ADCSS16

Iridium Next MHS*TR SW*

A first step towards mixed criticality

G. Veran – R. de Ferluc

26/10/2016

ThalesAlenia Space
A Thales / Finmeccanica Company

83230347-DOC-TAS-EN-003

Agenda

- IRN MHSTR SW integration
- Overview of OSTRALES for Time and Space partitioning
- FDIR strategy
- New challenges and solutions

IRN MHSTR SW Integration

- Iridium computer
  - PFC based on LEON3 UT699 including MMU and cache
- MHSTR overview :
  - 3 optical heads connected by 3 point-to-point SpW links to the processor module
  - MHSTR SW integrated in the satellite central Platform SW (OBSW)
- Benefits :
  - Reduction of HW on OH side => cost reduction of equipment
  - High benefits thanks to IRN constellation scale

IRN MHSTR SW Integration

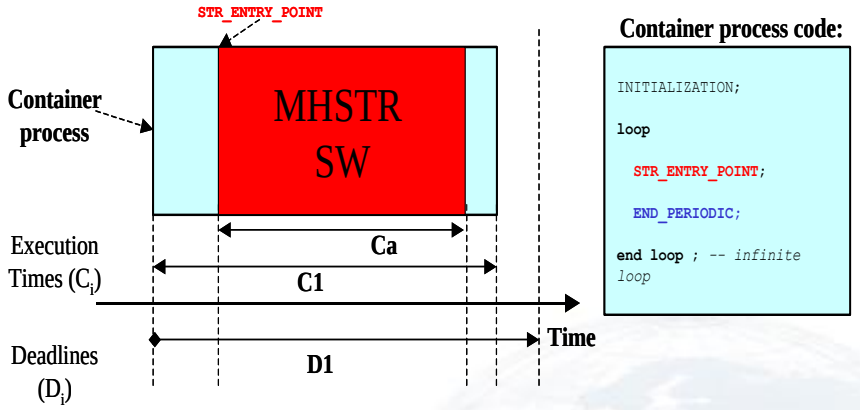
- MHSTR SW is provided as a C library by supplier
- MHSTR is built has an independent image
  - No link edition between OBSW and MHSTR !
  - Interface done with to two “trampoline” structures

```
type MHSTR_TO_PFSW_TRAMPOLINE_T is record
   -- offset: +0x00
   VERSION                  : BASIC_TYPES.UINT32_T;

   -- internal functions - offset: +0x100
   INIT_MHSTR_SW_ADDRESS    : SYSTEM.ADDRESS;
   EXE_MHSTR_SW_ADDRESS     : SYSTEM.ADDRESS;

   -- input variables - offset: +0x200
   MHSTR_TC_ADDRESS         : SYSTEM.ADDRESS;
   SV_VELOCITY_ADDRESS      : SYSTEM.ADDRESS;

   OH1_PELTIER_STATUS_ADDRESS  : SYSTEM.ADDRESS;
   OH2_PELTIER_STATUS_ADDRESS  : SYSTEM.ADDRESS;
   OH3_PELTIER_STATUS_ADDRESS  : SYSTEM.ADDRESS;

   MHSTR_TM_ADDRESS         : SYSTEM.ADDRESS;
   OH1_TM_ADDRESS           : SYSTEM.ADDRESS;
   OH2_TM_ADDRESS           : SYSTEM.ADDRESS;
   OH3_TM_ADDRESS           : SYSTEM.ADDRESS;

   OH12_FL_ALIGN_Q_IN_ADDRESS  : SYSTEM.ADDRESS;
   OH23_FL_ALIGN_Q_IN_ADDRESS  : SYSTEM.ADDRESS;
   OH31_FL_ALIGN_Q_IN_ADDRESS  : SYSTEM.ADDRESS;

   OH1_AIT_ALIGN_MAT_ADDRESS   : SYSTEM.ADDRESS;
   OH2_AIT_ALIGN_MAT_ADDRESS   : SYSTEM.ADDRESS;
   OH3_AIT_ALIGN_MAT_ADDRESS   : SYSTEM.ADDRESS;

end record;
```

- MHSTR SW can be reloaded separately in flight

IRN MHSTR SW Integration

- A SW container has been developed to host the library :
  - Manages the interface with other OBSW components
  - Manages the interface with the ground (TM/TC)
  - Manages SpW ACQ/CMD cycles
- Defined with TAS-F component framework (CCM almost ~ OSRA)
- MHSTR SW tasking
  - 2 entry points
    - Autonomous Attitude Determination
    - Autonomous Tracking
  - Scheduled by two RTOS processes (4Hz and 8Hz)

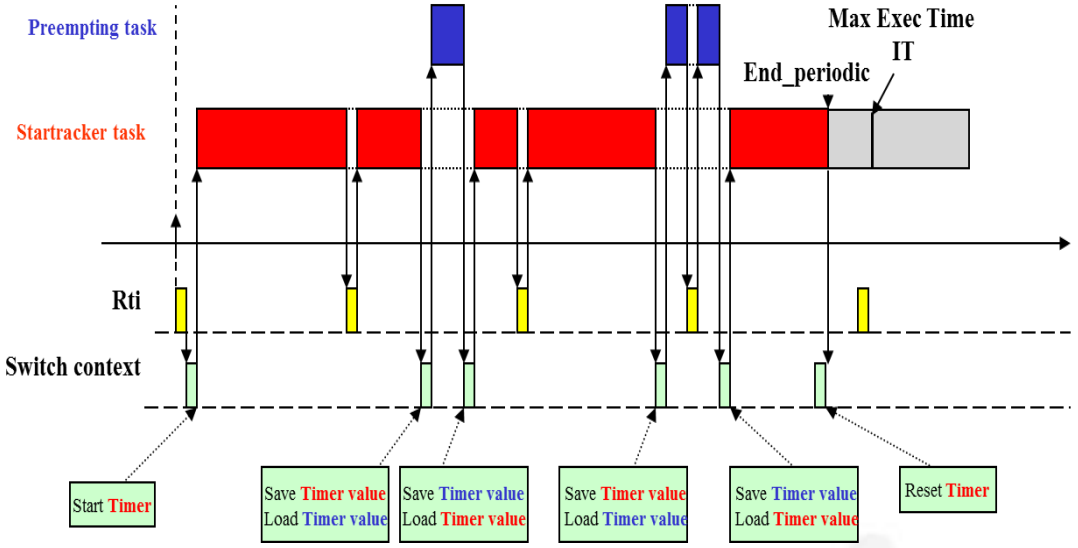


Overview of OSTRALES for TSP

- Introduction of Time and Space partitioning
  - The objectives were:
    - To avoid fault propagation from the MHSTR SW to the OBSW
    - To detect MHSTR SW faults
    - To take appropriate recovery actions
  - Time partitioning
    - Make sure the MHSTR SW does not exceed a predefined maximum execution time
  - Memory partitioning
    - Make sure that MHSTR SW does not write outside its memory areas
  - Instruction partitioning
    - Make sure the MHSTR SW does not use processor critical instructions

Overview of OSTRALES for TSP

- Time partitioning
  - A Maximal Execution Time (MAET) is defined for each task, based on:
    - Supplier WCET measurements
    - OBSW scheduling and margins
  - Execution time monitoring is implemented by RTOS using an HW timer
    - MAET value is loaded in the timer at each task cycle start
    - Timer value is saved and reloaded at each task preemption



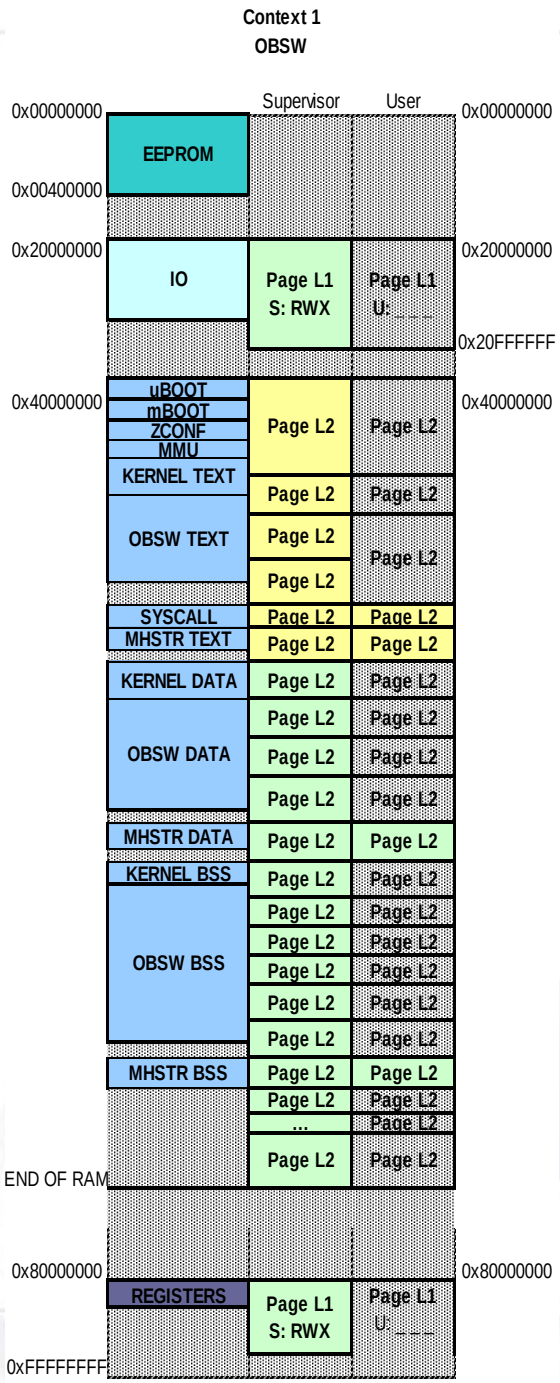
  - Timer is configured to trig an interrupt when MAET is reached

Overview of OSTRALES for TSP

- Instruction partitioning
  - OBSW tasks are executed in supervisor mode and so can execute all instructions including the privileged ones
  - MHSTR tasks are executed in user mode. If they attempt to execute a privileged instruction a trap is raised by the processor
  - The trap is handled by the supervisor software
  - LEON3 mode is configured by OSTRALES at each context switch

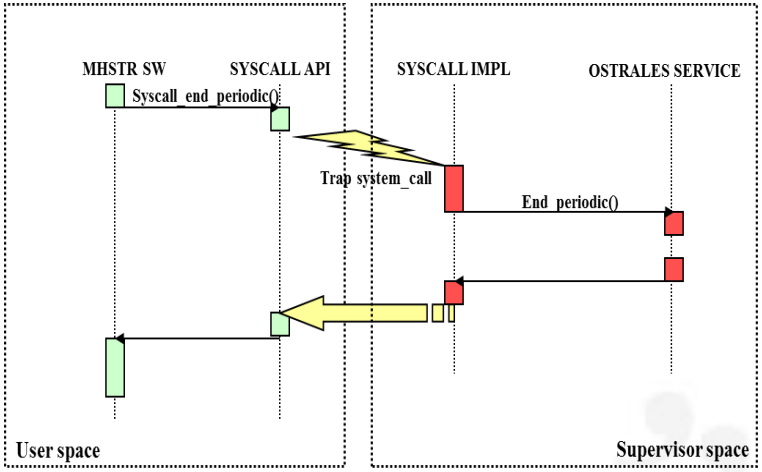
Overview of OSTRALES for TSP

- Space partitioning
  - Implementation based on HW CPU mechanisms
    - MMU
    - Processor user/supervisor mode
  - Direct 1-1 mapping between virtual&physical addresses
  - 1 MMU context for MHSTR SW giving access to :
    - MHSTR Text, Data and Bss sections
    - OSTRALES Syscall section
  - MHSTR SW cannot access all other memory areas
  - Additional memory protections are implemented to protect specific memory areas also from supervisor tasks (e.g. code area…)



Overview of OSTRALES for TSP

- Communication OBSW/MHSTR SW
  - OBSW can access the whole MHSTR SW memory
    - OBSW writes input data in a dedicated memory area of the MHSTR SW
    - MHSTR writes its output data in its own memory area
    - Scheduling prevents concurrent accesses
  - Synchronous services are implemented by OSTRALES System calls
    - getOBT, getRemainingExecTime…



FDIR strategy

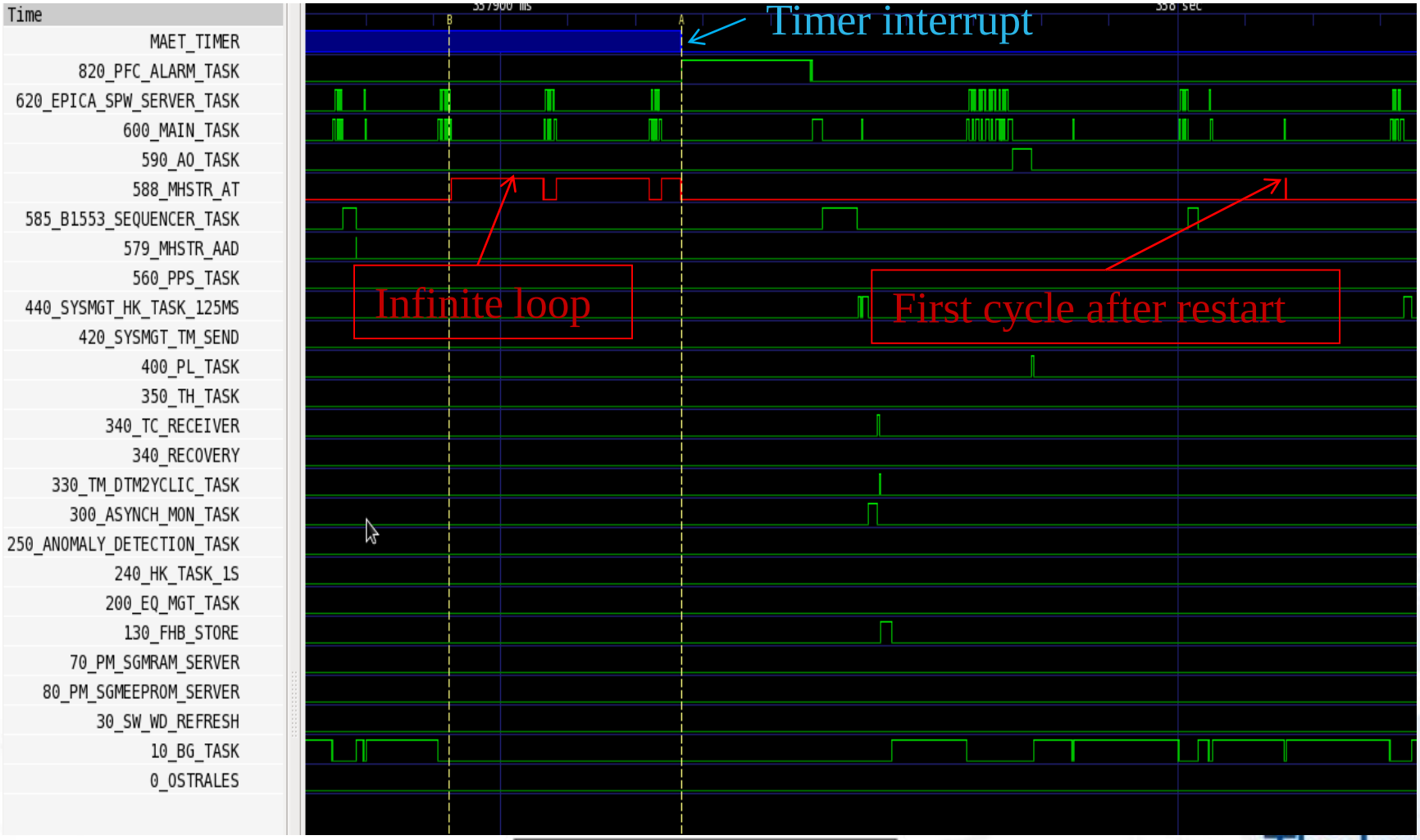
- Faults are detected by handling the following processor traps
  - MMU traps, Privileged instruction trap and MAET Timer interrupt
  - But also general Integer Unit traps (Division by zero, window overflow…)

- The FDIR defines corrective actions
  - A new FDIR level has been defined (level alpha)
    - Handles first MHSTR error occurrence
    - MHSTR SW is reset and restarted (no impact on other SW components)
    - Pure SW reconfiguration

  - In case of persistent error, FDIR escalation is done (classical FDIR steps)
    - 2nd occurrence : PM reset and restart in NOM
    - 3rd occurrence : PM reset and restart in SAFE (MHSTR not used in SAFE)

FDIR strategy

- Example (extracted from MHSTR test procedure):
  - MHSTR code is patched to introduce infinite loop during the computation



New challenges and solutions

- Current OSTRALES solution is well suited for applications :
  - Purely synchronous/periodic (whose scheduling can be easily managed)
  - With limited interfaces with other applications
  - With limited IO needs and constraints (response time, frequency…) that can be managed externally by Host SW
- But is not suitable for :
  - Complex applications with strong constraints (interface, IO, scheduling…)
- Future avionics will increase need of mixed SW criticality
  - Higher computing power will enable a higher centralization of avionics SW
    - GNSS, Payload, On-board planning, Nav Cam processing
- Additional constraints will have to be taken into account
  - Porting on new target (ARM, Power PC)
  - Multicore based computers
  - Application with security needs

New challenges and solutions

- The solutions will be based on hypervisors
  - PikeOs
  - XtratuM
- The best hypervisor configuration approach will have to be defined case by case
  - Mapping of partitions on cores
  - Strict time partitioning or priority based partition scheduling
  - Strict space partitioning or used of shared memory areas
- Additional studies has to be performed to managed resource interferences on such architecture
  - Internal processor bus
  - Cache and memories
  - IO

Questions

Questions ?