MTRN3500
Computing Applications in Mechatronics Systems

Process Management

T2 - 2020

Motivation

- We have to get the multi-module software system to operate. The problem is that they are all independent processes and they must be started independently.
- Given that each of the processes running have varying degrees of importance, we need to make sure they are all running as expected.
- At the end of the operation, we need to safely shutdown all the processes.
- To begin with we need a startup sequence.
- Then we need a way to start them in that sequence.
- During the operations we need to monitor the operations of all processes. Which process will be entrusted to do that? How will that process do the monitoring?
- Who will be responsible for shutting down.
- All of the above is process management.
- How do we handle a management failure?

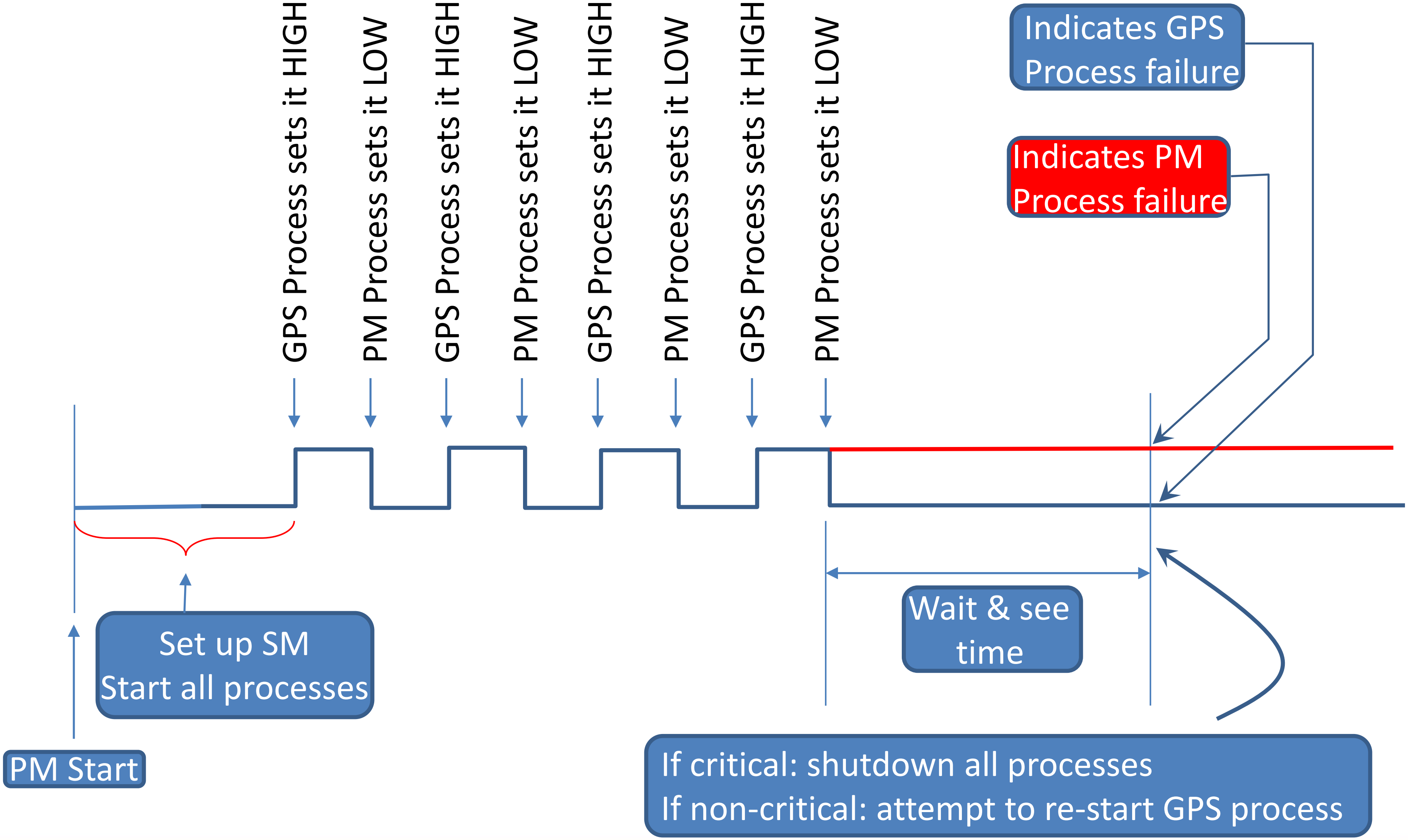
Process Management Strategies

- Among many ways one could think of, we will look at two methods
  - Heartbeats
  - Time stamps

Heartbeats

- We allocate heartbeats linking each module to the process management module.
- Each module must have its own heartbeat signal.
- The process management will monitor each of the heartbeats.
- If no heartbeat detected, the process management will go in to “wait and see” mode.
- If failed, the process management will determine the level of importance of the failed process.
- We consider only two levels: Critical and non-critical.
- If critical: Shutdown all
- If non-critical: Attempt to re-start.

How Does Heartbeats Work (PM & GPS processes)?



If critical: shutdown all processes
If non-critical: attempt to re-start GPS process

How Do We Handle Heartbeats in Software?

```
unsigned char Heartbeats; // Make it part of shared memory
```

				Vehicle	Xbox	Laser	GPS	PM
	x	x	x	x	x	x	x	x
GPS Process sets it HIGH	x	x	x	x	x	x	1	x
PM Process sets it LOW	x	x	x	x	x	x	0	x
GPS Process sets it HIGH	x	x	x	x	x	x	1	x
PM Process sets it LOW	x	x	x	x	x	x	0	x
				…				
GPS flag dormant at LOW: GPS process failed	x	x	x	x	x	x	0	x

How Do We Handle Heartbeats in Software?

```
unsigned char Heartbeats; // Make it part of shared memory
```

				Vehicle	Xbox	Laser	GPS	PM
	x	x	x	x	x	x	x	x
GPS Process sets it HIGH	x	x	x	x	x	x	1	x
PM Process sets it LOW	x	x	x	x	x	x	0	x
GPS Process sets it HIGH	x	x	x	x	x	x	1	x
PM Process sets it LOW	x	x	x	x	x	x	0	x
				…				
GPS flag dormant at HIGH: PM process failed	x	x	x	x	x	x	1	x

Data Structures to Handle Heartbeats

```
struct ModuleFlags
{
  unsigned char PM:1,
                GPS:1,
                Laser:1,
                Xbox:1,
                Vehicle:1,
                Unused:3;
};
```

```
//For collective handling
//of individual bits
union ExecFlags
{
  unsigned char Status;
  ModuleFlags Flags;
};
```

```
Struct PM
{
  ExecFlags Heartbeats;
  // . . .
};
```

Data Structures to Handle Heartbeats

```
int main()
{
    PM* PMSMPtr = nullptr;
    // . . .
    PMObj.SMAccess();
    PMSMPtr = (PM*)PMObj.pData;
    PMSMPtr->HeartBeats.Flags.PM = 1; // or 0
    PMSMPtr-> HeartBeats.Flags.GPS = 1;// or 0
    // . . .
    // Can take values in the range 0-255
    PMSMPtr-> HeartBeats.Status = 0xFF;
    // . . .
    return 0;
}
```

Example: Checking GPS Heartbeats by PM

```
while(1)
{
    if(PMSMPtr->Heartbeats.Flags.GPS == 1)
    {
       PMSMPtr->Heartbeats.Flags.GPS = 0;
    }
    else
    {
       if (GPS Critical)
        shutdown all (how?); break;
       else
       {
          if (!GPS running)
             // re-start GPS
          else
          {
            // kill GPS
            // restart GPS
          }
      }
   }
}
```

Example: Checking PM Heartbeats by GPS

```
while(1)
{
    if(PMSMPtr->PMHeartbeats.Flags.GPS == 1)
    {
      PMSMPtr->PMHeartbeats.Flags.GPS = 0;
        //Reset WaitAndSeeTime
    }
    else
    {
       //Accumulate WaitAndSeeTime
       if(WaitAndSeeTime > WAIT_TIME)
          //Request Shutdown all
    }
}
```

- PM Heartbeat checks must be carried out from all processes.
- An individual bit must be used for that purpose from within each process in order to not confuse with resetting by other modules.

Data Structures to Handle Heartbeats

```
struct ModuleFlags
{
  unsigned char PM:1,
                GPS:1,
                Laser:1,
                Xbox:1,
                Vehicle:1,
                Unused:3;
};
```

```
//For collective handling
//of individual bits
union ExecFlags
{
  unsigned char Status;
  ModuleFlags Flags;
};
```

```
Struct PM
{
  ExecFlags Heartbeats;
  ExecFlags Shutdown;
};
```

Shutdown Mechanisms

```
// To shutdown GPS process only
PMSMPtr->Shutdown.Flags.GPS = 1;
// To shutdown all processes
PMSMPtr->Shutdown.Status = 0xFF;
// Sequential shutdown
PMSMPtr->Shutdown.Flags.Vehicle = 1;
System::Threading::Thread::Sleep(100);
PMSMPtr->Shutdown.Flags.Laser = 1;
System::Threading::Thread::Sleep(100);
PMSMPtr->Shutdown.Flags.GPS = 1;
System::Threading::Thread::Sleep(100);
PMSMPtr->Shutdown.Flags.Xbox = 1;
System::Threading::Thread::Sleep(100);
```

Shutdown Statements

```
 PMSMPtr->Shutdown.Flags.GPS = 0;

 while(!PMSMPtr->Shutdown.Flags.GPS)
 {
     PMSMPtr->Heartbeats.Flags.GPS = 1; // Set heartbeat flag
     // calculate time stamp if needed
     // extract GPS data from receivers

     // Check PM heartbeat
     if(PMSMPtr->PMHeartbeats.Flags.GPS == 1)
     {
         PMSMPtr->PMHeartbeats.Flags.GPS = 0;
         //Reset WaitAndSeeTime
     }
     else
     {
        //Accumulate WaitAndSeeTime
        if(WaitAndSeeTime > WAIT_TIME)
           //Request Shutdown all
     }
     if(_kbhit()) break;
     System::Threading::Thread::Sleep(50);
}
```

Time Stamps

- Limitation: Must use the same clock.
- We maintain an array of Time Stamps.
- The process management will monitor each of the time stamps and their freshness.
- If excessive delay is detected, the process is considered dormant.
- The process management will then determine the level of importance of the dormant process.
- We consider only two levels: Critical and non-critical.
- If critical: Shutdown all
- If non-critical: Attempt to re-start.

Time Stamps

```
struct PM
{
   ExecFlags Shutdown;
   double TimeStamps[8];
};
```

Example: Checking GPS Time Stamps by PM

```
while(!PMSMPtr->Shutdown.Flags.PM)
{
    PMTimeStamp = GetTimeStamp();
    for(int i = 1; i < NumberOfProcesses; i++)
    {
       if(PMTimeStamp - PMSMPtr->TimeStamps[i] > WAIT_TIME)
       {
          if (Process[i] == Critical)
             PMSMPtr->Shutdown.Status = 0xFF; break;
          else
          {
             if (Process[i] != running)
                // re-start Process[i]
             else
             {
               // kill Process[i]
               // restart Process[i]
             }
          }
       }
    }
}
```

Example: Checking PM Time Stamps by GPS

```
while(!PMSMPtr->Shutdown.Flags.GPS)
{
    GPSTimeStamp = GetTimeStamp();
    if(GPSTimeStamp - PMSMPtr->TimeStamps[0] > WAIT_TIME)
    {
        PMSMPtr->Shutdown.Status = 0xFF;
    }
}
```