Linux Internals

Under the Guidance of :- Gopalakrishna polimera

Semester :- 8th
Presented By :- Deepak Kumar Beniya
Reg. No. :- 1901227415

Agenda

- Introduction
- Linux Shell commands
- Linux Shell Scripting
- Process Management (fork, exec, clone)
- Threads
- IPC(Inter Process Communication)
  - Pipe, FIPO, msgQ, Shared Memory, Semaphores, Socket
- Socket Programming
- Schedular

Introduction

- Linux is the most popular operating system on the Internet, and it is particularly well-suited for developers working on various domains such as embedded applications, network infrastructure, and cybersecurity.

- Linux Internals include the kernel, process/thread management, file management system, and inter process communication (IPC), which together form the Linux Operating System.

- Linux offers enhanced security, high productivity, and a wide range of development features, making it an exceptional operating system. In this presentation, we will delve deeper into Linux internals to explore its core components and mechanisms.

Linux Shell commands

- The Linux command line, also known as the shell, terminal, console, or prompt, is a text-based interface that enables users to interact with the operating system and execute various commands.

- Using commands like **mkdir**, **pwd**, **man**, **ls**, **cd**, and many more, users can perform a wide range of tasks such as creating directories, view manual pages, list files, and managing processes.

- The Linux command line is a powerful tool for developers, system administrators, and anyone who wants to work efficiently with the Linux operating system.

- In the simplest terms, a shell script is a file containing a series of commands. The shell reads this file and carries out the commands as though they have been entered directly on the command line.

➢ **Example-**

```
$ echo '#!/bin/sh' > my-script.sh
$ echo 'echo Hello World' >> my-script.sh
$ chmod 755 my-script.sh
$ ./my-script.sh
Hello World
$
```

- Shell scripts typically start with a **shebang (#!)** to specify the shell interpreter (e.g., **#!/bin/bash**).
- Standard input, output, and error streams (stdin, stdout, stderr) can be redirected using symbols (e.g., >, >>, <).

Process Management (fork, exec, clone)

- ✓ Process management in Linux involves creating and controlling processes, which are instances of running programs.
- ✓ Three key operations for process management are fork, exec, and clone.

1. **fork:**
   - ✓ The fork system call creates a new process by duplicating the existing (parent) process.
   - ✓ The new process, known as the child process, inherits the memory, file descriptors, and other attributes of the parent process.
   - ✓ The fork operation allows for the creation of a new process that can run concurrently with the parent process.

2. **exec:**
   - ✓ The exec system call replaces the current process with a new program.

- ✓ It loads the specified program into the current process's memory space and starts its execution.

- ✓ The exec operation is commonly used to launch a different program from within a process, replacing the process's code and data with the new program.

3. **clone:**
   - ✓ The clone system call is like fork but provides more control over process creation.
   - ✓ It allows creating a new process with different characteristics, such as sharing specific resources with the parent process or specifying a different entry point.
   - ✓ The clone operation is commonly used for thread creation, as it enables the creation of lightweight execution units within a process.

- **What is Threads?**

✓ Thread is an execution unit that is part of a process. A process can have multiple threads, all executing at the same time. It is a unit of execution in concurrent programming. A thread is lightweight and can we managed independently by a scheduler. It helps you to improve the application performance using parallelism.

- Multiple threads share information like data, code, file, etc. We can implement threads in two different ways:

1. **Kernel level threads**
2. **User level threads**

- **User level threads:**
- ✓ The **pthread_create** function is used to create a new user-level thread and associates it with the **thread_function**.
- **Syntax:**

```
#include <pthread.h>
int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
void *(*start_routine) (void *), void *arg);
```

- ✓ The **pthread_join** function is used to wait for the user-level thread to finish execution and retrieve its exit status.

- ✓ **Syntax:**

```
int pthread_join(pthread_t thread, void **retval);
```

IPC(Inter Process Communication)

- IPC in Linux facilitates communication and data exchange between processes, enabling collaboration, synchronization, and coordination.
- **IPC Mechanism :**
  - FIFO
  - Pipes
  - Message Queues
  - Shared Memory
  - Semaphore
  - Socket

✓ **FIFO :**

- A first-in, first-out file is a pipe that has a name in the file system.
- It also called named pipes.
- When a blocked **SCHED_FIFO** thread becomes runnable, it will be inserted at the end of the list for its priority.

Pipe, msgQ, Shared Memory, Semaphores, Socket

- **Pipe :**Unidirectional communication channels between related processes, with one process writing to the pipe and the other reading from it.

- **msgQ :** Message queues are two-way IPC mechanism for communicating structured message.
- **Shared Memory :** Shared memory allows two or more processes to access the same memory.
- When one processes changes the memory, all the other processes see the modification.

- **Semaphores :** Semaphores are synchronization primitives used to control access to shared resources in IPC.

- **Socket :** A sockets is hey communication mechanism that allows client/server system to be developed either locally on a single machine or a cross networks.

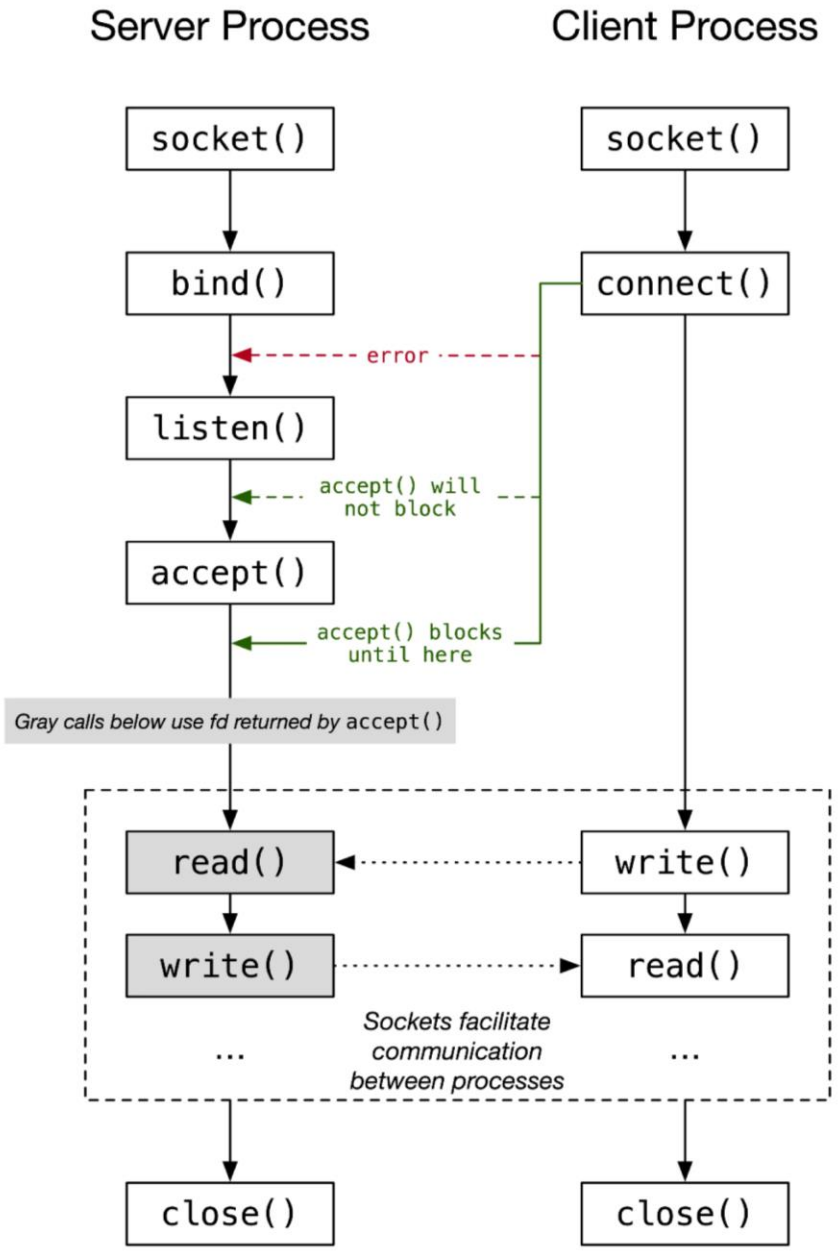
Socket Programming

- Socket Programming is a method to connect two nodes over a network to establish a means of communication between those two nodes. A node represents a computer or a physical device with an internet connection.

- **Syntax:**

  int socket(int domain, int type, int protocol)

- The nodes are divided into two types, server node and client node.
- The client node sends the connection signal and the server node receives the connection signal sent by the client node.



✓ **Schedular:**

- The scheduler is responsible for determining the order and duration of process execution on the CPU.

- It allocates CPU time to processes based on scheduling algorithms like First-Come, First-Served (FCFS), Round Robin, or Priority Scheduling.

- The scheduler ensures fairness, efficiency, and responsiveness in the execution of processes.

Rugged Board

https://community.ruggedboard.com

Agenda

- Introduction
- Rugged Board Interfaces
    - Interfaces
    - Block Diagram
    - Functional Components
        - Power Supply
        - Jumpers
        - Switches
        - User LED (GPIO)
        - Industrial Field Interfaces
        - Ethernet
        - USB 2.0
        - Secure digital Memory card + SIM (Dual Connector)

Introduction

- The phyCORE-A5D2x RuggedBoard is a System in Peripheral (SIP) that serves as a low-cost software development platform for Microchip's A5D2x CPU.

- It provides a feature-rich solution with many kinds of standard interfaces, giving it an adaptable basis for a wide range of applications.

- Features:

- 1 x Ethernet
- 2 x RS-232
- 1 x RS-485 (Isolated)
- 1 x CAN
- 4x DIN (Isolated)
- 4x DOUT
- 1 x LVDS Display
- 1 x Micro SD
- 1 x SIM
- 2 x USB 2.0
- 1 x mikroBUS
- 1 x 60 PIN Expansion Headers

Block Diagram

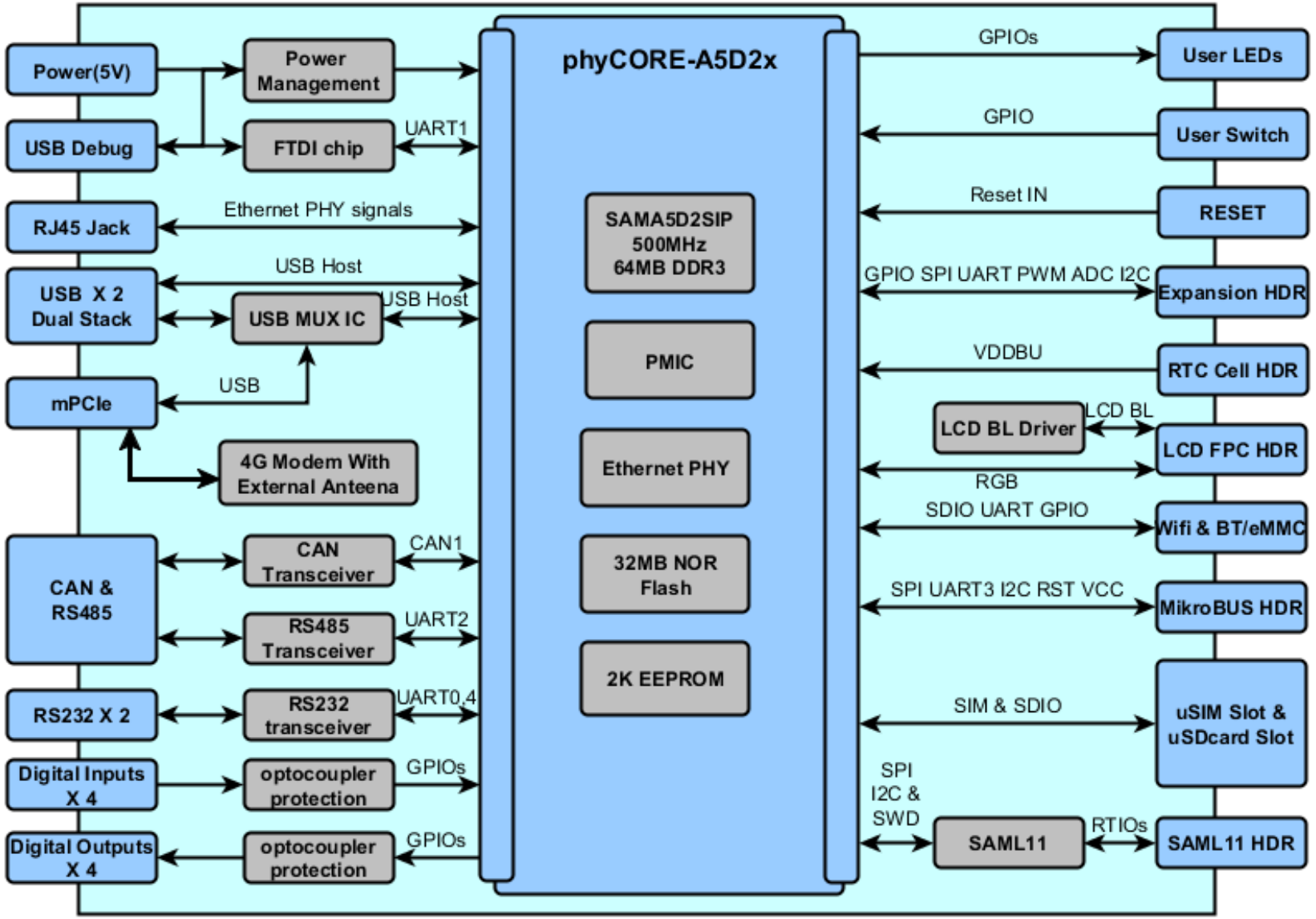


Fig 2: Block Diagram

Functional Components

- This section describes the functional components of the RuggedBoard. Each subsection details a particular connector/interface and associated jumpers for configuring that interface. Figure 3 below shows the front side of RuggedBoard-A5D2x.

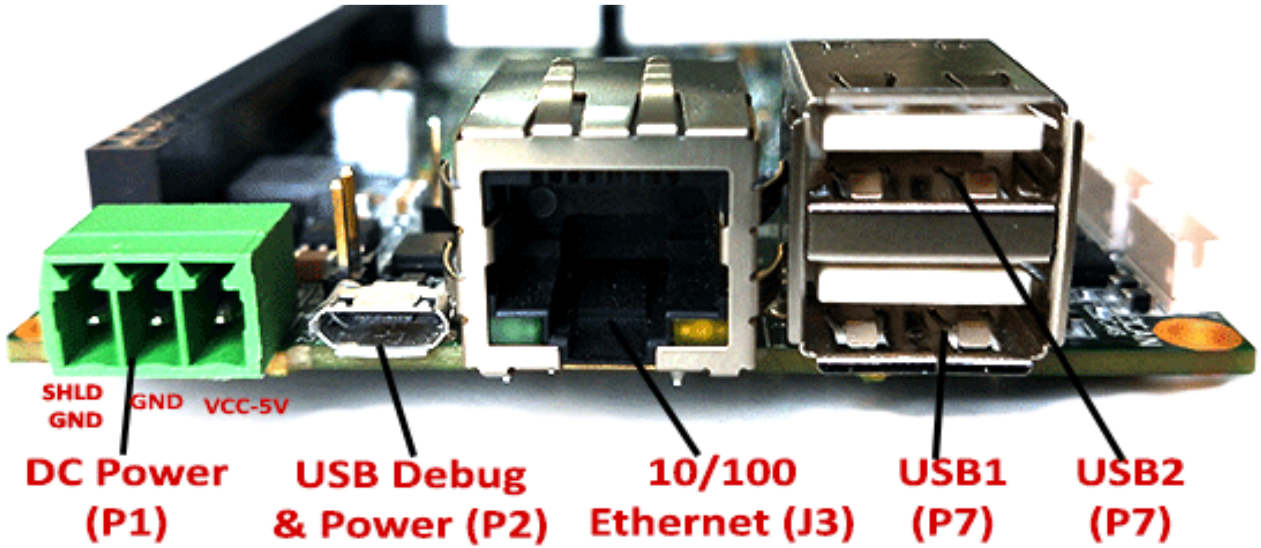


Fig 3: Front Panel

Power Supply and Jumpers (J2)

- **Power Supply:-** The RuggedBoard is available with two different power supply connectors. Power in through industrial standard three pin connector and microUSB connector.

Fig 4: Power Supply Connector

- **Jumpers (J2):-**This jumper (J2) is used to ON/OFF of the Board. If jumper (J2) is not present in the board then the board will not power on. So jumper (J2) must present on the board.

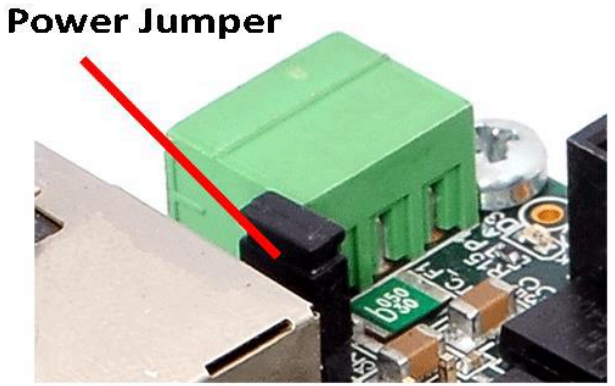


Fig 5 : Power Jumper

Switches and User LED (GPIO)

✓ **Switches:** The RuggedBoard contains two switches .

a. System Reset Button(RST.SW2):

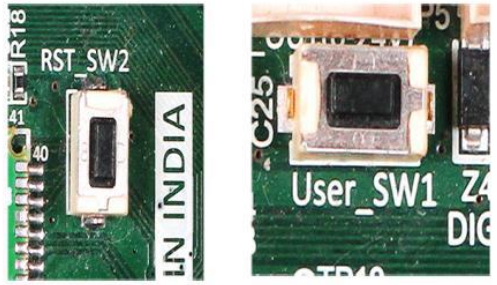


Fig 6: Reset Switch and User Level Switch

b. User_SW1 :

PIN	Switch No	SIGNAL NAME	MRAA Mapped Pins
NO1	SW1	PC12/GPIO_EN	35

User_SW1 button is used for GPIO user level input. The User Level Switch is shown in the Figure 7.

- **User LED (GPIO) :** The RuggedBoard populated with three user controllable LEDs.Fig. 8 shows the location of the LEDs. Their functions are listed in Table given below.

PIN	LED No	SIGNAL NAME	MRAA Mapped Pins
NO1	LED_1 (D4)	PC13/GPIO_LED	61
2	LED_2 (D7)	PC17/GPIO_LED	62
3	LED_3 (D17)	PC19/GPIO_LED	63

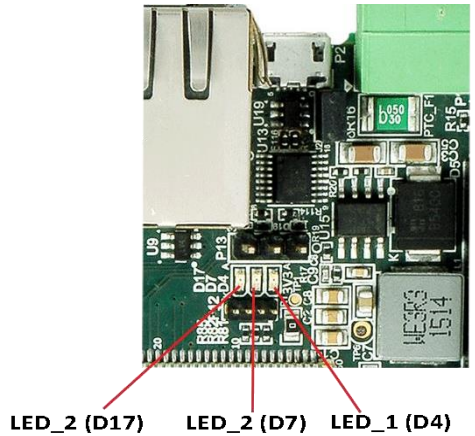


Fig 7: User LEDs

Industrial Field Interfaces

- **Industrial Field Interfaces :**
- RuggedBoard-A5D2x equipped with multiple Industrial field interfaces. It has 1x RS485, 1x CAN, 2x RS232, 4x DIN, 4x D Out.

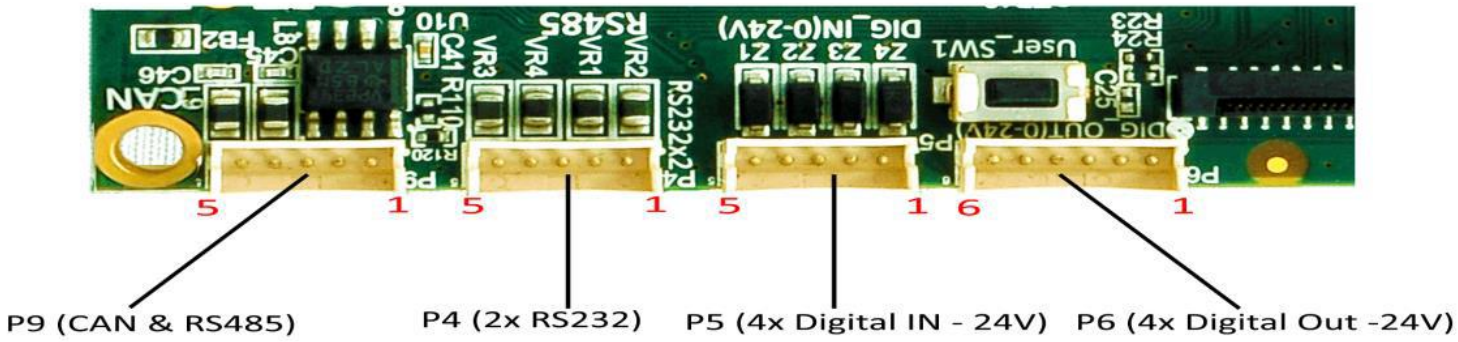


Fig 8 : Field Interfaces

Ethernet (J3) and USB 2.0 (P7)

- **Ethernet (J3) :** The on-board SOM has a 10/100 Mbps Ethernet controller (KSZ8081RNA), which enables direct connection to any 10/100 Mbps Ethernet-based Local Area Network, allowing full interaction with local servers and wide area networks such as the Internet. The SOM's ethernet signals are linked via an RJ45 MagJack.

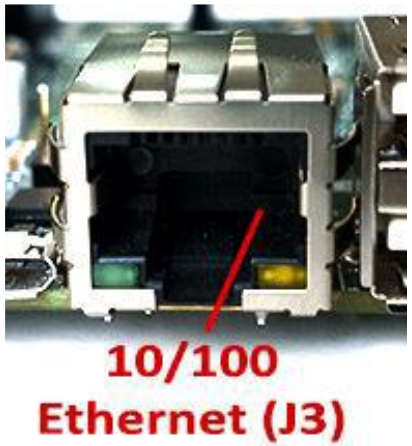


Fig 9: Ethernet connector

- **USB 2.0 :** In RuggedBoard there are two stacked USB2.0 Host Ports. Both USB1 &USB2 are configured as Host. USB2 signal are also used for the mPCIe port (P8). The switching happens through USB mux switch configuration. This configuration can be done by either Software or Hardware method.

Fig 10: USB Dual Stack Connector

Secure digital Memory card + SIM (Dual Connector)

- The RuggedBoard provides a standard micro SDHC card slot at J4 for connection to MMC/SD interface cards. It allows easy and convenient connection to peripheral devices like SD-Card and MMC cards. Power to the SD-Card interface is supplied by inserting the appropriate card into the MMC/SD connector, which features card detection, a lock mechanism and a smooth extraction function by Push-in/ Push-out of card.

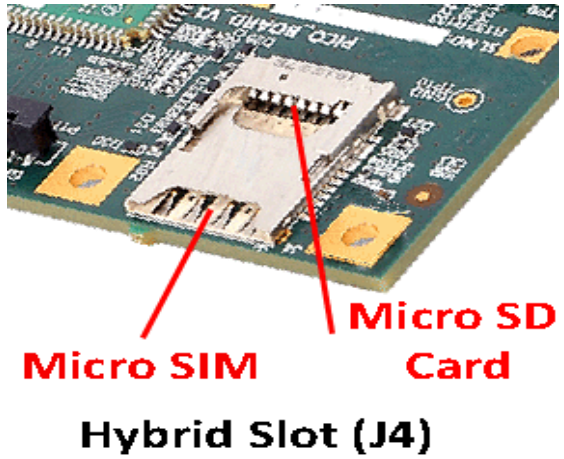


Fig 11: Memory Card and SIM Dual connector


Thank You