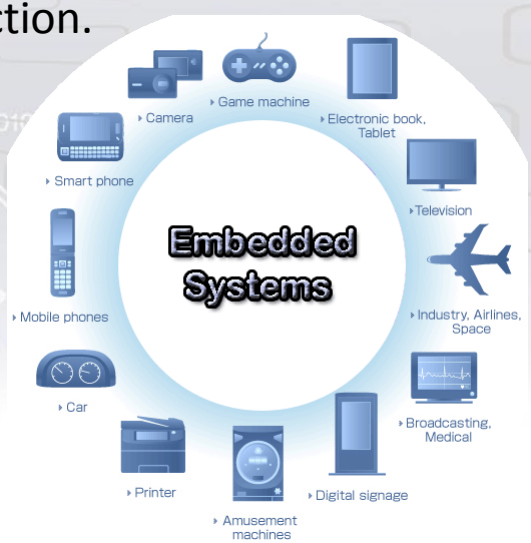
Embedded system

https://en.wikipedia.org/wiki/Embedded_system

Embedded system

- An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints.
- It is embedded as part of a complete device often including hardware and mechanical parts.
- Embedded systems control many devices in common use today.
- Ninety-eight percent of all microprocessors are manufactured as components of embedded systems.

- An Embedded System is simply a combination of computer hardware and software, either fixed in operability or programmable, which is designed to perform a specific function.



Embedded system

Properties of typically embedded computers when compared with general-purpose counterparts are:

- low power consumption,
- small size,
- rugged operating ranges,
- low per-unit cost.

- This comes at the price of limited processing resources, which make them significantly more difficult to program and to interact with.
- However, by building intelligence mechanisms on top of the hardware, taking advantage of possible existing sensors and the existence of a network of embedded units, one can both optimally manage available resources at the unit and network levels as well as provide augmented functions, well beyond those available.
- For example, intelligent techniques can be designed to manage power consumption of embedded systems.

Embedded system

- Modern embedded systems are often based on microcontrollers (i.e. CPU's with integrated memory or peripheral interfaces)
- Ordinary microprocessors (using external chips for memory and peripheral interface circuits) are also common, especially in more-complex systems. In either case, the processor(s) used may be types ranging from general purpose to those specialised in certain class of computations, or even custom designed for the application at hand.
- A common standard class of dedicated processors is the digital signal processor (DSP).
- Due to the falling costs very popular are also ready made computer boards intended for small, low-volume embedded and ruggedized systems, mostly x86-based.
- Their another advantage is standarisation and worldwide accepted platform.

Embedded system

- Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.
- Embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, and largely complex systems like hybrid vehicles, Magnetic resonance imaging MRI, and avionics. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

Applications

- Embedded systems are commonly found in consumer, cooking, industrial, automotive, medical, commercial and military applications.

Applications

Telecommunications systems employ numerous embedded systems

- telephone switches for the network
- cell phones at the end user
- computer networking uses dedicated routers and network bridges to route data.

Applications

Consumer electronics

- MP3 players,
- mobile phones,
- videogame consoles,
- digital cameras,
- GPS receivers,
- Printers

Household appliances include embedded systems to provide flexibility, efficiency and features in:

- microwave ovens,
- washing machines
- dishwashers

Advanced HVAC (Heating, ventilation and air conditioning) systems use

- networked thermostats to more accurately and efficiently control temperature that can change by time of day and season.

Home automation uses wired- and wireless-networking that can be used to control

- lights,
- climate,
- security,
- audio/visual,
- surveillance, etc.,

all of which use embedded devices for sensing and controlling.

Applications

Transportation systems from flight to automobiles increasingly use embedded systems.
Airplanes:

- advanced avionics
- inertial guidance systems
- GPS receivers

(have considerable safety requirements).

Various electric motors — brushless DC motors, induction motors and DC motors — use electric/electronic motor controllers.

Automobiles, electric vehicles, and hybrid vehicles increasingly use embedded systems to maximize efficiency and reduce pollution.

Automotive safety systems:

- anti-lock braking system (ABS),
- Electronic Stability Control (ESC/ESP),
- traction control (TCS) ,
- automatic four-wheel drive.

Applications

Medical equipment uses embedded systems for :

- vital signs monitoring,
- electronic stethoscopes for amplifying sounds,
- various medical imaging (PET, SPECT, CT, and MRI) for non-invasive internal inspections.

Embedded systems within medical equipment are often powered by industrial computers.

Applications

A new class of miniature wireless devices called motes are networked wireless sensors. Wireless sensor networking, WSN, makes use of miniaturization made possible by advanced IC design to couple full wireless subsystems to sophisticated sensors, enabling people and companies to measure a myriad of things in the physical world and act on this information through IT monitoring and control systems.

These motes are completely self-contained, and will typically run off a battery source for years before the batteries need to be changed or charged.

Embedded Wi-Fi modules provide a simple means of wirelessly enabling any device which communicates via a serial port.

Applications

Embedded systems are used in transportation, fire safety, safety and security, medical applications and life critical systems, as these systems can be isolated from hacking and thus, be more reliable.

For fire safety, the systems can be designed to have greater ability to handle higher temperatures and continue to operate. In dealing with security, the embedded systems can be self-sufficient and be able to deal with cut electrical and communication systems.

Characteristics

Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks.

- Some also have real-time performance constraints that must be met, for reasons such as safety and usability.
- Others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs.

Characteristics

- Embedded systems are not always standalone devices.
- Many embedded systems consist of small parts within a larger device that serves a more general purpose.
- The program instructions written for embedded systems are referred to as firmware, and are stored in read-only memory or flash memory chips.
- They run with limited computer hardware resources: little memory, small or non-existent keyboard or screen.

User interface

- Embedded systems range from no user interface at all, in systems dedicated only to one task, to complex graphical user interfaces that resemble modern computer desktop operating systems.
- Simple embedded devices use buttons, LEDs, graphic or character LCDs (HD44780 LCD for example) with a simple menu system.

User interface

- More sophisticated devices which use a graphical screen with touch sensing or screen-edge buttons provide flexibility while minimizing space used.
- The meaning of the buttons can change with the screen, and selection involves the natural behavior of pointing at what is desired.
- Handheld systems often have a screen with a "joystick button" for a pointing device.

User interface

- Some systems provide user interface remotely with the help of a serial (e.g. RS-232, USB, I^2C, etc.) or network (e.g. Ethernet) connection.

This approach gives several advantages:

- extends the capabilities of embedded system,
- avoids the cost of a display,
- simplifies BSP (board support package)
- allows one to build a rich user interface on the PC.

Webbased interface

- A good example of this is the combination of an embedded web server running on an embedded device (such as an IP camera) or a network router.

The user interface is displayed in a web browser on a PC connected to the device, therefore needing no software to be installed.

Processors in embedded systems

Embedded processors can be broken into two broad categories.

- Ordinary microprocessors (µP) use separate integrated circuits for memory and peripherals.
- Microcontrollers (µC) have on-chip peripherals, thus reducing power consumption, size and cost.

- In contrast to the personal computer market, many different basic CPU architectures are used, since software is custom-developed for an application and is not a commodity product installed by the end user.
- Both Von Neumann as well as various degrees of Harvard architectures are used. RISC as well as non-RISC processors are found.

Ready made computer boards

- Another branch of embeded systems are ready made computer boards intended for small, low-volume embedded and ruggedized systems, mostly x86-based ora ARM , but not only.
- These are often physically small compared to a standard PC, although still quite large compared to most simple (8/16-bit) embedded systems.
- They often use DOS, Linux, NetBSD, or an embedded real-time operating system such as MicroC/OS-II, QNX or VxWorks.

Ready made computer boards

- In certain applications, where small size or power efficiency are not primary concerns, the components used may be compatible with those used in general purpose x86 personal computers.
- Boards such as the VIA EPIA (VIA Embedded Platform Innovative Architecture)range help to bridge the gap by being PC-compatible but highly integrated, physically smaller or have other attributes making them attractive to embedded engineers.
- The advantage of this approach is that low-cost commodity components may be used along with the same software development tools used for general software development.
- Systems built in this way are still regarded as embedded since they are integrated into larger devices and fulfill a single role.

Ready made computer boards

- However, most ready-made embedded systems boards are not PC-centered and do not use the ISA or PCI buses.
- When a system-on-a-chip processor is involved, there may be little benefit to having a standarized bus connecting discrete components, and the environment for both hardware and software tools may be very different.
- One common design style uses a small system module, perhaps the size of a business card, holding high density BGA chips such as an ARM-based system-on-a-chip (SoC) processor and peripherals, external flash memory for storage, and DRAM for runtime memory.

Ready made computer boards

- The module vendor will usually provide boot software and make sure there is a selection of operating systems, usually including Linux and some real time choices.
- These modules can be manufactured in high volume, by organizations familiar with their specialized testing issues, and combined with much lower volume custom mainboards with application-specific external peripherals.
- Implementation of embedded systems have advanced, embedded systems can easily be implemented with already made boards which are based on worldwide accepted platform. These platforms include, but are not limited to, Arduino, Intel Galileo and Raspberry Pi.

ASIC and FPGA solutions

- A common array of n configuration for very-high-volume embedded systems is the system on a chip (SoC) which contains a complete system consisting of multiple processors, multipliers, caches and interfaces on a single chip.
- SoCs can be implemented as an application-specific integrated circuit (ASIC) or using a field-programmable gate array (FPGA).

Peripherals

Embedded systems talk with the outside world via peripherals, such as:

- Serial Communication Interfaces (SCI): RS-232, RS-422, RS-485, UART etc.
- Synchronous Serial Communication Interface: I2C, SPI, SSC and ESSI (Enhanced Synchronous Serial Interface)
- Universal Serial Bus (USB)
- Multi Media Cards (SD cards, Compact Flash, etc.)
- Networks: Ethernet, LonWorks, etc.
- Fieldbuses: CAN-Bus, LIN-Bus, PROFIBUS, etc.
- Timers: PLL(s), Capture/Compare and Time Processing Units
- Discrete IO: aka General Purpose Input/Output (GPIO)
- Analog to Digital/Digital to Analog (ADC/DAC)
- Debugging: JTAG, ISP, ICSP, BDM Port, BITP, and DB9 ports.

Reliability

- Embedded systems often reside in machines that are expected to run continuously for years without errors, and in some cases recover by themselves if an error occurs.
- Therefore, the software is usually developed and tested more carefully than that for personal computers, and unreliable mechanical moving parts such as disk drives, switches or buttons are avoided.

Specific reliability issues may include:

- The system cannot safely be shut down for repair, or it is too inaccessible to repair. Examples include space systems, undersea cables, navigational beacons, bore-hole systems, and automobiles.
- The system must be kept running for safety reasons. "Limp modes" are less tolerable. Often backups are selected by an operator. Examples include aircraft navigation, reactor control systems, safety-critical chemical factory controls, train signals.
- The system will lose large amounts of money when shut down: telephone switches, factory controls, bridge and elevator controls, funds transfer and market making, automated sales and service.

Reliability

A variety of techniques are used, sometimes in combination, to recover from errors—both software bugs such as memory leaks, and also soft errors in the hardware:

- watchdog timer that resets the computer unless the software periodically notifies the watchdog subsystems with redundant spares that can be switched over to software "limp modes" that provide partial function
- Designing with a Trusted Computing Base (TCB) architecture ensures a highly secure & reliable system environment
- A hypervisor designed for embedded systems, is able to provide secure encapsulation for any subsystem component, so that a compromised software component cannot interfere with other subsystems, or privileged-level system software. This encapsulation keeps faults from propagating from one subsystem to another, improving reliability. This may also allow a subsystem to be automatically shut down and restarted on fault detection.
- Immunity Aware Programming - programming techniques which improve the tolerance of transient errors in the program counter or other modules of a program that would otherwise lead to failure. Transient errors are typically caused by single event upsets, insufficient power, or by strong electromagnetic signals transmitted by some other "source" device.

High vs. low volum

For high volume systems such as portable music players or mobile phones, minimizing cost is usually the primary design consideration. Engineers typically select hardware that is just “good enough” to implement the necessary functions.

- For low-volume or prototype embedded systems, general purpose computers may be adapted by limiting the programs or by replacing the operating system with a real-time operating system.

Embedded software architectures

There are several different types of software architecture in common use:

- Simple control loop
- Interrupt-controlled system
- Cooperative multitasking
- Preemptive multitasking or multi-threading
- Microkernels and exokernels
- Monolithic kernels
- Additional software components

Simple control loop

In this design, the software simply has a loop.

The loop calls subroutines, each of which manages a part of the hardware or software. Hence it is called a simple control loop or control loop.

Interrupt-controlled system

Some embedded systems are predominantly controlled by interrupts. This means that tasks performed by the system are triggered by different kinds of events; an interrupt could be generated, for example, by a timer in a predefined frequency, or by a serial port controller receiving a byte.

These kinds of systems are used if event handlers need low latency, and the event handlers are short and simple. Usually, these kinds of systems run a simple task in a main loop also, but this task is not very sensitive to unexpected delays.

Sometimes the interrupt handler will add longer tasks to a queue structure. Later, after the interrupt handler has finished, these tasks are executed by the main loop. This method brings the system close to a multitasking kernel with discrete processes.

Cooperative multitasking

A nonpreemptive multitasking system is very similar to the simple control loop scheme, except that the loop is hidden in an API. The programmer defines a series of tasks, and each task gets its own environment to “run” in. When a task is idle, it calls an idle routine, usually called “pause”, “wait”, “yield”, “nop” (stands for no operation), etc.

The advantages and disadvantages are similar to that of the control loop, except that adding new software is easier, by simply writing a new task, or adding to the queue.

Preemptive multitasking or multi-threading

In this type of system, a low-level piece of code switches between tasks or threads based on a timer (connected to an interrupt). This is the level at which the system is generally considered to have an "operating system" kernel. Depending on how much functionality is required, it introduces more or less of the complexities of managing multiple tasks running conceptually in parallel.

As any code can potentially damage the data of another task (except in larger systems using an MMU) programs must be carefully designed and tested, and access to shared data must be controlled by some synchronization strategy, such as message queues, semaphores or a non-blocking synchronization scheme.

Preemptive multitasking or multi-threading

Because of these complexities, it is common for organizations to use a real-time operating system (RTOS), allowing the application programmers to concentrate on device functionality rather than operating system services, at least for large systems; smaller systems often cannot afford the overhead associated with a generic real time system, due to limitations regarding memory size, performance, or battery life.

The choice that an RTOS is required brings in its own issues, however, as the selection must be done prior to starting to the application development process.

This timing forces developers to choose the embedded operating system for their device based upon current requirements and so restricts future options to a large extent.

Preemptive multitasking or multi-threading

The restriction of future options becomes more of an issue as product life decreases. Additionally the level of complexity is continuously growing as devices are required to manage variables such as serial, USB, TCP/IP, Bluetooth, Wireless LAN, trunk radio, multiple channels, data and voice, enhanced graphics, multiple states, multiple threads, numerous wait states and so on. These trends are leading to the uptake of embedded middleware in addition to a real-time operating system.

Microkernels and exokernels

A microkernel is a logical step up from a real-time OS. The usual arrangement is that the operating system kernel allocates memory and switches the CPU to different threads of execution. User mode processes implement major functions such as file systems, network interfaces, etc.

In general, microkernels succeed when the task switching and intertask communication is fast and fail when they are slow.

Exokernels communicate efficiently by normal subroutine calls. The hardware and all the software in the system are available to and extensible by application programmers.

Monolithic kernels

In this case, a relatively large kernel with sophisticated capabilities is adapted to suit an embedded environment. This gives programmers an environment similar to a desktop operating system like Linux or Microsoft Windows, and is therefore very productive for development.

It requires considerably more hardware resources, is often more expensive, and, because of the complexity of these kernels, can be less predictable and reliable.

Common examples of embedded monolithic kernels are embedded Linux and Windows CE.

Despite the increased cost in hardware, this type of embedded system is increasing in popularity, especially on the more powerful embedded devices such as wireless routers and GPS navigation systems.

Monolithic kernels – reasons of popularity

- Ports to common embedded chip sets are available.
- They permit re-use of publicly available code for device drivers, web servers, firewalls, and other code.
- Development systems can start out with broad feature-sets, and then the distribution can be configured to exclude unneeded functionality, and save the expense of the memory that it would consume.
- Many engineers believe that running application code in user mode is more reliable and easier to debug, thus making the development process easier and the code more portable.
- Features requiring faster response than can be guaranteed can often be placed in hardware.

Additional software components

In addition to the core operating system, many embedded systems have additional upper-layer software components.

These components consist of networking protocol stacks like CAN, TCP/IP, FTP, HTTP, and HTTPS, and also included storage capabilities like FAT and flash memory management systems.

If the embedded device has audio and video capabilities, then the appropriate drivers and codecs will be present in the system.

In the case of the monolithic kernels, many of these software layers are included.

In the RTOS category, the availability of the additional software components depends upon the commercial offering