- One of the biggest differences between embedded system platforms and conventional computing platforms is the lack of available resources. When we talk about laptops and servers, they come with memories with a storage capacity of gigabytes. Microcontroller Resources are measured in kilobytes.

8.2 TYPES OF MEMORY:

- Memory is the most essential element of a computer system because without it the computer cannot perform simple tasks. Computer memory comes in two basic types -

8.2.1 Primary Memory

- Primary / Main memory only contains data and instructions that are currently processed by the computer. Its capacity is limited and data will be lost if the power is turned off. It usually consists of semiconductor components.
- These memories are not as fast as registers. Processed data and commands are in main memory. It is divided into two subcategories: RAM (mostly volatile memory) and ROM (mostly non-volatile memory).

Characteristics of Main Memory

- These are semiconductor memories.
- It is known as the main memory.
- Usually volatile memory.
- Data is lost in case power is switched off.
- It is the working memory of the computer.
- Faster than secondary memories.
- A computer cannot run without the primary memory.

Types of Primary Memory

8.2.1.1 Random Access Memory (RAM)

- RAM (Random Access Memory) is the internal memory of the CPU, used to store data, programs, and program results. It is a read/write memory used to store data until the machine works. Once the device is turned off, the data will be deleted. The access time in the RAM does not depend on the address, that is, each storage location in the memory can be accessed as easily as other locations and requires the same time.
- The data in RAM can be accessed at will, but it is very expensive. RAM is volatile, i.e., data stored in it is lost when we switch off the computer or if there is a power failure. Hence, a backup Uninterruptible Power System and is often used with computers.

- RAM is small, both in terms of its physical size and in the amount of data it can hold. RAM is further classified into two types SRAM (Static Random-access Memory) and DRAM (Dynamic Random-Access Memory).

8.2.1.1 a) Static RAM (SRAM)

- The word static indicates that memory retains its contents as long as it is energized. However, data is lost when the power is turned off due to instability. The SRAM uses an array of 6 transistors with no capacitors. The transistors don't need a power supply to prevent leakage, so SRAM doesn't need to be updated regularly.
- There is extra space in the die, so SRAM uses more chips than DRAM for the same amount of memory, which increases production costs. Thus, SRAM is used as a cache and has very fast access.

Characteristic

- Long life
- No need to refresh
- Faster
- Used as cache memory
- Large size
- Expensive
- High power consumption

8.2.1.1 b) Dynamic RAM (DRAM)

- DRAM, unlike SRAM, needs to be constantly updated to preserve data. This is done by placing memory in the update circuit, which overwrites the data several hundred times per second. Most system memories use DRAM because it is cheap and small.

All DRAMs are made up of memory cells, which are made up of one capacitor and one transistor.

Characteristics

- Short data lifetime
- Needs to be refreshed continuously
- Slower as compared to SRAM
- Used as RAM
- Smaller in size
- Less expensive
- Less power consumption

8.2.1.2 Read Only Memory (ROM)

ROM stands for Read Only Memory. A memory from which we can only read, but not write. This type of memory is non-volatile. Information is permanently stored in such storage devices during production. The ROM contains instructions for starting the computer. This operation is called bootstrapping. ROM chips are used not only in a computer, but also in other electronic systems or where programming does not require changes. Examples: washing machines and microwaves, calculators and peripherals.

Types of Read Only Memory (ROM)

8.2.1.2 a) PROM (Programmable Read Only Memory) - can be programmed by the user. Once programmed, the data and instructions contained therein cannot be changed.

8.2.1.2 b) EPROM (Erasable Programmable Read Only Memory) - Can be reprogrammed. To erase data from it, expose it to ultraviolet light. To reprogram it, delete all previous data.

8.2.1.2 c) EEPROM (Electrically Erasable Programmable Read Only Memory) - data can be erased by applying an electric field, without the need for UV light. We can only erase parts of the chip. d) Mask ROM (MROM): This is a type of read-only memory (ROM) whose contents are programmed by the chip manufacturer (not the user). The earliest ROMs were hardware devices that contained a pre-programmed set of data or instructions. These types of ROMs are known as masked ROMs and are inexpensive.

Advantages of ROM

- Non-volatile in nature
- Cannot be accidentally changed
- Cheaper than RAMs
- Easy to test
- More reliable than RAMs
- Static and do not require refreshing
- Contents are always known and can be verified

8.2.2 Secondary Memory

This type of storage is also called external storage or non-volatile storage. It is slower than the main memory. These are used to store data/information permanently. The CPU does not directly access this memory, but through input-output routines. The contents of the auxiliary storage are first transferred to the main storage, and then the CPU can access them. For example, floppy disks, CDROM, DVD, etc.

Characteristics of Secondary Memory

- These are magnetic and optical memories.
- It is known as the backup memory.
- It is a non-volatile memory.

- Data is permanently stored even if power is switched off.
- It is used for storage of data in a computer.
- Computers may run without the secondary memory.
- Slower than primary memories.

8.3 MAKING THE MOST OF YOUR RAM

- When you only have a few kilobytes or tens of kilobytes of RAM, it causes your device to malfunction or crash and it is easier to make the best compromise between maximum RAM utilization and reliability if memory usage is deterministic, that is, if you know the maximum amount of memory that will be used. The way to achieve this result is not to allocate memory dynamically, that is, while the program is running.
- In the deterministic model, instead of reserving space for the entire page, you reserve space for important information to be extracted, as well as a memory buffer that can be used as a work area when downloading and processing files on the page.
- is not a whole Pages are downloaded into memory, but in chunks-fill the buffer each time, and then process the chunk of data before proceeding to next?
- However, pre-installed programs consume a lot of system memory, so disabling unnecessary startup programs can help maximize memory usage. To switch programs, right-click on the taskbar and select Task Manager from the context menu, then open the "Start" tab.

8.3.1 Organizing RAM: Stack vs Heap

What is Stack?

- A stack is a special area of computer memory that stores temporary variables created by a function. On the stack, variables are declared, stored, and initialized at runtime.
- This is temporary storage. After the end of the calculation task, the variable memory will be automatically cleared. The stack section mainly contains methods, local variables, and reference variables.

What is Heap?

- The heap is the memory used by programming languages to store global variables. By default, all global variables are stored in the heap space. It supports dynamic memory allocation.
- Heap is not automatically managed for you, nor is it closely managed by the CPU. It is more like a floating memory area.

Stack vs Heap

i. Stack is a linear data structure whereas Heap is a hierarchical data structure.

ii. Stack memory will never become fragmented whereas Heap memory can become fragmented as blocks of memory are first allocated and then freed.

iii. Stack accesses local variables only while Heap allows you to access variables globally.

iv. Stack variables can't be resized whereas Heap variables can be resized.

v. Stack memory is allocated in a contiguous block whereas Heap memory is allocated in any random order.

vi. Stack doesn't require to deallocate variables whereas in Heap deallocation is needed.

vii. Stack allocation and deallocation are done by compiler instructions whereas Heap allocation and deallocation is done by the programmer.

8.4 PERFORMANCE AND BATTERY LIFE

- When it comes to writing code, performance and battery life usually goes hand in hand which is good for one, usually good for the other.
- What needs to be optimized depends on the application. For battery or solar powered items, or items that need to respond immediately when the user presses a button, it's worth paying attention to performance or energy consumption.
- The greatest energy savings come from hardware design, especially if the device can shut down system modules when not in use, or put the entire CPU into a low-power hibernation state when code is ready or waiting for something to happen.
- However, software optimization is still important! After all, the sooner the main code completes execution, the faster the hardware goes to sleep.
- One of the easiest ways to increase code performance is to move to an event-driven model instead of requesting changes.
- The reason for this is to allow the device to remain in a low-power state longer and start operating when needed, rather than performing routine activities on a regular basis to check if something has changed and if it has actual work to be done.

8.5 LIBRARIES

- Nowadays, when you develop software for servers or desktops, you are used to having access to a huge number of possible libraries and frameworks that make your task easier. In an embedded world, tasks are often a bit more complex.
- With the increasing supply of system chips and their use of embedded Linux, where most server packages can be included in the same way as in "normal" Linux systems.
- On the other hand, microcontrollers still have too limited resources to simply use the libraries and code of the main operating system.
- We might be able to use the code as a starting point for writing your own version, but if it does lots of memory allocations or extensive processing, you probably are better
- off starting from scratch or finding one that's already written with microcontroller limitations in mind.
- Here are a few libraries which might be of interest:
- lwIP: lwIP, or LightWeight IP, is a full TCP/IP stack which runs in low-resource conditions. It requires only tens of kilobytes of RAM and around 40KB of ROM/flash. The official Arduino WiFi shield uses a version of this library.
- uIP: uIP, or micro IP, is a TCP/IP stack targeted at the smallest possible systems. It can even run on systems with only a couple of kilobytes of RAM. It does this by not using any buffers to store incoming packets or outgoing packets which haven't been acknowledged. It's quite common on Arduino systems which don't use the standard Ethernet shield and library, such as the Nanode board, using the Ethercard port for AVR.
- uClibc: uClibc is a version of the standard GNU C library (glibc) targeted at embedded Linux systems. It requires far fewer resources than glibc and should be an almost drop-in replacement. Changing code to use it normally just involves recompiling the source code.
- Atomthreads: Atomthreads is a lightweight real-time scheduler for embedded systems. You can use it when your code gets complicated enough that you need to have more than one thing happening at the same time (the scheduler switches between the tasks quickly enough that it looks that way, just like the multitasking on your PC).
- BusyBox: Although not really a library, BusyBox is a collection of a host of useful UNIX utilities into a single, small executable and a common and useful package to provide a simple shell environment and commands on your system.

8.6 DEBUGGING

- One of the most frustrating parts of writing software is knowing there is an error in your code.
- In embedded systems, this can be doubly frustrating, as there are usually fewer ways to find out what is happening so that a problem can be tracked down.
- The creation of devices for the Internet of Things further complicates the situation due to the introduction of both specialized electronic circuits (which may be malfunctioning or improperly designed) and communication with servers over the network.
- Modern IDEs for desktops have excellent support for delving into what is happening as code is running.
- You can set breakpoints that stop execution when a predefined set of conditions is met, in which case you can search the memory to see what's in it, evaluate the expression to see if your definition is correct, then traverse the code line by line to see what happens.
- You can even change the contents of memory or variables on the fly to affect the rest of the code execution, and in more advanced systems, rewrite the code while the program is running.
- Debugging environments for embedded systems are generally more primitive. If your embedded platform is running a more complete operating system such as Linux, then you are in a better position than if you are developing on a small microcontroller.
- Systems like embedded Linux usually support remote debugging with tools like gdb, the GNU debugger.
- This utility allows a debugger from a desktop computer to be connected to the built-in motherboard, usually via a serial connection, but sometimes also via Ethernet or a similar network channel.
- Once connected, you have access to many desktop-like debugging features - the ability to set breakpoints, step through code, and inspect variables and memory contents.

8.7 LETS SUM UP

To run embedded code we need some storage space. In this unit we have seen different times of memories like primary and secondary memory and different ways to manage yem. Making best use of RAM and organizing it with the concept of stack & heap has been also illustrated with an appropriate diagram. To enhance the performance we need good battery life and we have some inbuilt libraries like uClibc,LwIp that works well with the embedded platform and takes less storage space to run the code. In embedded systems, it is frustrating to identify the exact line of code or connection that is giving an error, however there are different ways to find out what is happening so that a problem can be tracked down. One of the common ways to find errors is with the GNU debugger.