EMBEDDED SYSTEMS AND REAL TIME OPERATING SYSTEMS

What are embedded systems ?

As one may be aware that the Embedded Systems is not a new concept. The day microprocessors and micro controllers were invented, Embedded Systems took a birth. Those who were Engineering students in the late 80's are quite aware of Microprocessor Programming.

Most embedded systems are small enough to sit on the end of your thumb and are usually hidden within much larger and more complex mobile computing or electronic devices, so they often go unnoticed. But embedded systems actually represent the vast majority of semiconductor sales. According to the World Semiconductor Trade Statistics blue book, there are an estimated 5 billion embedded microprocessors in use today - a whopping 94 percent share of the world market. (By comparison, unit sales of high-profile PC processors, such as the Intel Pentium and Motorola PowerPC, check in at only 6 percent market share.)

The software for the embedded systems is called firmware. The firmware will be written in Assembly language for time or resource critical operations or using higher level languages like C or embedded C. The software will be simulated using microcode simulators for the target processor. Since they are supposed to perform only specific tasks, these programs are stored in Read Only Memories (ROMs) . Moreover they may need no or minimal inputs from the user, hence the user interface like monitor, mouse and large keyboard etc. may be absent.

Embedded systems are also known as real time systems since they respond to an input or event and produce the result within a guaranteed time period. This time period can be few microseconds to days or months. Real time systems are further classified as hard real time systems and soft real time systems, based on the strictness to the time period. A hard real time system should complete the specified task within the stipulated time frame. A failure to do so is treated as the failure of the system. So hard real time systems are deterministic. A soft real time system is not very strict. Not completing a task within the time frame is pardonable, provided it will not affect the overall system performance.

A real time system may be doing some simple task to complex tasks with many decision loops and interrupts. Real Time Operating Systems (RTOS) are used to schedule tasks in complex systems. In simple terms an RTOS can be defined as a context switcher. An RTOS helps to schedule and execute tasks based on priority in a predictable manner. So RTOS has to be multithreaded and pre-emptible. It also should have predictable thread synchronization mechanisms. Usage of RTOS will enable you to break the complex system into smaller tasks without worrying about the inter task timing problems. Functions of RTOS and it details are beyond the scope of this article.

Getting smarter

Thanks to the Internet and the market explosion of new technologies, embedded systems are becoming smarter and more network-friendly every day. So, whereas today's embedded systems may perform such mundane tasks as synchronizing the clock in a microwave oven, tomorrow's systems might download recipes via the Internet or alert repair companies of product malfunction.

Clearly, the growing use of embedded systems on the Internet also represents a lucrative proving ground for vendors looking to extend their reach of desktop operating systems and development tools. Because there is not yet an established market leader in embedded operating systems, companies such as Microsoft (with Windows CE), Sun Microsystems (with Java 2 Micro Edition), and Red Hat Software (with Linux for Embedded Developers) all face an excellent opportunity to branch their operating systems into this promising market.

The challenge these companies face, of course, is to deliver quality products while ensuring that their operating systems can interoperate with similar devices in the market. No obvious leader has yet taken the lead in this field, and the next few years should offer some fascinating developments from competing vendors, each of which is struggling to achieve an edge.

Thinking small

Most experts predict that the current explosion of activity in the embedded technology sector is only going to get bigger. A recent report by market research firm IDC predicts that by 2002, Internet appliances - primarily consisting of embedded systems - could rival the unit volume numbers posted by all PC vendors combined.

Dataquest echoes this sentiment, predicting that by 2003, 400 million Internet appliances will be in use, and that by 2010, all home PCs will be replaced by embedded system-based devices. In this scenario, most home offices would probably use one or more separate Internet appliances, which will either be industry-specialized or will converge many technologies (phone, fax, Internet, and TV) into one device.

Keys to growth

The first factor is the ongoing emergence of standards-based operating systems for embedded devices. Current usage trends show the market to be fragmented with developers employing a combination of commercial, free, and proprietary operating systems for development (see related pie chart). With this in mind, many major operating system vendors are repurposing their wares for the embedded market place.

Embedded system-based devices

Not to be outdone is Sun Microsystems, which, in conjunction with a variety of industry partners, has developed the Java 2 Micro Edition (J2ME) standard, a development language that combines a small-footprint JVM (Java virtual machine) with a set of APIs for use in a range of embedded applications.

Of course, Linux vendors are also thinking small. For example, Red Hat Software's Red Hat for Embedded Developers provides a toolkit for developers looking to create open-source applications for embedded-system devices that contain as little as 32KB of memory.

As embedded operating system choices continue to percolate to the top, the real winners will be the developers who are able to decrease the time to market for business applications. Ultimately, this will allow businesses to innovate more quickly, delivering new services to customers at a much more rapid rate.

To complement the additional embedded operating system choices, integrated software developers have also made strides to streamline application development. Only a few short years ago, embedded software development environments paled in comparison to their native desktop PC counterparts. But the gap has closed considerably.

Development scenario of various embedded systems

If we recall that in 1980's, when microcomputers started coming into picture, the processor used to be based on 8 bit or 4 bit. The programs, which were made to run on Microprocessor, were also referred embedded programs. Over a period of time, human requirement started increasing on the similar lines. Systems which scientists and engineers started conceiving require more powerful microprocessors and micro controllers. This lead to increase in computational horsepower of processors. 16-bit microprocessors came into picture in late 80's and subsequently in 90's, 32 bit microprocessors and currently 64-bit microprocessors are put into action. Apart from this, dedicated processors also came into picture, which are used for very high computational speed.

The arrival of operating system concepts

The increase in the internal bus architecture gave rise to new set of complexities. Namely, how to manage this processor power for specific devices under considerations. This has lead to the development of Operating System concepts. Although Operating Systems were developed much early when computers started pouring in, Operating System popularity was restricted only till the development platform like PC, Mini and Mainframe computers. Operating System used to play a major role on these systems, as these were made as General Purpose Platform where one can do programming for various purposes.

Non-suitability of general purpose operating systems for embedded systems

Earlier OS concepts were never a choice of tools for Embedded System Programmers. The reason being Embedded System Program always used to work in constraints in terms of memory, processing speed and processor utilization. Embedded System Programmer was more concerned about this because, such programmers were designing a product based on Microprocessor or Micro controller for specific application. However, each new upcoming application was becoming more vital, intrinsic, dynamic and constrained. These became a forcing condition for Embedded System Programmer to look in for OS type of concepts suitable for Embedded Application Design. The common Operating Systems failed to satisfy the need of Embedded System Programmer's requirements. To be short, an Embedded System designer expects two important issues to be satisfied from his prospective in a given Operating System.

a. Scalability
b. Determinacy

That is, Operating Systems should be such that, it can be scaled as per the requirement of an application and the second is that throughput for an application should be deterministic. If these two issues are looked upon, Embedded System Programmer would not have found them in the conventional OS. These issues were non-substantial for general purpose Operating Systems, as they were catering to the need of PC, Mini, and Mainframe computational machine market.

Arrival of Real time operating systems (RTOS)

As can be perceived from the previous discussions, need for a time critical and mission critical RTOS was felt badly to assist in development of embedded systems. Fortunately, by that time Real Time Operating Systems had already been was pressed into action in Aerospace and Defense Systems. Here it was being used heavily, since a long time for Aircraft Control, Missile Launch Control, Space Craft Control etc.

The Mission Critical Systems, which are Time dependent and constrained.

In System theory terminology, these systems are hard real time systems, which are time critical. Interestingly, time critical systems will naturally satisfy the determinacy criteria, which is one of the prerequisite conditions for OS to become useful for Embedded System Designers. RTOS is catering to the need of timing and it naturally becomes scalable. Hence the present generation of Embedded System Programmers use the 'RTOS' as an OS to design Embedded Applications.

Generation-wise classification of embedded system scenario

The Embedded System Designers/Programmers in early 80's used to work on 8 bit microprocessor like 8085 / Z80 etc and 4 bit micro controller. The targets were commonly applications like microprocessor based stepper motor control, microprocessor based relays, LED panel display System, Digital Telephone Keypad handling etc. The programming used to be in assembly, as 8-bit processors were having simple instruction set. So Embedded System

programmer never thought of OS as a requirement.

2nd generation Embedded System programmer started using 16-bit microprocessor and 8 bit/16 micro controllers. The instruction sets were a bit more difficult than the previous ones. The applications that were targeted were Motor controls, UPS controls, and Data acquisition systems, SCADAS, Low-end switches/routers for local area network, Modems etc . Very few of the applications may be those involving use of some network stacks required the intervention of Operating Systems. Thus the use of RTOS was started as a custom in some of the applications involving networking apart from the developments in defense and aeronautical sectors as discussed earlier.

3rd generation Embedded System Programmers started using 32 bit microprocessors, 16 bit micro controllers and 16 bit special processors called DSP processors. The instruction set became very complex and pipelining became active component of processors. The Processors were now classified in accordance with the target applications that they were being used for. RISC/CISC/SIMD/MIMD architecture became popular in processors. Instruction set became VLIW instruction set. Design of embedded applications now invariably required RTOS. Market became flooded with different types of processors. To quote some of them
On the microprocessor side, X86 - Pentium, PPC-MBX-86X, Mot - 68 XXX etc started gaining ground.
On the micro controller side, 80196, 8051 SM-9 etc became popular.

On the DSP processor side, ADSP - 21 XX, ADSP2106X, TMS320CXX etc started picking up in the market.

The market also got flooded with embedded applications especially catering to the needs of networking due to the Internet applications gaining ground. To name a few of them.

- Switches, Routers, Hubs, Gateways etc
- Process Controller and Robotics
- Medical devices and instructions
- Palm Pilots
- Web enabled devices
- Digital Telephone and exchanges, Packet switched networks.
- Switches for hybrid networking
- HDTV
- Video Conferencing etc.

4th generation Embedded System Programmers started using different cores of microprocessors. The cores became popular because some application demands combination of microprocessor along with DSP and controllers, especially in the area of transmitting digital data over different type of media like wireless, wireline. Thus cores are becoming high in demand, however it has added a complexity in terms of programming and designing high-end applications. That is why; Embedded market is flooded with cores of different types of microprocessors/DSP/micro controllers. Applications were now developed such that they were based on their requirement.

Hence came the concept of various combinations of cores into a chip which were also programmed beforehand for specific applications. So this generation Embedded System Programmers need skill not only from RTOS angle, but also from other streams like that of DSP and VLSI/ASIC/VHDL designs, thus making the task more difficult for Embedded System Programmers. Interestingly, bigger challenges are faced in view of imbibing skills in a single individual.

The continual rise of Internet usage by consumers and businesses has also lit a fire under the growth of embedded technology. Because these systems can interconnect with one another via a myriad of wired and wireless networking options, they are supremely viable in this marketplace. For example, high-level protocol support for TCP/IP and SMTP is now commonplace among the embedded systems used in most handheld devices.

Comprehensive, seamless, and worldwide connected embedded systems may still be a pipe dream today, but they are quickly becoming more accessible and controllable thanks to LANs, WANs, and the Internet. Many businesses are already using embedded technology to innovate with voice, video, and data traffic, hoping to set the stage for a competitive advantage in the future.

With so much attention being paid to making the Internet pervasive, smart, and easy to use, the sky is the limit for embedded systems. Beyond their sheer numbers and dwindling price points, embedded systems are becoming smarter and more controllable - a potent combination that benefits both businesses and consumers alike.

In view of incorporating these skills in professionals/engineers (fresh), we at Accelerated Systems have successfully started an unique program which imparts a sound grounding in technology in an extremely good learning environment.

An Engineering graduate after completing the training at Accelerated Systems comes up as a thoroughbred professional ready to face this fast changing industry.

The types of courses offered vary in their content and nature of training depending upon the different technologies. Various courses offered are

1. RTOS and Embedded Systems
2. Digital Signal processing
3. VLSI (with emphasis on ASIC Design)

These courses can be undertaken individually or on a combination of two courses depending on the previous background of an individual and his future goals.

A course in RTOS gives very intensive training on conceptual design of RTOS and then step by step learning of its important features. Here candidate is exposed to very intricate details of OS design etc.