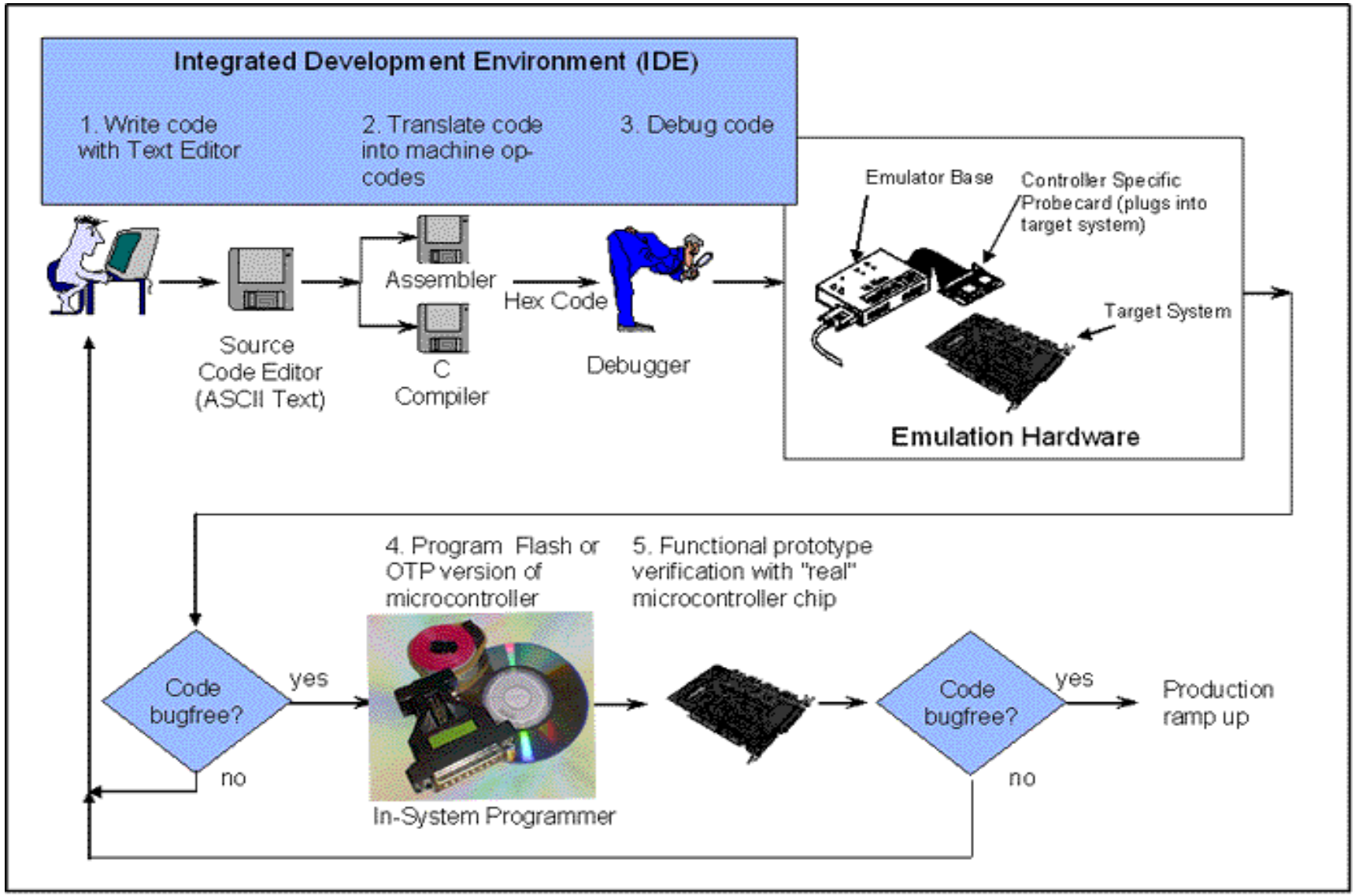
# embedded system development environment – IDE,

- An Integrated Development Environment (IDE) is software that assists programmers in developing software
- IDEs normally consist of a source code editor, a compiler, a linker/locater and usually a debugger.
- Sometimes, an IDE is devoted to one specific programming language or one (family of) specific processor or hardware
- but more often the IDEs support multiple languages, processors, etc. Some commonly used IDEs for embedded systems are the GNU compiler collection (gcc), Eclipse, Delphi,




The Microcontroller Development Cycle

- **Editor**
- A source code editor is a text editor program designed specifically for editing source code to control embedded systems. It may be a standalone application or it may be built into an integrated development environment (e.g. IDE). Source code editors may have features specifically designed to simplify and speed up input of source code, such as syntax highlighting and auto complete functionality. These features ease the development of code

- **Compiler**A compiler is a computer program that translates the source code into computer language (object code). Commonly the output has a form suitable for processing by other programs (e.g., a linker), but it may be a human-readable text file. A compiler translates source code from a high level language to a lower level language (e.g., assembly language or machine language). The most common reason for wanting to translate source code is to create a program that can be executed on a computer or on an embedded system. The compiler is called a cross compiler if the source code is compiled to run on a platform other than the one on which the cross compiler is run. For embedded systems the compiler always runs on another platform, so a cross compiler is needed.

- **Linker**
- A linker or link editor is a program that takes one or more objects generated by compilers and assembles them into a single executable program or a library that can later be linked to in itself. All of the object files resulting from compiling must be combined in a special way before the program locator  will produce an output file that contains a binary image that can be loaded into the target ROM. A commonly used linker/locater for embedded systems isld (GNU).

- **Debugger**
- A debugger is a computer program that is used to test and debug other programs
- A debugger is a piece of software running on the PC, which has to be tightly integrated with the emulator that you use to validate your code.
- A **Debugger** allows you to download your code to the emulator's memory and then control all of the functions of the emulator from a PC

# Debugging Tools

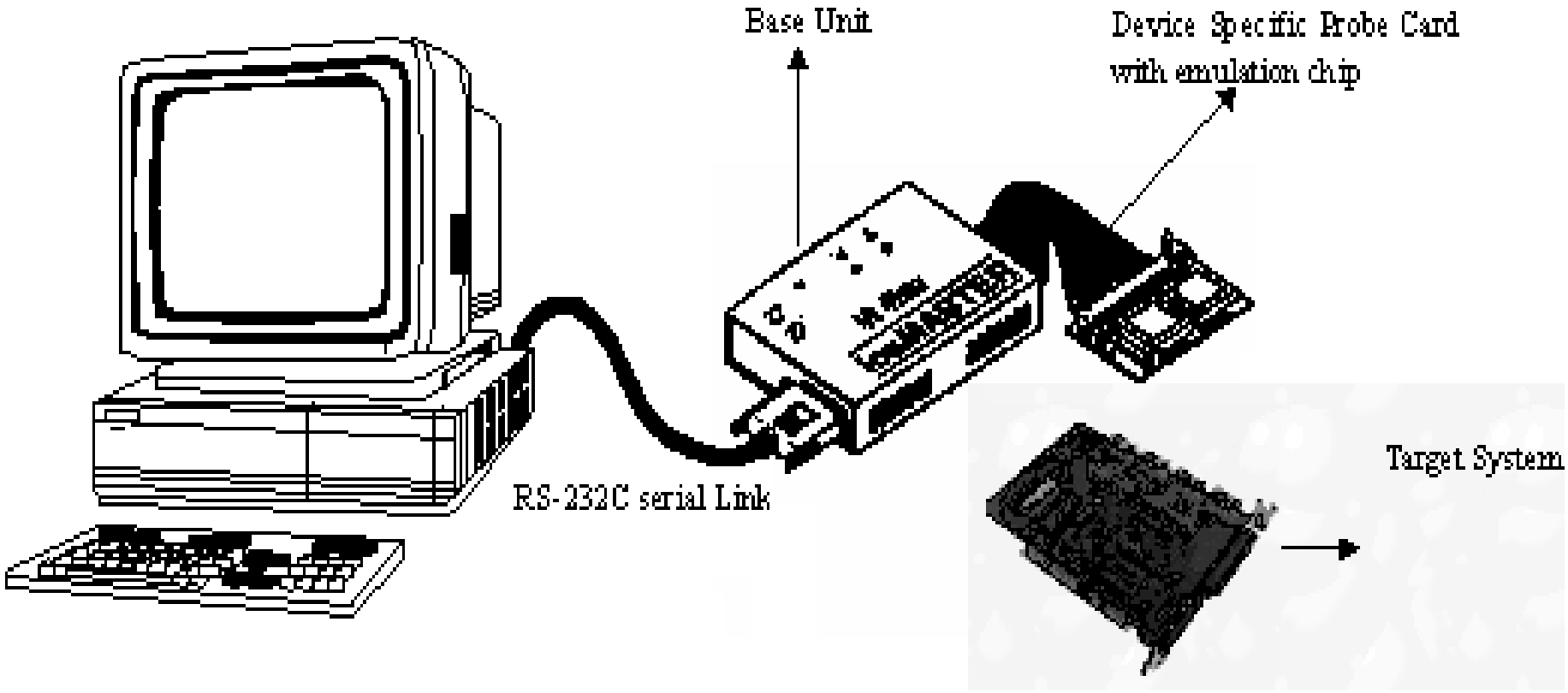
- When it comes to debugging your code and testing your application there are several different tools you can utilize that differ greatly in terms of development time spend and debugging features available. In this section we take a look at simulators, and emulators.

- **Simulators** try to model the behavior of the complete microcontroller in software. Some simulators go even a step further and include the whole system (simulation of peripherals outside of the microcontroller). No matter how fast your PC, there is no simulator on the market that can actually simulate a microcontroller's behavior in real-time. Simulating external events can become a time-consuming exercise, as you have to manually create "stimulus" files that tell the simulator what external waveforms to expect on which microcontroller pin. A simulator can also not talk to your target system, so functions that rely on external components are difficult to verify. For that reason simulators are best suited to test algorithms that run completely within the microcontroller

- An emulator is a piece of hardware that ideally behaves exactly like the real microcontroller chip with all its integrated functionality. It is the most powerful debugging tool of all. A microcontroller's functions are emulated in real-time
- Because, depending on memory technology, a microcontroller's program memory can not (ROM) or only once (OTP) be programmed, an emulator uses external static RAM as the emulated micro's program memory. Even some Flash based microcontrollers can, depending on manufacturer, only be re-programmed 100 to 1000 times, which warrants the use of external RAM memory rather than the micro's integrated Flash for emulation. RAM memory allows for code to be changed quickly and an "indefinite" number of times during the software debugging process.

- bond-out chips have additional pins that allow the emulator electronics to feed the externally stored program information to the microcontroller in place of the on-chip memory contents in real time; control the program execution flow; and access on-chip registers and data memory.

- **Base Unit and Probecard**
- Many emulators consist of a **base unit** and a **"probecard"**. The base unit is connected to a PC via the serial, parallel or USB port. It contains the majority of the emulator electronics, with the exception of the emulation chip itself. The emulation chip is a special bond-out version of the actual microcontroller and is mounted on a separate small PCB, called a probecard. This probecard connects via a ribbon cable to the base unit and has a pin adapter at the bottom, which allows the probecard to be plugged into a socket on the actual target application board in place of the actual microcontroller.


Base Unit
Device Specific Probe Card
with emulation chip
RS-232C serial Link
Target System

# Emerging trends in embedded system

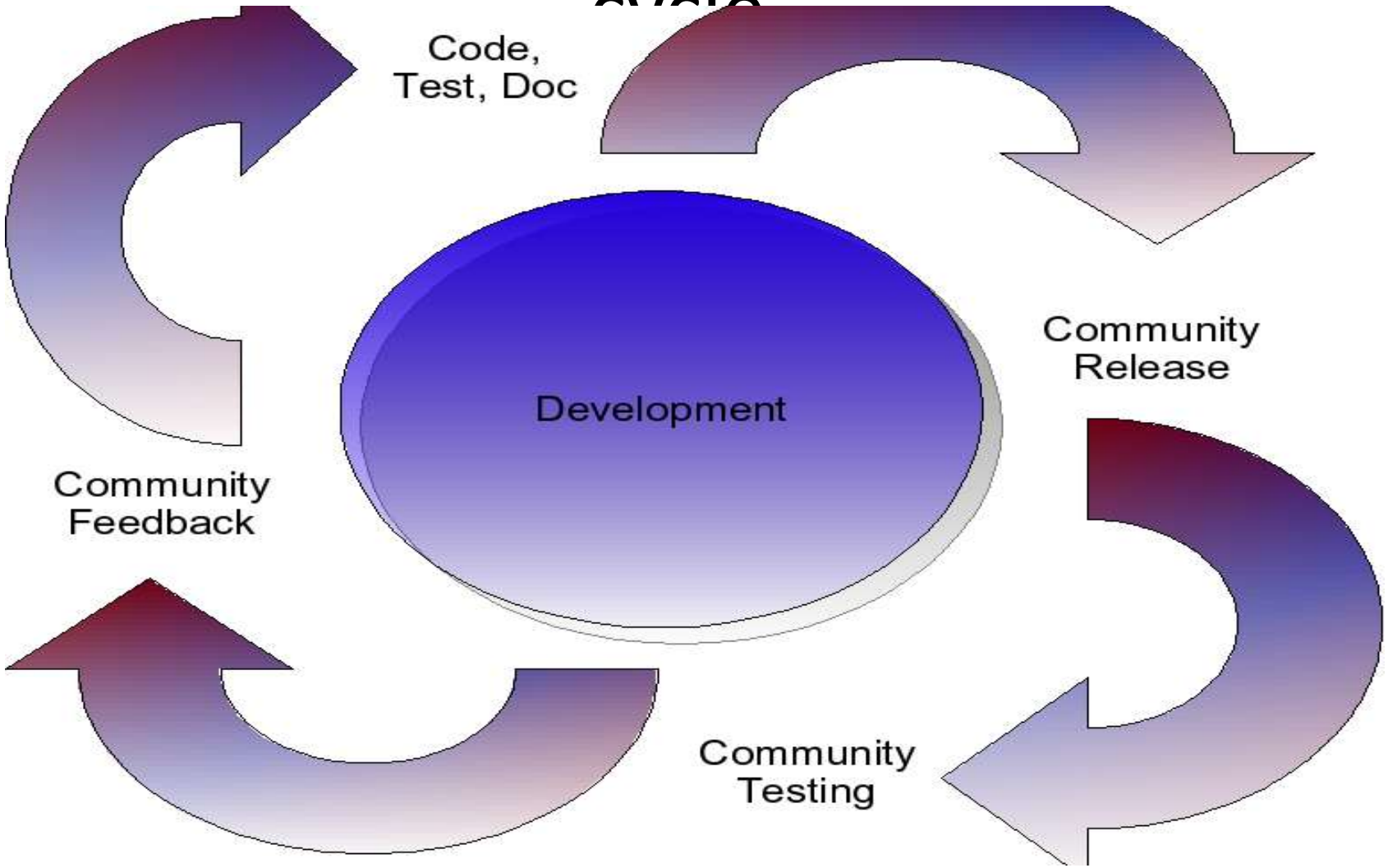
- The embedded systems industry was born with the invention of microcontrollers and since then it has evolved into various forms, from primarily being designed for machine control applications to various other new verticals with the convergence of communications.

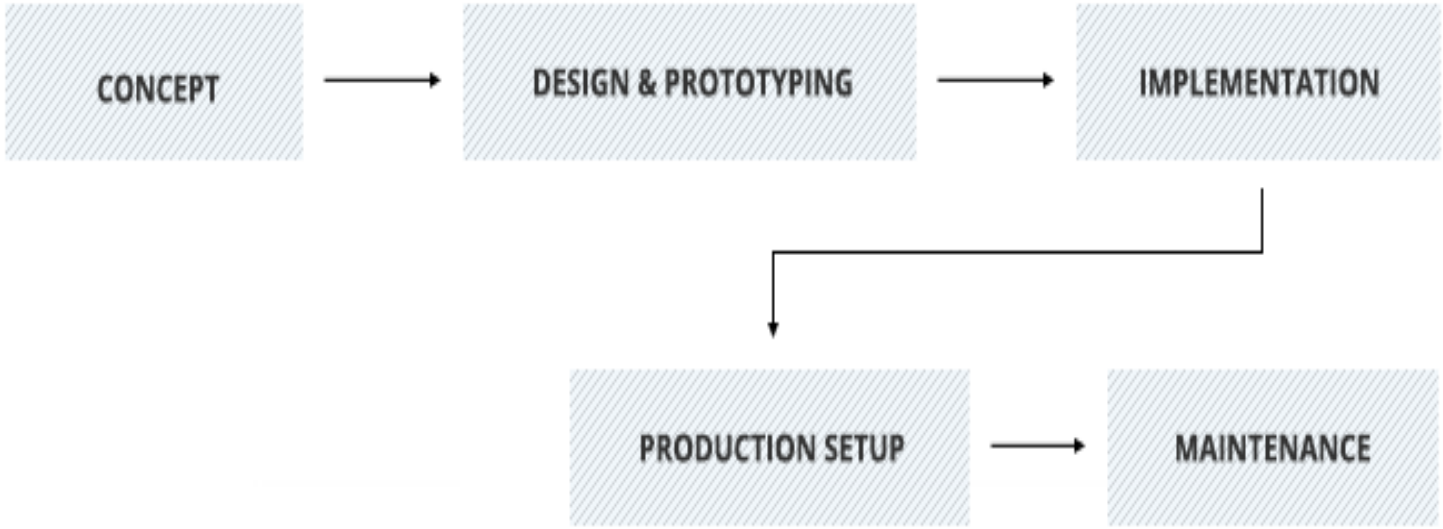
- Various classes of embedded systems such as home media systems, portable players, smart phones, embedded medical devices and sensors, automotive embedded systems have surrounded us and with continued convergence of communications and computing functions within these devices, embedded systems are transforming themselves into really complex systems,  thus creating newer opportunities and challenges to develop and market more powerful, energy efficient processors, peripherals and other accessories

- in embedded system is more than the electronics as most people perceive it. It has electronics – both digital and analog, special purpose sensors and actuators, software, mechanical items etc., and with design challenges of space, weight, cost and power consumption.

- In order to achieve key requirements, generally embedded systems are restricted to limited resources in terms of computing, memory, display size etc. With continued convergence of other technologies a lot more functionalities are being pushed into embedded devices which were once part of traditional computing platforms

# embedded product development life-cycle




CONCEPT
DESIGN & PROTOTYPING
IMPLEMENTATION
PRODUCTION SETUP
MAINTENANCE

- **Concept**
- Description of the original idea in a formal technical form (verbal requirements)
- Investigation of the existing prototypes and/or models that match the idea
- Comparative analysis of existing implementations
- Proposal of implementation and materials options

- **Design**
- **Functional Requirements**
  - Development of hardware functional specification
  - Development of software and firmware functional requirements
  - Analysis of the third-party requirements documentation
- **Architecture Design**
  - Development of the system architecture concept
  - Design of the mechanics parts/molding of the system
  - Development of hardware design documentation (including FPGA design)
  - Development of the detailed software design specification
  - Analysis of the third-party design documents

- **Hardware Modeling**
  - Schematics design
  - PCB Layout Design
  - Re-engineering and repairing
  - Samples & Prototypes Assembly
- **Prototyping**
  - Product prototyping (including all types of mechanics, hardware, software and the whole system prototyping)
  - Mechanical parts manufacturing (including molding/press forms manufacturing)
  - Hardware development
  - Software and firmware coding
  - System integration (software with hardware and mechanics)

- **Implementation**
- **Porting**
  - Porting of an existing system to a new hardware platform
  - Product certification (preparation of hardware and software for further certification process)

- **System Optimization**
  - Optimization of system performance, usability, cost, time to market and more
  - Analysis of the third-party implementation with suggested improvements
  - System benchmarking documentation
- **Testing/Debugging**
  - Creation of a sophisticated test system to verify a product on each life cycle stage
  - Development of testing documentation
  - Remote hardware test system setup to allow customer run their own applications in a sophisticated hardware/software environment
  - Quality improvement by analyzing the third-party products for existing caveats and issues, and performing the corresponding debugging

- **Transition to Manufacturing**
- Schematics design
- PCB Layout Design
- Re-engineering and repairing
- Samples & Prototypes Assembly
- **Maintenance**
- Debugging of the known problems
- Development of an ECO system by developing additional demo applications that can be used as a starting base for the system development
- System upgrades (new hardware, new software, new mechanics, etc.)