Module 4

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.

Learning Objectives

- Learn the different techniques for embedding firmware into hardware
  - steps involved in the Out of system Programming
  - In System Programming (ISP) for firmware embedding
  - In Application Programming (IAP) for altering an area of the program storage memory for updating configuration data, tables, etc.
  - technique used for embedding OS image and applications into the program storage memory of an embedded device

Learning Objectives

- Discuss the different entities of the embedded system development environment
  - Integrated Development Environments (IDEs) for embedded firmware development and debugging
  - Identify the different types of files generated
    - during the cross compilation of a source file written in high level language like Embedded C
    - during the cross assembling of a source file written in Assembly Language
  - Discuss about disassembler and decompiler, and their role in embedded firmware development
  - Explain Simulators, In Circuit Emulators (ICE), and Debuggers and their role in embedded firmware debugging

Introduction

- **Integration testing of the embedded hardware and firmware** is the immediate step following the embedded hardware and firmware development.
- The final **embedded hardware constitute** of a PCB with all necessary components affixed to it as per the original schematic diagram.
- **Embedded firmware represents** the control algorithm and configuration data necessary to implement the product requirements on the product.
- Embedded firmware will be in a target processor/controller understandable format called machine language
- Both embedded hardware and firmware should be **independently tested** (Unit Tested) to ensure their proper functioning.
- Functioning of individual hardware sections can be verified by writing small utilities which checks the operation of the specified part.
- Functionalities of embedded firmware can easily be checked by the simulator environment provided by the embedded firmware development tool's IDE.

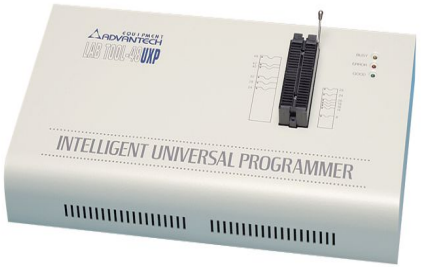
INTEGRATION OF HARDWARE AND FIRMWARE

- Deals with embedding of firmware into the target hardware board
- It's the process of *embedding intelligence* to the product
- The embedded processors/controllers used in the target board may or may not have built in code memory
- For **non-operating system based embedded products**, if the processor / controller contains internal memory and the total size of the firmware is fitting into the code memory area, the code memory is downloaded into the target controller/processor.
- If the processor/controller does not support built in code memory or the size of the firmware is exceeding the memory size supported by the target processor/controller, EPROM/Flash chips are used
- This chip is interfaced to the processor/ controller

- A variety of techniques are used for embedding the firmware into the target board.
- The non-OS based embedded systems store the firmware either in the onchip processor/controller memory or offchip memory (FLASH/NVRAM, etc.).
- The commonly used firmware embedding techniques for a non-OS based embedded system are :
  - Out-of-Circuit Programming
  - In System Programming(ISP)
  - In Application Programming
  - Use of Factory programmed chip

Out-of-Circuit Programming

- Its performed outside the target board

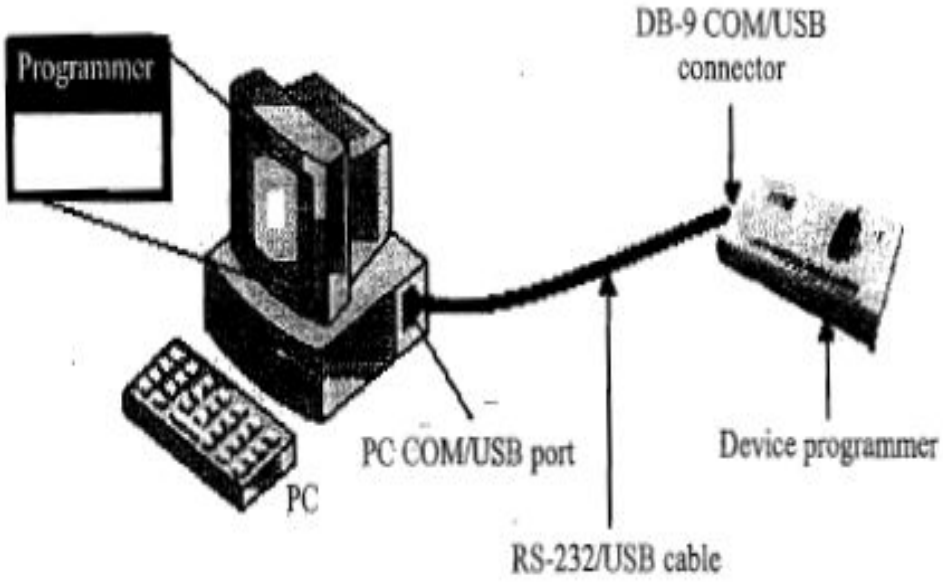


Labtool-48UXP

Out-of-Circuit Programming cont..

- The processor or memory chip into which the firmware needs to be embedded is taken out of the target board and its programmed with the programming device
- The programming device is a dedicated unit which contains the necessary hardware circuits generate the programming signals
- The programmer contains ZIF socket locking pin to hold the device to be programmed
- Programmer is interfaced to pc through RS232/USB/Parallel port interface

Out-of-Circuit Programming cont..



- Interfacing of device programmer with pc

Out-of-Circuit Programming cont..

- The sequence of operations for embedded firmware with a programmer is

1. Connect the programming device to specified port of PC (USB /COM / Parallel port)
2. Power up the device
3. Execute the programming utility on the pc and ensure proper connectivity between pc and programmer. In case of error, turn off device power and try connecting it again
4. Unlock the ZIF socket by turning the lock pin
5. Insert the device to be programmed into the open socket
6. Lock ZIP socket

Out-of-Circuit Programming cont..

7. Select the device name from the list of supported devices
8. Load the hex file which is to be embedded into the device
9. Program the device by “program” option of utility program
10. Wait till the completion of programming operation
11. 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)
12. Unlock the ZIF socket & take device out of programmer

- Now the firmware is successfully embedded into the device.
- Insert the device into the board, power up the board and test it for the required functionalities

Out-of-Circuit Programming cont..

- If security is required, enable the memory protection on the utility before programming the device
- The programmer usually erases the existing content of the chip before programming the chip.
- Only EEPROM and FLASH memory chips are erasable by the programmer.
- Some old embedded systems may be built around UVEPROM chips and such chips should be erased using a separate ‘UV Chip Eraser’ before programming.

Out-of-Circuit Programming cont..

- Drawback
- **High development time**
  - Whenever the firmware is changed, the chip should be taken out of the development board for re-programming.
  - This is tedious and prone to chip damages due to frequent insertion and removal.
  - **Solution:** Better use a socket on the board side to hold the chip till the firmware modifications are over.

Out-of-Circuit Programming cont..

- **Not suitable for batch production**
  - The programmer facilitates programming of only one chip at a time and it is not suitable for batch production.
  - **Solution:** Using a 'Gang Programmer' resolves this limitation to certain extent.
  - A gang programmer is similar to an ordinary programmer except that it contains multiple ZIF sockets (say 4 to 8) and capable of programming multiple devices at a time.
  - But it is **bit expensive** compared to an ordinary programmer.
- **Very difficult to update**
  - once the product is deployed in the market in a production environment, it is very difficult to upgrade the firmware

Out-of-Circuit Programming cont..

- The out-of-system programming technique is used for firmware integration for low end embedded products which runs without an operating system.
- Out-of-circuit programming is commonly used for development of low volume products and Proof of Concept (PoC) product Development.

In System Programming

- Programming is done ‘within the system’
- Firmware is embedded into the target device without removing it from the target board
- Most flexible and easy way of firmware embedding
- Only pre-requisite : target device must have ISP support
- Needs target board, PC, ISP cable and ISP utility, no other additional hardware is required for ISP.
- Chips supporting ISP generates the necessary programming signals internally, using the chip’s supply voltage.

- The communication between the target device and ISP utility will be in a serial format
- Usually used serial protocols are Joint Test Action Group (JTAG)' or ' Serial Peripheral Interface (SPI)' or any other proprietary protocol.
- Target device (microcontroller/microprocessor) should be powered up in a special 'ISP mode' in order to perform ISP operations
- ISP mode allows the device to communicate with PC through a serial interface
- The device receives commands and data from the host, erases and reprograms code memory according to the received command.
- Once the ISP operations are completed, the device is re-configured so that it will operate normally by applying a reset or a re-power up.

In System Programming cont..

- The key player behind ISP is a factory programmed memory (ROM) called ‘Boot ROM’.
- The Boot ROM normally resides at the top end of code memory space and it varies in the order of a few Kilo Bytes
- It contains a set of Low-level Instruction APIs and these APIs allow the processor/ controller to perform the FLASH memory programming, erasing and Reading operations.
- The contents of the Boot ROM are provided by the chip manufacturer and the same is masked into every device.

In System Programming cont..

- The Boot ROM for different family or series devices is different.
- By default the Reset vector starts the code memory execution at location 0000H.
- If the ISP mode is enabled through the special ISP Power up sequence, the execution will start at the Boot ROM vector location.
- **Advantages**
- In System Programming technique is the most recommended programming technique for development work since the effort required to re-program the device in case of firmware modification is very little.
- Firmware upgrades for products supporting ISP is quite simple.

In Application Programming

- 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 the embedded application.
- Updating calibration data, look-up tables, etc., which are stored in 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.
- Thus it is possible for an end-user application to perform operations on the Flash memory.

In Application Programming cont..

- A common entry point to these API routines is provided for interfacing them to the end-user's application.
- Functions are performed by setting up specific registers as required by a specific operation and performing a call to the common entry point.
- Like any other subroutine call, after completion of the function, control will return to the end user's code.
- The Boot ROM is shadowed with the user code memory in its address range.
- This shadowing is controlled by a status bit.
- When this status bit is set, accesses to the internal code memory in this address range will be from the Boot ROM.

In Application Programming cont..

- When cleared, accesses will be from the user's code memory.
- Hence the user should set the status bit prior to calling the common entry point for IAP operations.
- Though the underlying principle in IAP is the same, the shadowing technique used for switching access between Boot ROM and code memory may be different for other family of devices).

Use of Factory Programmed Chip

- 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’.
- Once the firmware design is over and the firmware achieved operational stability, the firmware files can be sent to the chip fabricator to embed it into the code memory.
- Factory programmed chips are convenient for mass production applications and it greatly reduces the product development time.
- It is not recommended to use factory programmed chips for development purpose where the firmware undergoes frequent changes. Factory programmed ICs are bit expensive

Firmware Loading for Operating System Based Devices

- 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
- ‘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.
- The ‘Boot loader’ comes as pre-loaded or it can be loaded to the memory using the various interface and it contains necessary driver initialisation implementation for initialising the supported interfaces like UART, TCP/IP etc.

Firmware Loading for Operating System Based Devices cont..

- Bootloader implements menu options for selecting the source for OS image to load (Typical menu item examples are Load from FLASH ROM, Load from Network, Load through UART etc).
- In case of the network based loading, the bootloader broadcasts the target's presence over the network, and the host machine on which the OS image resides, can identify the target device by capturing this message.
- Once a communication link is established between the host and target machine, the OS image can be directly downloaded to the FLASH memory of the target device.

THANK YOU