Embedded firmware design approaches:

Firmware design depends on the complexity of functions to be performed and speed of operation required. Two basic approaches of Embedded firmware design are:

① Conventional procedural based firmware design (Super loop model)

② Embedded operating system (OS) based design.

① Super loop based model:

* Super loop firmware design approach is adopted for applications that are not time critical and where the response time is not so important.
* These are the embedded systems where the code is executed task by task where mining deadlines are acceptable.
* Super loop based design doesnot require an operating system since there is no need for scheduling and assigning of priority to each task.
* In this design, priority of tasks are fixed and the order of execution is also fixed.

Ex:- Reading/Writing data to and from a card using card reader requires a sequence of operations like checking the presence of card, authentication of the operation, reading/writing etc.

It should follow a sequence and the combination of all these series of tasks constitute a single task.

Advantages:

* Superloop design is simple and straight forward without any OS related overheads.
* No need of special operating system (OS)

Disadvantages:

* Any failure in the part of task will affect the total system.
* Lack of real timeliners brings the probability of missing events.

The firmware execution flow of superloop model will be

① Configure the command parameters and perform initialization of various hardware components, memory, registers etc.
② Start the first task and execute it.
③ Execute second task
④ Execute next task
⑤ ⋮
⑥
⑦ Execute the last defined task.
⑧ Jump back to the first task and follow the same flow.

Ex:-
```
Void main()
{
    configurations();
    Initializations();
    while(1)
    {
        Task1();
        Task2();
        ⋮
    } Taskn();
}
```

② Embedded Operating System (OS) based approach:

There are two types of operating systems. They are:

(a) General Purpose Operating System (GPOS)

[Ex:- Windows XP, Unix, Linux]

(b) Real time Operating System (RTOS)

[Ex:- Symbian, Elinux, Thread X, Vx Works etc.]

(a) General Purpose Operating System:

This is very similar to conventional personal computer based application development where the device contains an operating system and user applications will run on top of it.

Ex:- Windows, Unix, Linux etc.

Examples of Embedded products which use GPOS are Personal Digital Assistants (PDA's), Hand held devices/ portable devices and point of sales (POS) terminals.

(b) Real time Operating System:

RTOS Contains real time kernel responsible for performing pre-emptive multitasking, schedules for scheduling task, multiple threads etc.

RTOS allows flexible scheduling of system resources like the CPU and memory and offers some way to communicate between tasks.

Ex:- Windows CE, PSOS, Thread X, Embedded linux, Symbian Micro C/OS-II etc. Most of the mobile phones are built around the popular RTOS 'symbian'.