - Test Data In (TDI): It is used for sending debug commands serially from remote debugger to the target processor
  - Test Data Out (TDO): Transmit debug response to the remote debugger from target CPU.
  - Test Clock (TCK): Synchronizes the serial data transfer
  - Test Mode Select (TMS): Sets the mode of testing.
  - Test Reset (TRST): It is an optional signal line used for resetting the target CPU.

KTU NOTES
the learning campanion

**TARGET HARDWARE DEBUGGING**

Firmware is bug free and everything is intact with the board. Still embedded product need not function as per the expected behavior in the first attempt. This is due to hardware related reasons like :

- Dry soldering of components
- Missing connections in PCB due to un-noticed error in PCB layout design
- Misplaced components
- Signal corruption due to noise

**Hardware Debugging**

- The monitoring of various signals of the target board like address/data lines, checking the inter connection among various components, circuit continuity checking etc.
- Not similar to firmware debugging

For more visit www.ktunotes.in

Hardware Debugging Tools

- Magnifying Glass
- Multimeter
- Digital CRO
- Logic Analyser
- Function Generator

***Magnifying Glass***

- It is a powerful visual inspection tool.
- Used for examining the target board for :
  - Dry soldering components
  - Missing components
  - Improper placement of components
  - Improper soldering
  - Tracks PCB damage
  - Short of track

***Multimeter***

- Used for measuring various electrical quantities like Voltage, Currents, Resistance, Capacitance, Continuity checking, Transistor checking, Cathode and Anode identification of diode etc.
- Primary debugging tool for physical contact based hardware debugging.
- It is mainly used in embedded hardware debugging for:
  - Checking the circuit continuity between different points on the board
  - Measuring the supply voltage
  - Checking the signal value, polarity etc.
  - Both analog and digital version are available

***Digital Cathode Ray Oscilloscope***

Used for

- Waveform capturing and analysis
- Measurement of signal strength
- Analysing interference noise in the power supply line and other signal lines
- Connecting point under observation on the target board to the channels of the oscilloscope, waveform can be captured and analysed for expected behavior.

KTU NOTES the learning companion

For more visit www.ktunotes.in

- Digital CRO are available for high frequency support and incorporates modern technique for recording waveform over a period of time, capturing the waves on the basis of a configurable event for target board.

***Logic Analyser***

- It is similar to CRO
- Used to capture digital data (logic 1 or 0) from digital circuitry, whereas CRO is used to capture all kind of waves.
- The total no of logical signals that can be captured using CRO is limited to the no of channels. Logical analyser contain special connectors and chips which can be attached to the target board for capturing the digital data.
- In target board debugging applications, a logic analyser captures the states of various port pins, address bus and data bus of the target processor/controller, etc.
- Logic analysers give an exact reflection of what happens when a particular line of firmware is running.

***Function Generator***

- Function generator produce various periodic waveforms like sine wave,square wave, saw-tooth wave etc. with different frequency and amplitude.
- It is not a debugging tool, it is an input stimulator tool.
- The target board requires periodic waveform with particular frequency as input to some part of the board.

KTU NOTES
the learning campanion

For more visit www.ktunotes.in