4) Phase/State driven code.

- It uses nested if-then statements, case-statements or a finite state automaton to break up the processing of a function into discrete code segments.
- This allows temporary suspension of code segment without loss of critical data.

```
procedure task;
begin
    case flag of
1: begin
      perform_part_1;
      flag:=2
   end
2: begin
      perform_part_2;
      flag:=3
   end
3: begin
      perform_part_3;
      flag:=1
   end
end.
```

At the end of each state a flag is set and the process is terminated. Upon restarting, the process resumes where it left off.

5. Coroutines:-

→ In this scheme, two or more tasks are coded in the state-driven fashion, just and after each phase is completed, a call is made to a central dispatcher, The dispatcher holds the program counter for a list of tasks that are executed in round robin fashion. ie it selects next task to execute. This task then executes until its next phase is completed and the central dispatcher is called again.

→ If there is only 1 task, then it becomes cyclic. & The communication both task is achieved by global variable.

```
void task_a(void)
{
    switch (state_a)
    {
        case1: phase_a1();
               break; /* to dispatcher */
        case2: phase_a2();
               break;
        case3: phase_a3();
               break;
    }
```

```
void task_b(void)
{
    switch (state_b)
    {
        case1: phase_b1();
               break;
        case2: phase_b2();
               break;
        case3: phase_b3();
               break;
    }
```

State_a & state_b are global variable managed by dispatcher.

2. Interrupt-Driven System:-

- Interrupt ko concept ho:
  - interrupt only system
  - H/W interrupt आउदा Real Time Clock or External device send interrupt to Interrupt controller (IC).
  - IC send interrupt signal to CPU depending on order of arrival & priority
  - ~~Waile context switching,~~
    ↓
    ~~is sme~~
  - Context is a snapshot of task being processed
  - While context switching, sufficient info is saved so that resuming interrupted task is possible.
  - Context switching ko bela मा Interrupt बन्द हुन्छ।

3. Preemptive Priority System:-

- higher priority task आउदा भने lower priority task preempt हुन्छ।
- higher priority task ले Resource लिए lower priority task को resource starvation हुन सक्छ।
- Rate monotonic system मा higher the execution frequency, higher the priority.

4) Hybrid Scheduling Systems:

① combination of preemptive & round robin scheduling
↳ same priority handled with round robin fashion.

② combinations of interrupt & fixed rates or not fixed rate. (sporadically).

③ combination of foreground & background systems

The Task Control Block Model

Rate Monotonic Algorithms:-

	Burst Time	Arrival Time	Period.
P_1	2	0	10
P_2	1	0	5
P_3	5	0	30
P_4	2	0	15.

Here N = 4

Least Upper bound of utilization for scedulability $= N(2^{1/N} - 1)$
$= 4(2^{1/4} - 1)$
$= 0.7565.$

Now; Utilization of $P_1 = \frac{T_1}{P_1} = \frac{2}{10} =$

Utilation of $P_2 = \frac{T_2}{P_2} = \frac{1}{5}$

" " $P_3 = \frac{T_3}{P_3} = \frac{5}{30}$

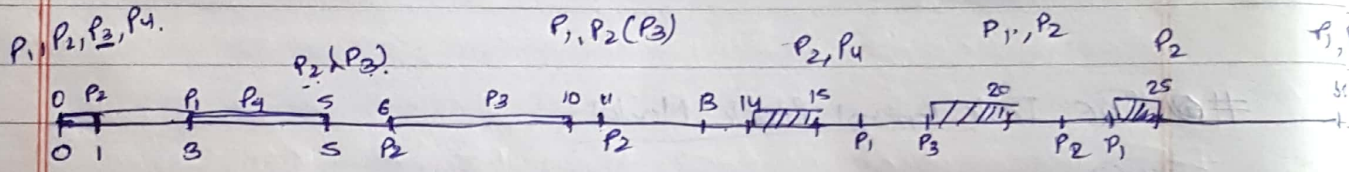
" " $P_4 = \frac{T_4}{P_4} = \frac{2}{15}.$

$\therefore$ Total Utilization $= \frac{2}{10} + \frac{1}{5} + \frac{5}{30} + \frac{2}{15} = 0.7$

Since $0.7 < 0.7568$. Thus these processes can be scheduled by rate monotonic scheduling.

~~Highest~~ L.C.M of Periods = 2 | 10, 5, 30, 15
5 | 5, 5, 15, 15
3 | 1 1 3 3
1 1 1 1

Here, highest common divisor is 5.



P2