ASTERIOS for Space Systems

Erwan Coz (KRONO-SAFE)

December 3, 2019

ASTERIOS

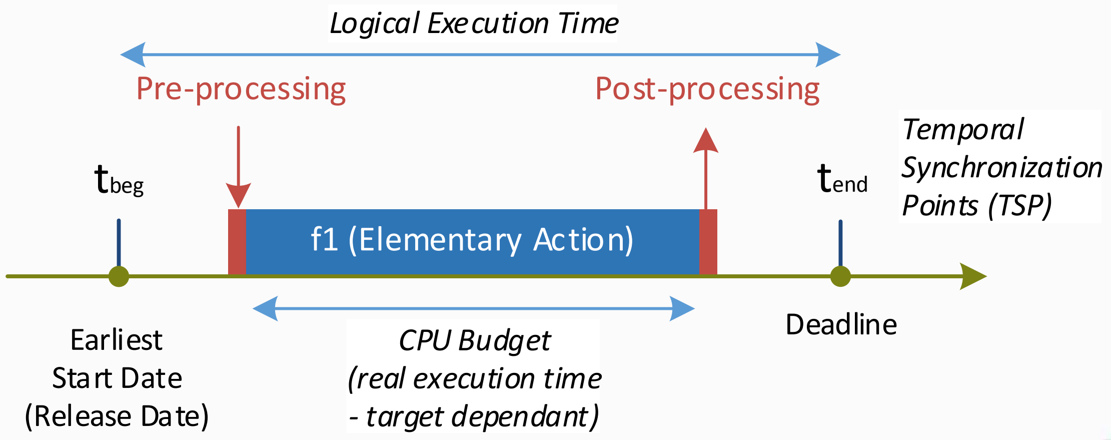
Project Overview

- Schedule: 6 months (actual 1 year)
- Purpose:
  - Demonstrate feasibility of safety critical applications with ASTERIOS development approach
  - Evaluate the impact of the ASTERIOS development process of safety critical applications for space systems
- Prime contractor:
  - KRONO-SAFE
- Work performed:
  - Analysis and selection of the software application to be ported on ASTERIOS
  - Porting of the selected software application on ASTERIOS
  - Demonstration and evaluation of the performance of the ported software application
  - Evaluation of the impact of the ASTERIOS development process on the space standard ECSS-E-ST-40C
  - Evaluation of the effort of qualifying ASTERIOS with respect to ECSS standard

KRONO-SAFE
Safe design in real-time

Time-Triggered Paradigm

- Two instants bound a processing (an Elementary Action)
  - the execution early start date
  - the deadline
- Behaviors are synchronized from those instants (Temporal Synchronization Points)
  - Task activation only set by the current date
  - No lock/semaphore/mutex nightmare for synchronization: no deadlocks
- Instants source can be:
  - A timer source, i.e. an hardware timer
  - An external events source, i.e. Ethernet frame reception, wheel tooth, engine's crankshaft



Logical Execution Time
Pre-processing
Post-processing
t_{beg}
t_{end}
Temporal Synchronization Points (TSP)
f1 (Elementary Action)
Earliest Start Date (Release Date)
CPU Budget (real execution time - target dependant)
Deadline

ASTERIOS main principles

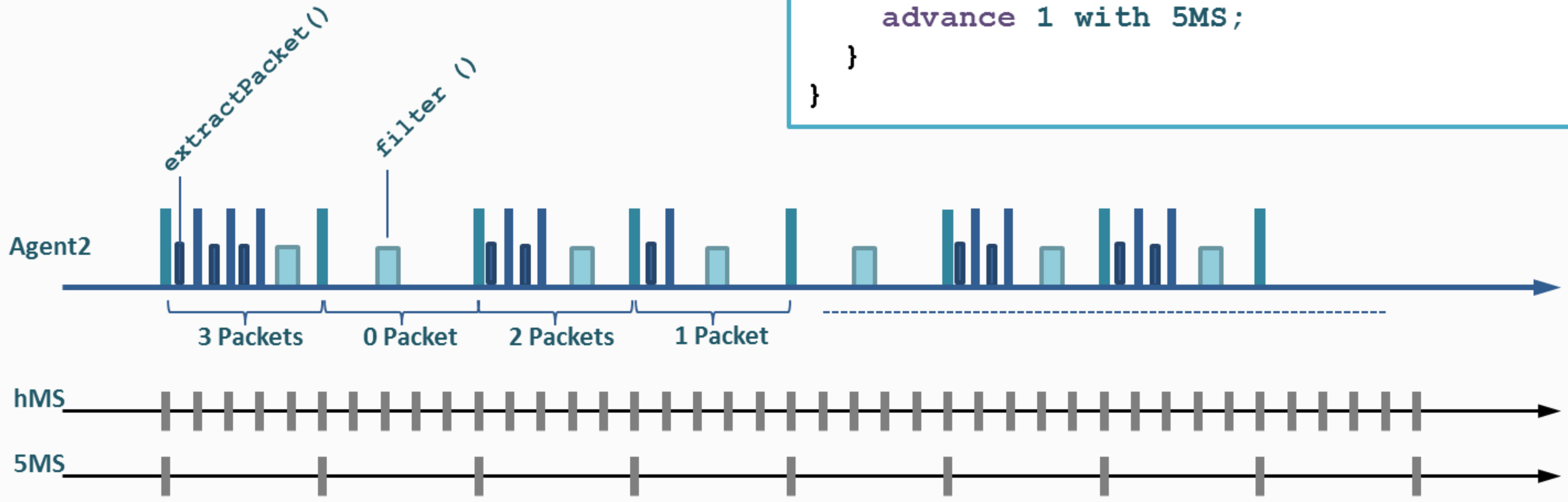
- **Formal & flexible design (PsyC)**
  - **Timing, partition** and **dataflow** description (periodic and aperiodic)

```
source realtime;

clock hMS = 1000 * realtime;
clock 5MS = 5 * hMS;

agent agent2 ( starttime=1 ) with hMS
{
   ...
```

```
  ...
  body start
  {
    while[3]( isReceivedPacket()==TRUE )
    {
       extractPacket();
       advance 1;
    }
    filter ();
    advance 1 with 5MS;
  }
}
```

Psy instructions

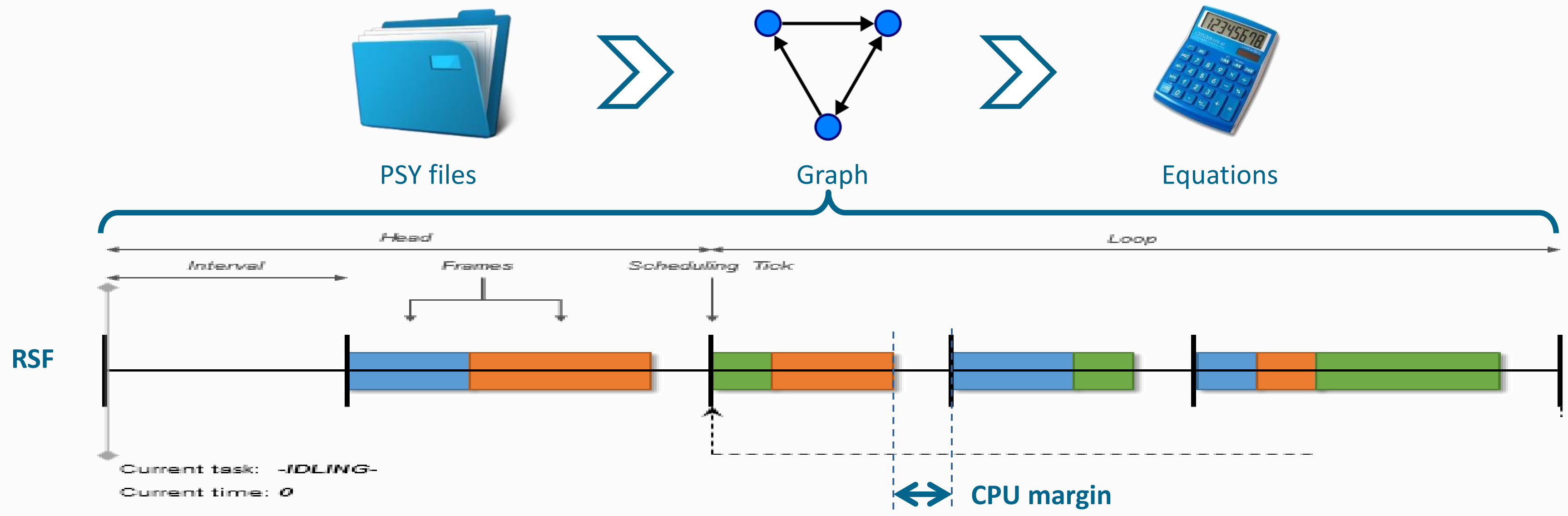
C functions



KRONO-SAFE

ASTERIOS main principles

- **Automated integration**
  - **Runtime tables** off-line computation
    - Based on PsyC program and timing budgets, ASTERIOS tool chain is able to generate:
      - the optimal scheduling for each core (RSF = Repetitive Sequence of Frame)
      - the memory partitioning configuration for the tasks and the kernel
      - the optimal configuration of the communication buffers
    - During the execution on the hardware target, the RTK:
      - enforces both spatial and temporal isolation between tasks according to the runtime tables
      - implements the hardware initialization, the error management and health monitoring features



Issue for determinism

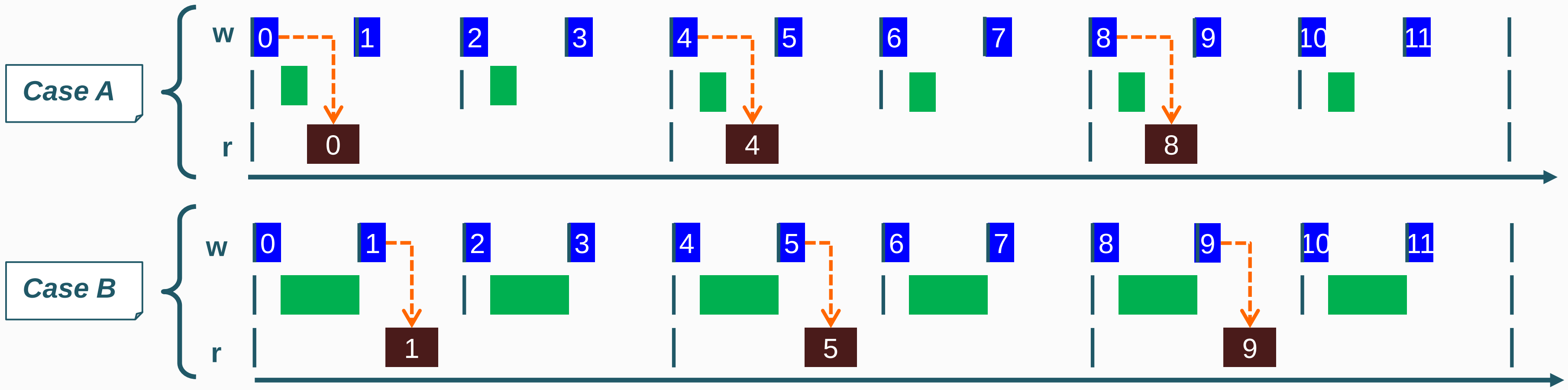
Immediate communication mechanism

- Asynchronism effect
  - **Non-deterministic behavior:** results of cases A and B depend on the scheduling, so they may depend on small variation of any execution duration

Writer: 🟦
```
static int c = 0;
output = c++;
```

Reader: 🟫
```
int x = input;
// use x
```

Other: 🟩



This problem already exists with single-core and is worst with multi-core

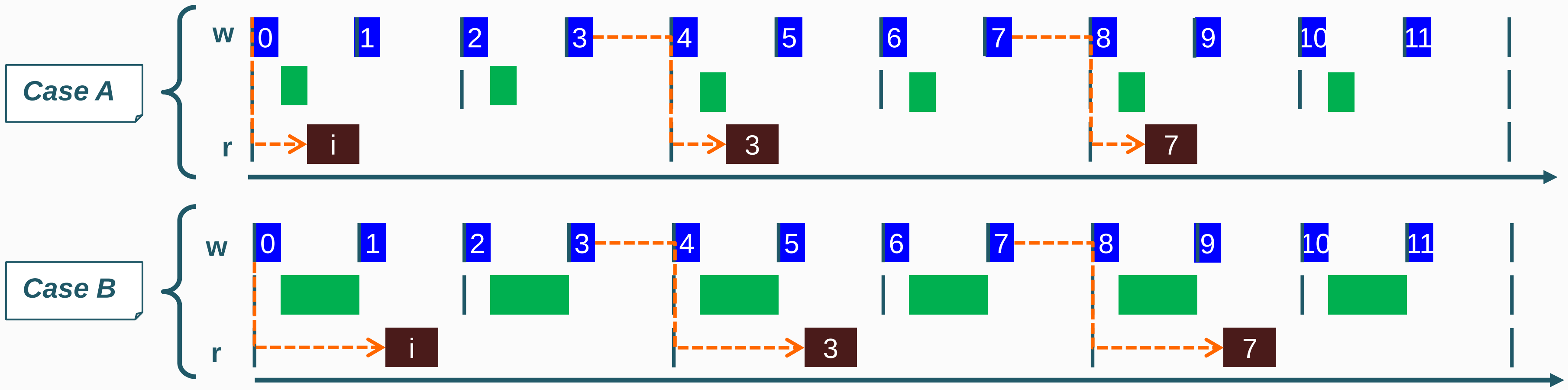
Solution for determinism

- **Synchronized communication mechanism (LET)**
  - No asynchronism effect
    - **Deterministic behavior:** results of cases A and B don't depend on the scheduling thanks to a time stamping of dataflow

Writer: 🟦
```
static int c = 0;
output = c++;
```

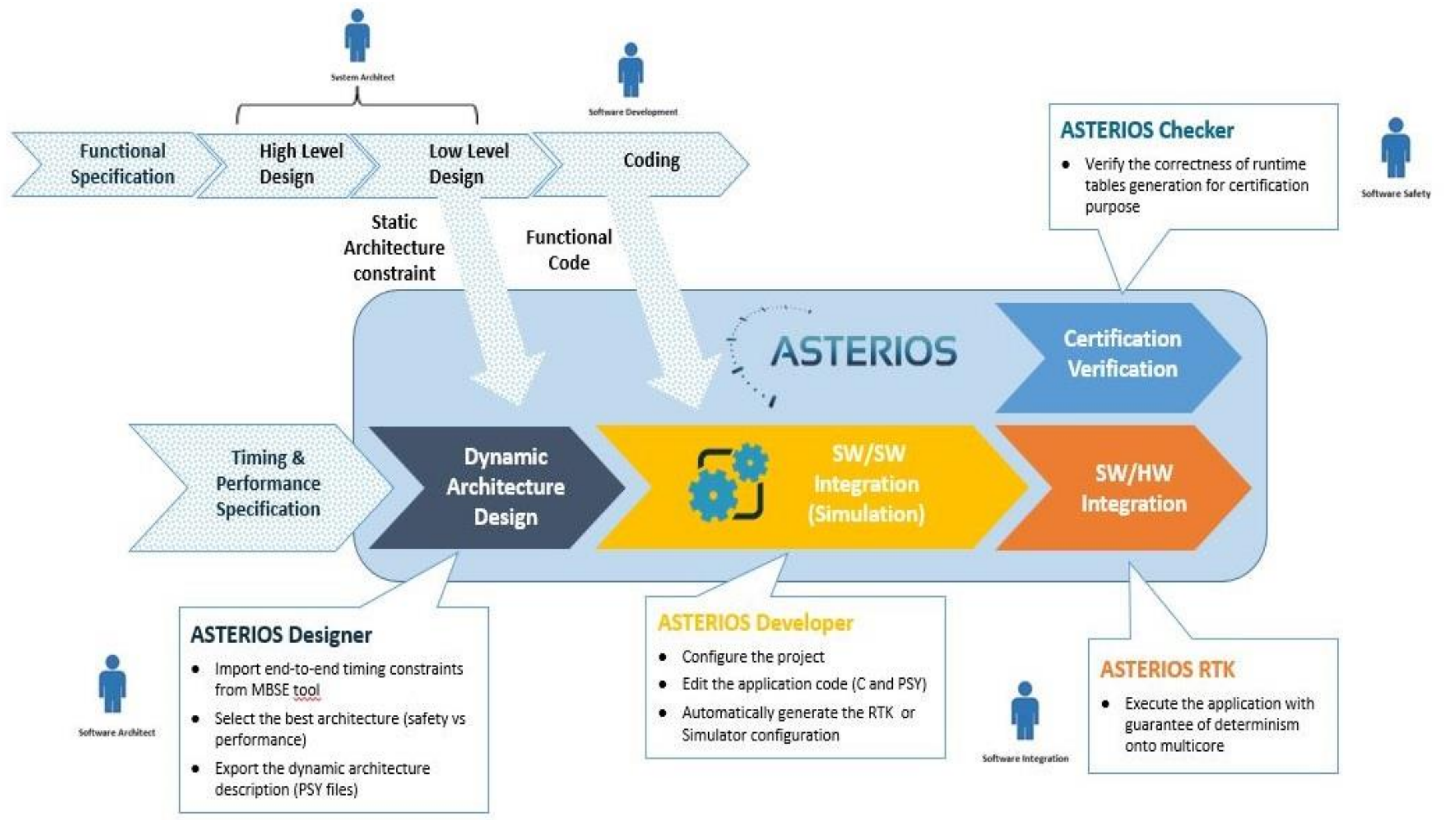
Reader: 🟫
```
int x = input;
// use x
```

Other: 🟩



✓ **This solution is operational with single-core and multi-core as well**

ASTERIOS Tool Suite Flow



ASTERIOS Checker

- Verify the correctness of runtime tables generation for certification purpose

ASTERIOS Designer

- Import end-to-end timing constraints from MBSE tool
- Select the best architecture (safety vs performance)
- Export the dynamic architecture description (PSY files)

ASTERIOS Developer

- Configure the project
- Edit the application code (C and PSY)
- Automatically generate the RTK or Simulator configuration

ASTERIOS RTK

- Execute the application with guarantee of determinism onto multicore

Work Logic

- Define criteria to select application Software to port
- Apply criteria to Eagle Eye and Global Navigation Satellite Systems Software-Defined Receiver (GNSS-SDR) applications => Eagle Eye application selection
- Design a dynamic architecture of the Eagle Eye application for ASTERIOS taking as basis the Eagle Eye application running on ARM on top of XtratuM hypervisor which partitions are executing RTEMS RTOS tasks or bare metal software
- Validate this dynamic architecture and perform SW/SW integration with the help of ASTERIOS simulator on a PC
- Perform SW/HW integration of this dynamic architecture to run on top of ASTERIOS RTK (Real Time Kernel) for ARM based board (Apalis board with i.MX6 quad cortex A9 SOC)
- Validate ported Eagle Eye application on top of ASTERIOS and Apalis ARM based board with IO data transferred through UDP over Ethernet to the test environment
- Analyze the impact of the ASTERIOS development process on the space standard ECSS-E-ST-40C

Application to port selection

- Eagle Eye and Global Navigation Satellite Systems Software-Defined Receiver (GNSS-SDR) applications are candidate to a port on top of ASTERIOS
- Criteria used for selection:
  - Programming language : ASTERIOS toolchain is supporting C and assembly language in its current version. Using another language (e.g ADA) is possible but requires significant work effort.
  - Dependency with external libraries: ASTERIOS RTK is not providing built-in libraries, any needed libraries will be integrated inside the application
  - Current execution environment (HW/SW): should be close to targeted environment
  - Hard real time requirements: ASTERIOS technology well suited for hard real time applications
  - Space and time partitioning requirements: Native support in ASTERIOS
  - Test environment reuse: Minimal effort to reuse an existing test environment with the targeted HW/SW (standalone board with Ethernet and serial connections)

Application to port selection

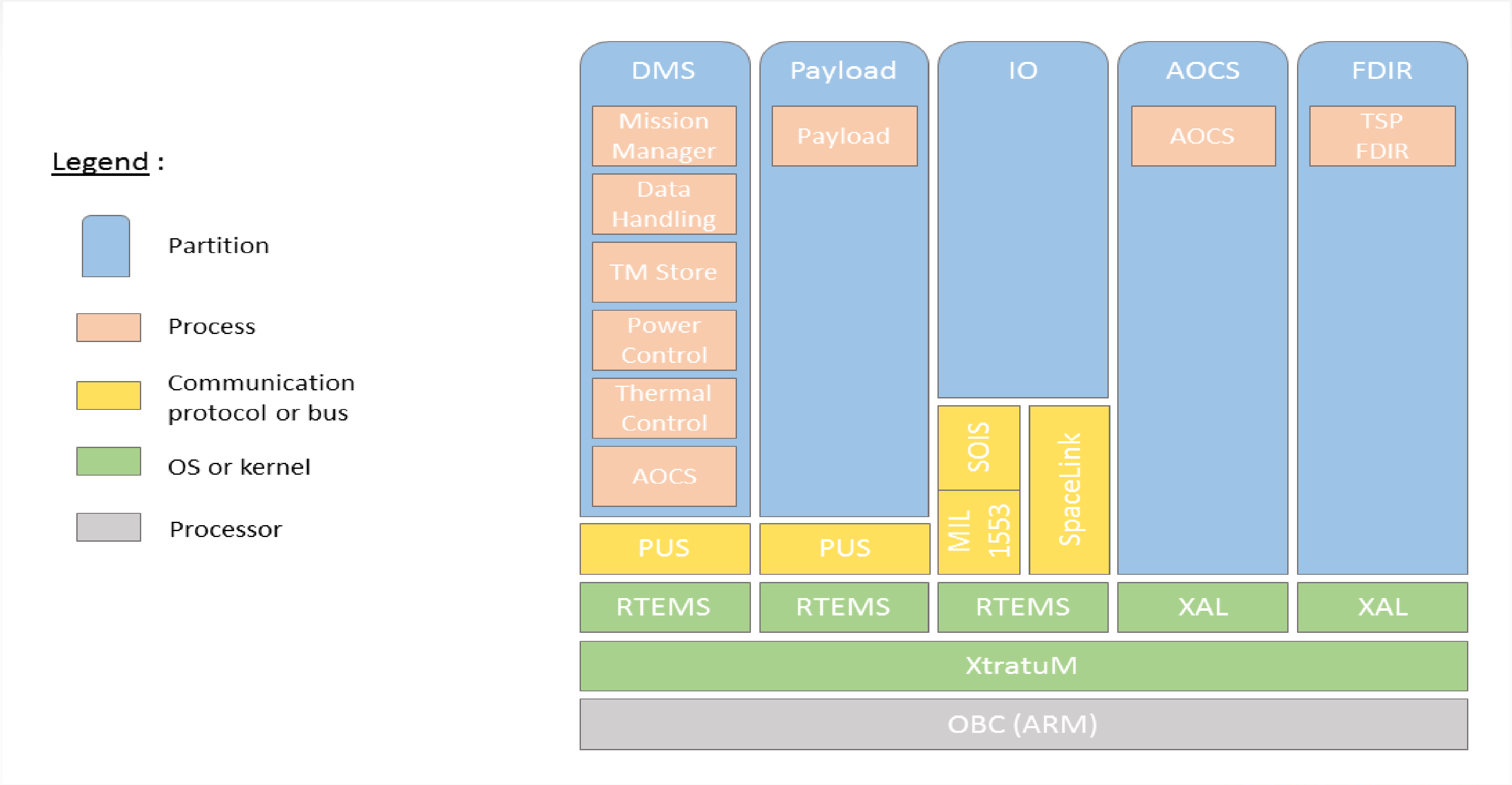
- Decision Matrix => Eagle Eye application selected

Criteria	Eagle Eye	GNSS-SDR
Programming Language compatibility with ASTERIOS	C fully compatible	C++ not compatible
Dependency with external libraries	None	Dependency with a number GNU/Linux libraries and packages
Current execution environment (software, hardware)	XtratuM ARINC 653 hypervisor and RTEMS RTOS	Linux OS, Windows, Mac OS
Need for real-time	Yes, control/command application	Yes, need to process signal in real time
Need for space and time partitioning	Yes, application is implementing 5 partitions running on different time slots	No
Test environment reuse	High: Possibility to reuse in standalone mode via UDP communication the existing test environment in order to exercise application Input/Outputs (Space link TM/TC) and to emulate sensors/actuators (MIL1553 bus)	Low reuse

KRONO-SAFE
Safe design in real-time

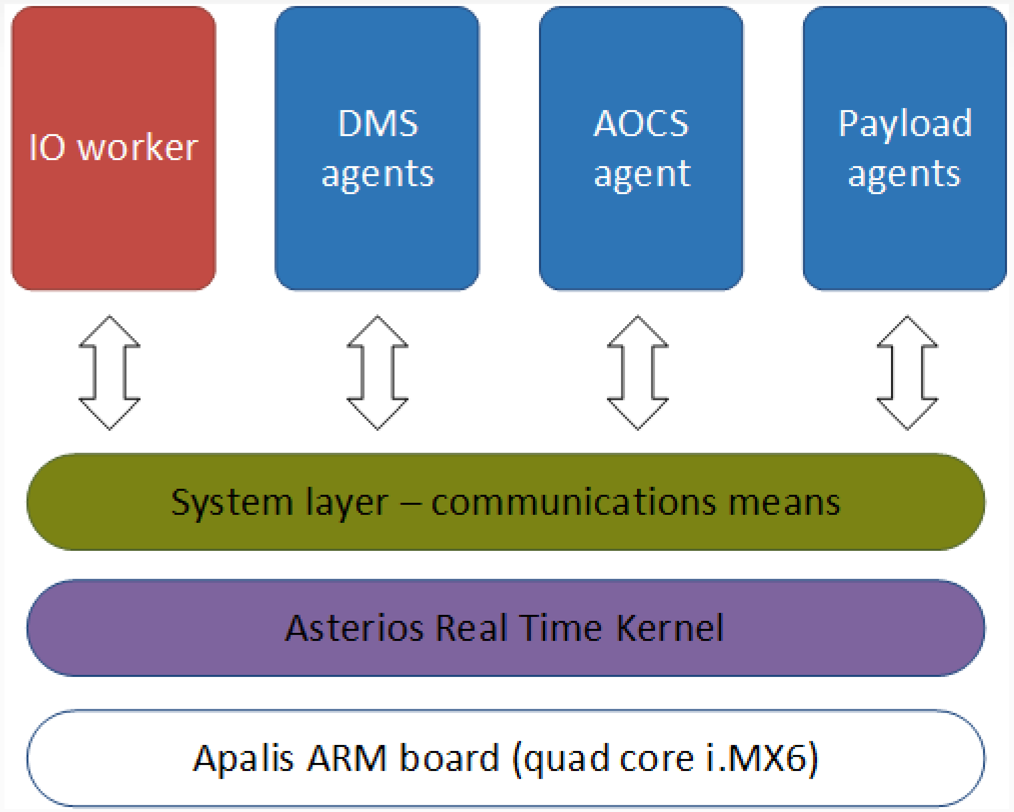
Eagle Eye port starting point

- Eagle Eye version 7 Central Software (Xtratum HV + RTEMS RTOS) running on TEMU emulator platform
- Starting point architecture



Eagle Eye port on ASTERIOS

- Eagle Eye version 7 Central Software integrated in ASTERIOS application (agents, workers) running on ARM based board
- Targeted architecture


IO worker
DMS agents
AOCS agent
Payload agents
System layer – communications means
Asterios Real Time Kernel
Apalis ARM board (quad core i.MX6)

Eagle Eye port strategy

- Preserve spatial functions partitioning (DMS, Payload, IO, AOCS)
- Remove FDIR partition managing health monitoring and error management since this can be handled by ASTERIOS native error management framework which is able to detect and manage errors (HW exceptions and time execution monitoring)
- Preserve Xtratum IPC mechanism: ARINC 653 queueing and sampling ports are replaced by ASTERIOS communication mechanisms (temporal variables and streams)
- Define partitions functions blocks and their inputs/outputs
- Preserve temporal scheduling of the partitions:
  - Major frame = 250 ms
  - IO, DMS, AOCS, Payload execution time slots preserved
  - Build functional temporal constraints based on RTEMs task frequencies
- Build ASTERIOS dynamic architecture based on the above assumptions

  => functions allocations into execution units (agent, worker)

  => communications means between execution units

  => describe timing constraints with PSY C language

Eagle Eye dynamic architecture

- First step : function units layout build
  - Function units identification (from initial application tasks/processes)
  - Identification of data flow between function units
  - Input and output data structures of each function unit definition/identification
  - Relationship between inputs and outputs of all functional units

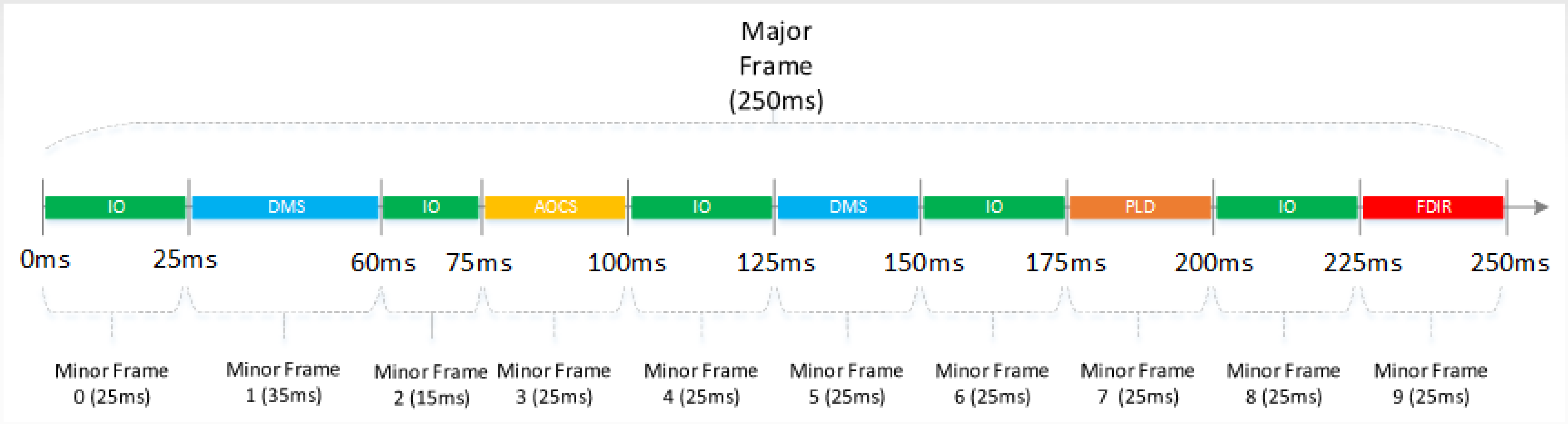
- Second step: functions allocation to execution units
  - Allocate functions into ASTERIOS execution units (agents for functional code, workers for driver code)
  - Define ASTERIOS communication means based on functions units data flow defined on step 1 for input/output data exchanges between execution units
  - Use local variables for communications of functions which are allocated inside the same execution unit

Eagle Eye dynamic architecture

Functional Unit	Application Process(es)	Input(s)	Output(s)
AOCS partition			
AOCS software	AOCS (AOCS partition)	AOCS commands, AOCS software new inputs	AOCS software outputs, AOCS software current inputs
Payload partition			
Payload-specific TC routing	-	Payload TC	Decoded Payload TC, PUS1 TM, Event
PUS8 TC execution (specific to Payload)	Payload	PUS8 TC	Payload commands, PUS1 TM, Event
IO partition			
SpaceLink switching (SpaceLink)	-	SpaceLink control commands	SpaceLink control responses
TC receiving (SpaceLink)	-	Ethernet packet	TC
TM sending (SpaceLink)	-	TM	Ethernet packet
Data receiving (MIL bus)	-	External IO (Ethernet)	Sensor readings, Actuator readings, Temperature readings, Heater states, Power load values, Power states
Sensor commands sending (MIL bus)	-	Sensor commands	External IO (Ethernet)
Actuator commands sending (MIL bus)	-	Actuator commands	External IO (Ethernet)
Power command sending (MIL bus)	-	Power command	External IO (Ethernet)
Payload command sending (MIL bus)	-	Payload command	External IO (Ethernet)

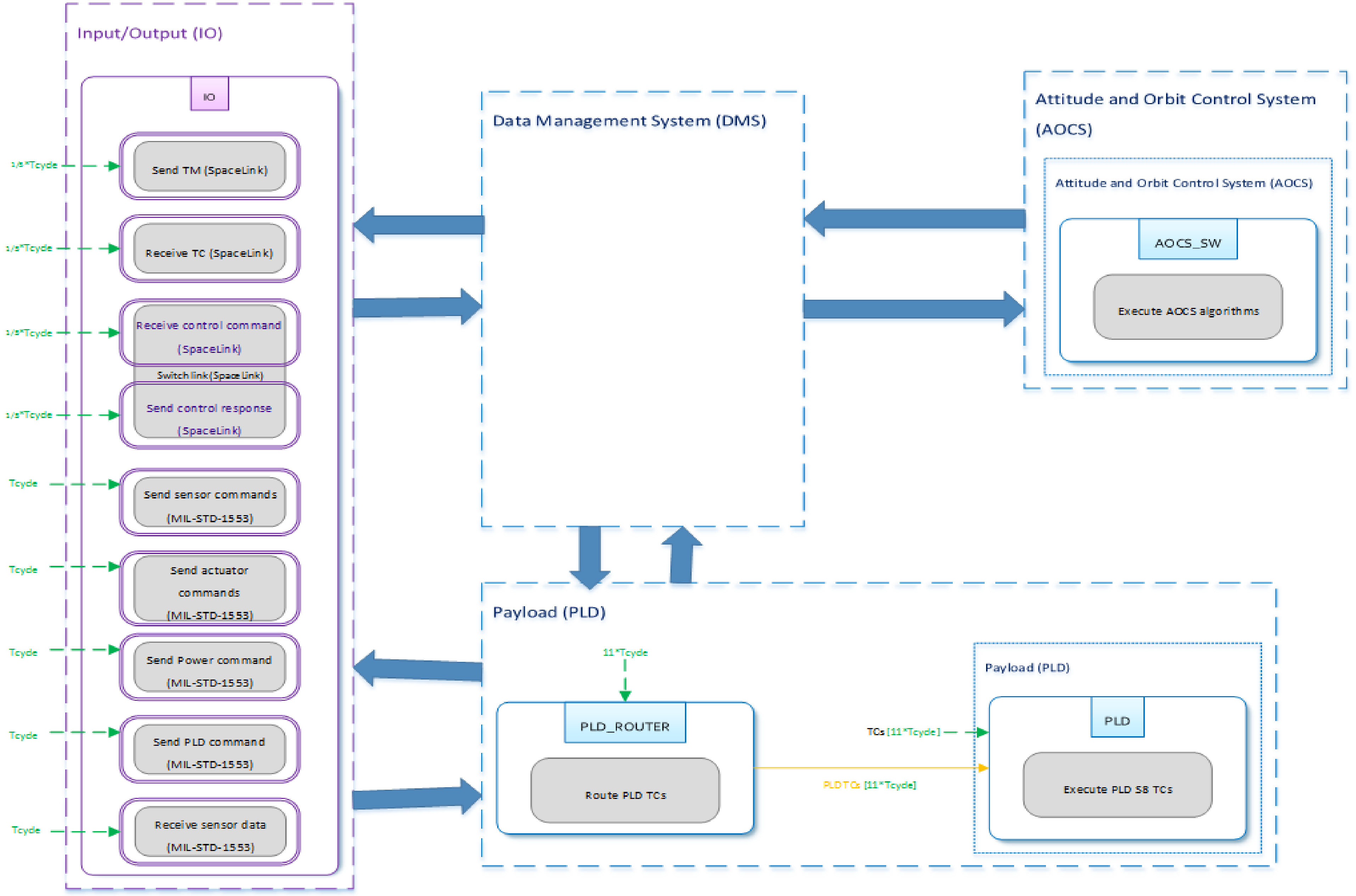
Eagle Eye dynamic architecture

- Third step : define function and execution units timing constraints
  - Respect below partition cadencing

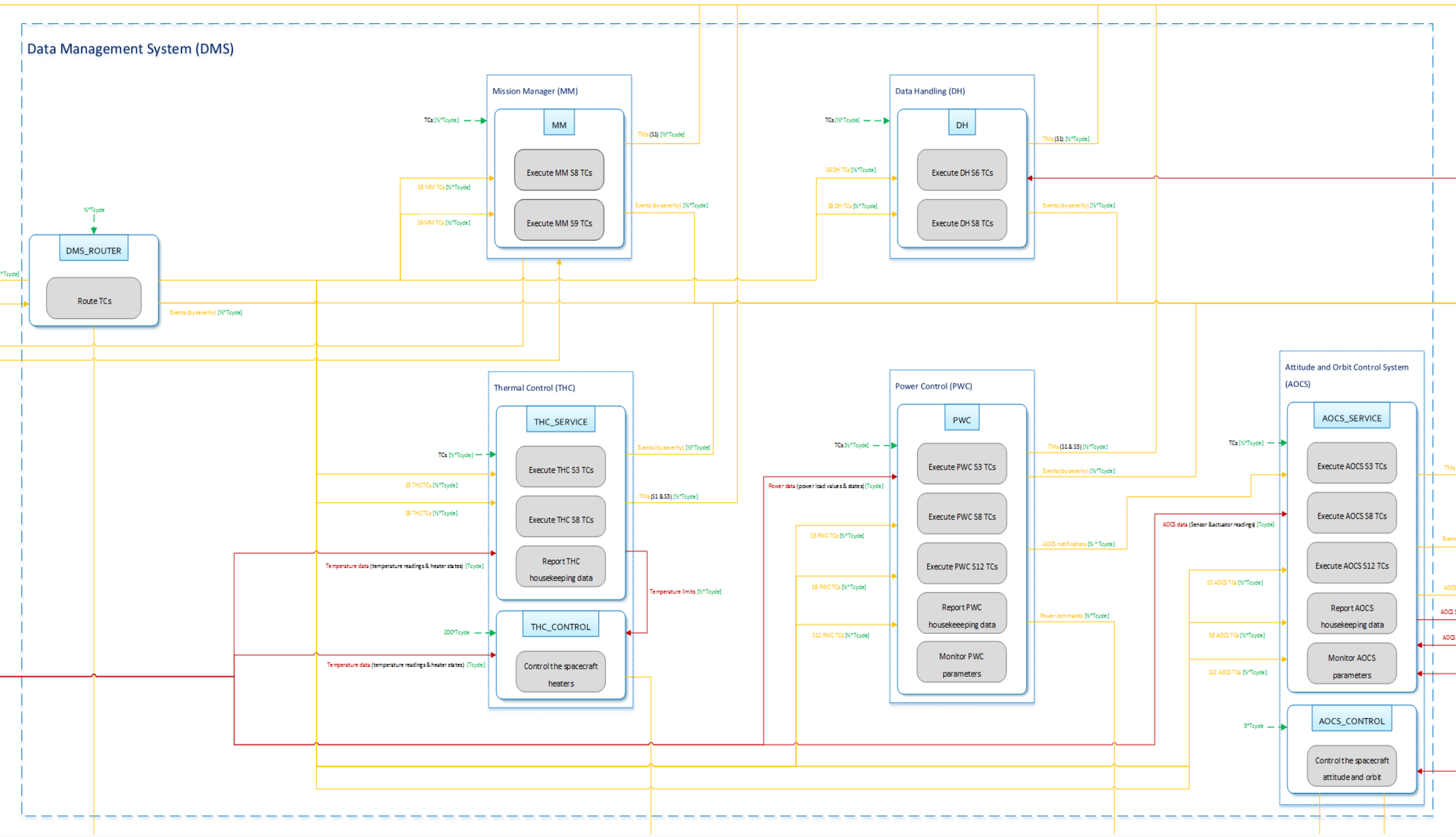


  - Define temporal constraints of different functions based on RTEMS task periods
    - Global TC routing happens 2 times per cycle of 250 ms
    - TC execution (except for Payload-related) happen every time a TC is received, 2 times per cycle
    - Housekeeping data reporting, parameter monitoring, is done every 20 cycles
    - Event reporting happens 5 times per cycle
    - Spacecraft attitude and orbit control is done every 5 cycles
    - Spacecraft thermal control is done every 200 cycles
    - Payload TC routing, as well as Payload TC execution, happen every 11 cycles
    - All Space Link related functional units can be executed 5 times per cycle
    - All MIL bus-related functional units can be executed once a cycle

Eagle Eye dynamic architecture

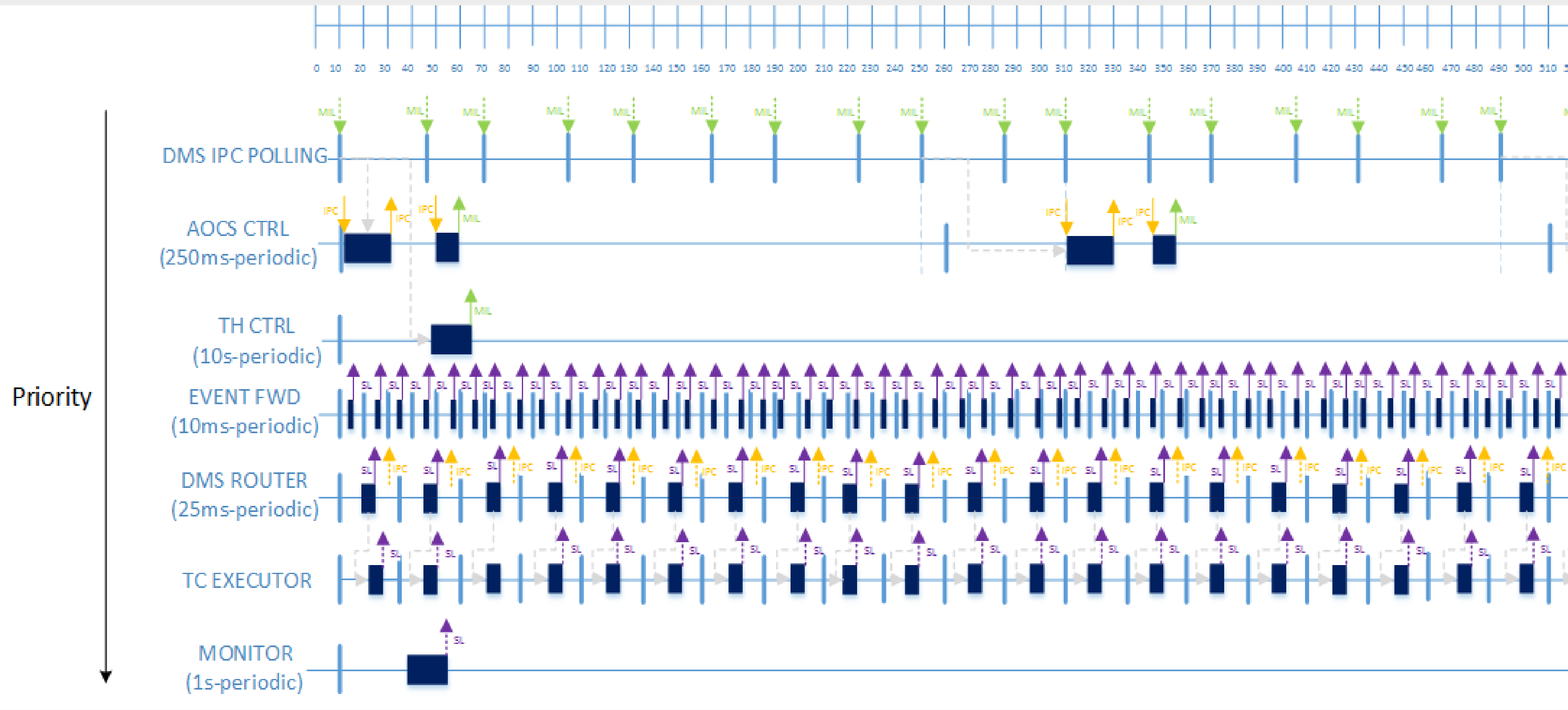

Input/Output (IO)
IO
1/8*Tcycle
Send TM (SpaceLink)
1/8*Tcycle
Receive TC (SpaceLink)
1/8*Tcycle
Receive control command (SpaceLink)
Switch link (Space link)
1/8*Tcycle
Send control response (SpaceLink)
Tcycle
Send sensor commands (MIL-STD-1553)
Tcycle
Send actuator commands (MIL-STD-1553)
Tcycle
Send Power command (MIL-STD-1553)
Tcycle
Send PLD command (MIL-STD-1553)
Tcycle
Receive sensor data (MIL-STD-1553)
Data Management System (DMS)
Attitude and Orbit Control System (AOCS)
Attitude and Orbit Control System (AOCS)
AOCS_SW
Execute AOCS algorithms
Payload (PLD)
11*Tcycle
PLD_ROUTER
Route PLD TCs
TCs [11*Tcycle]
PLD TCs [11*Tcycle]
Payload (PLD)
PLD
Execute PLD S8 TCs

Eagle Eye dynamic architecture

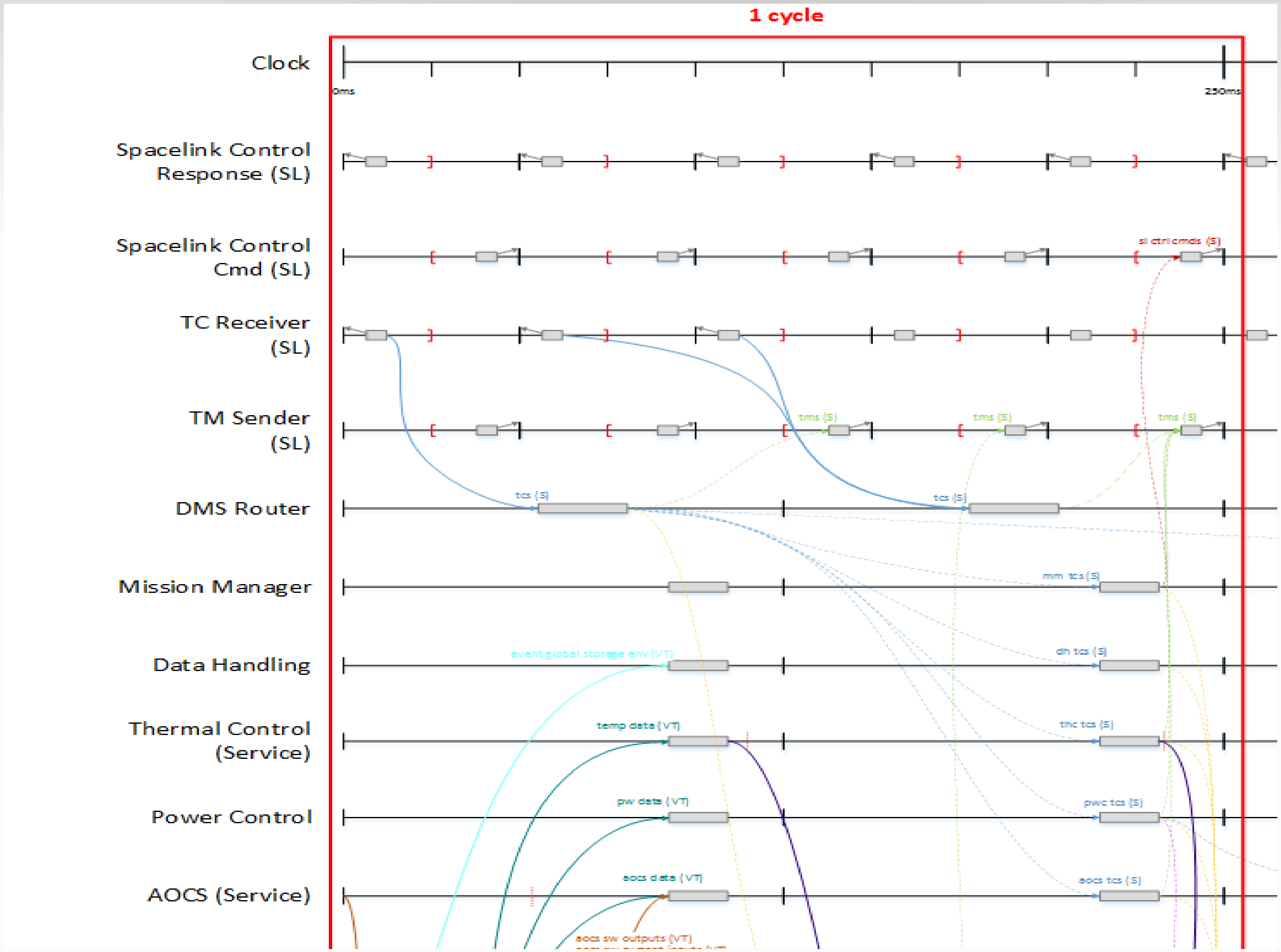


Eagle Eye dynamic architecture

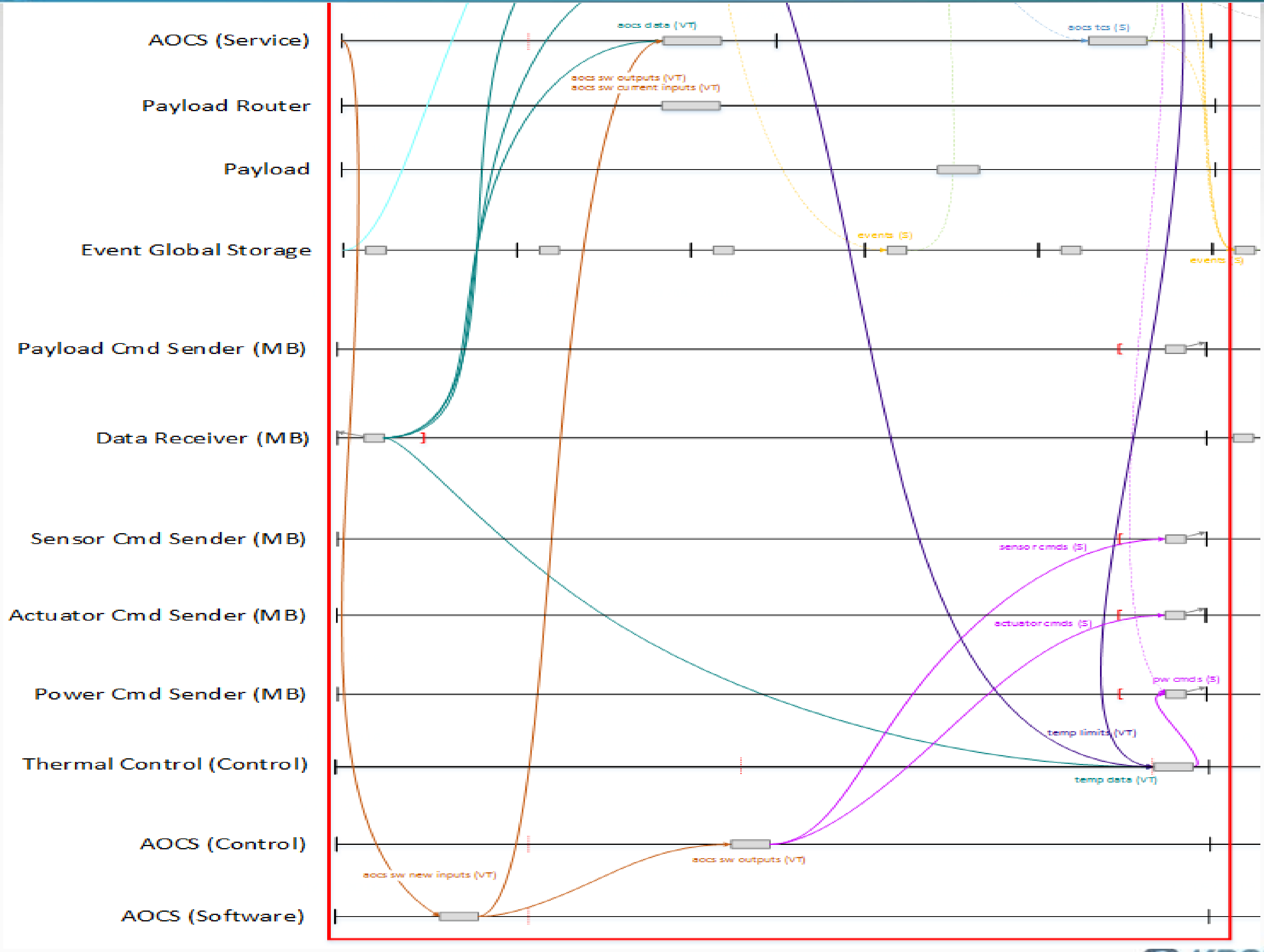
- DMS cadence view



Eagle Eye dynamic architecture



Eagle Eye dynamic architecture



KRONO-SAFE
Safe design in real-time

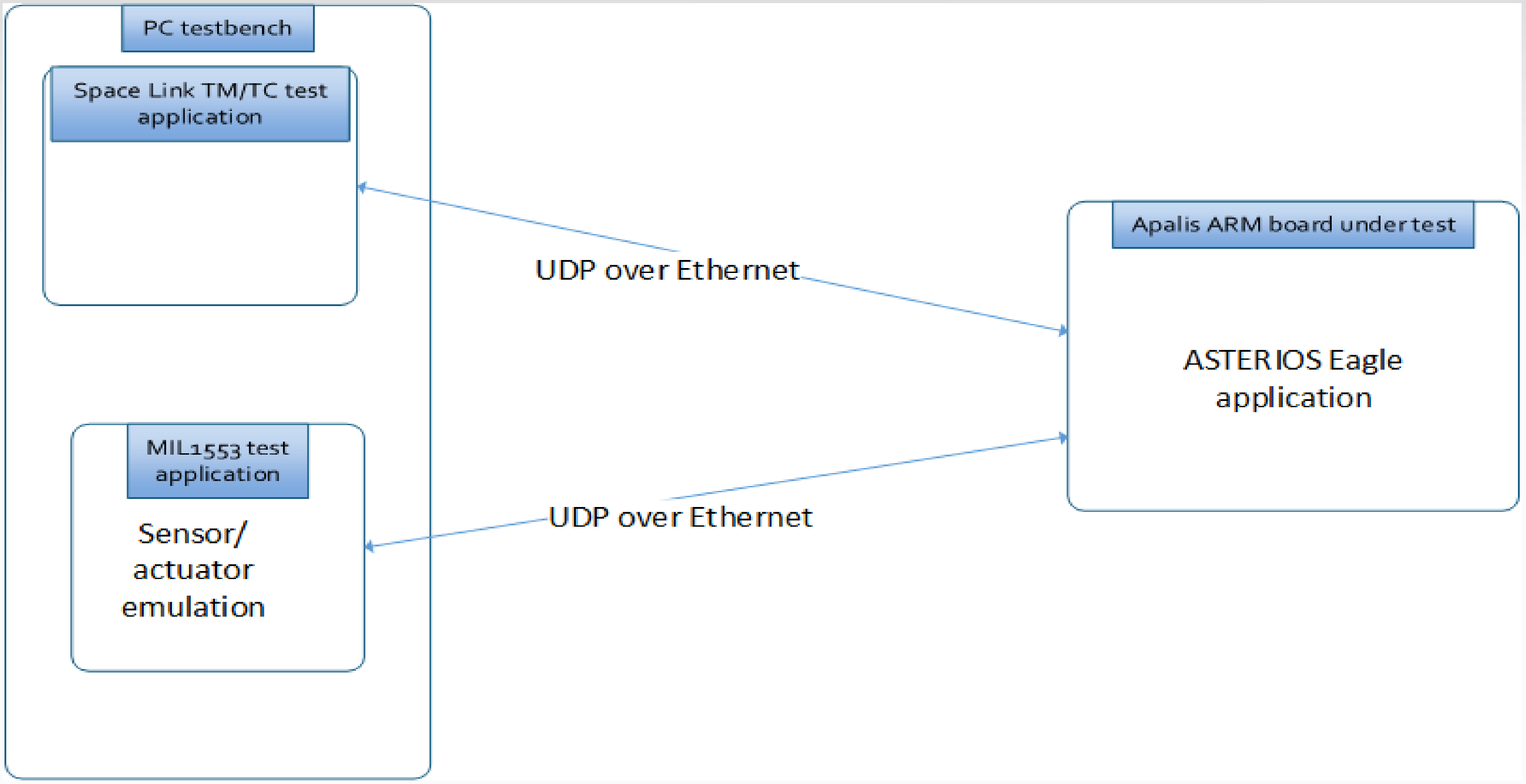
SW/SW integration

- Dynamic architecture formally expressed with PSY language :
  - Execution units (agents, workers)
  - Sub execution unit (body, jobs)
  - Cadence (clocks, cycles, jitter, advance)
  - Communications mechanisms (temporal variables, streams)
- Code integration
  - C Source code of Eagle Eye v7 for ARM taken as basis
  - Remove all code related to Xtratum (APIs related to partitions, IPC)
  - Remove all code related to RTEMS (tasks, mutex) even the one inside libraries (libpus, gnc)
  - Rewrite PUS services media management to be integrated directly in agents
  - Add standard libraries (needed part of C library, mathematical library) from used GCC compiler
  - Remove FDIR partitions and replace it with ASTERIOS error management framework
- Test with ASTERIOS simulator
  - Use MingGW GCC version 8.1.0 for Windows as C compiler and C linker
  - Build and execute Eagle Eye application for simulator
  - Verify temporal behavior and data flow with the help of generated timing diagram

SW/HW integration

- Psy design not impacted
- Configure application build for HW
  - Setup for RTK Apalis board embedding ARM cortex A9 quad core i.MX6 SOC from NXP
  - Use arm-none-eabi-none GCC version 4.9.3 for Windows as C cross-compiler and C linker
  - Use standard libraries (needed part of C library, mathematical library) from used GCC cross-compiler toolchain
- C files depending on HW (drivers) changed at compilation time : add specific Ethernet driver to exchange IO data with external test applications
- Peripheral access right configuration (Ethernet controller access for IO worker)
- Agent/worker to core allocation based on budget execution time requirements
- Budget configuration refining based on profiling execution and comparison with requirements
- Validate application sizing (scheduling and memory protection)

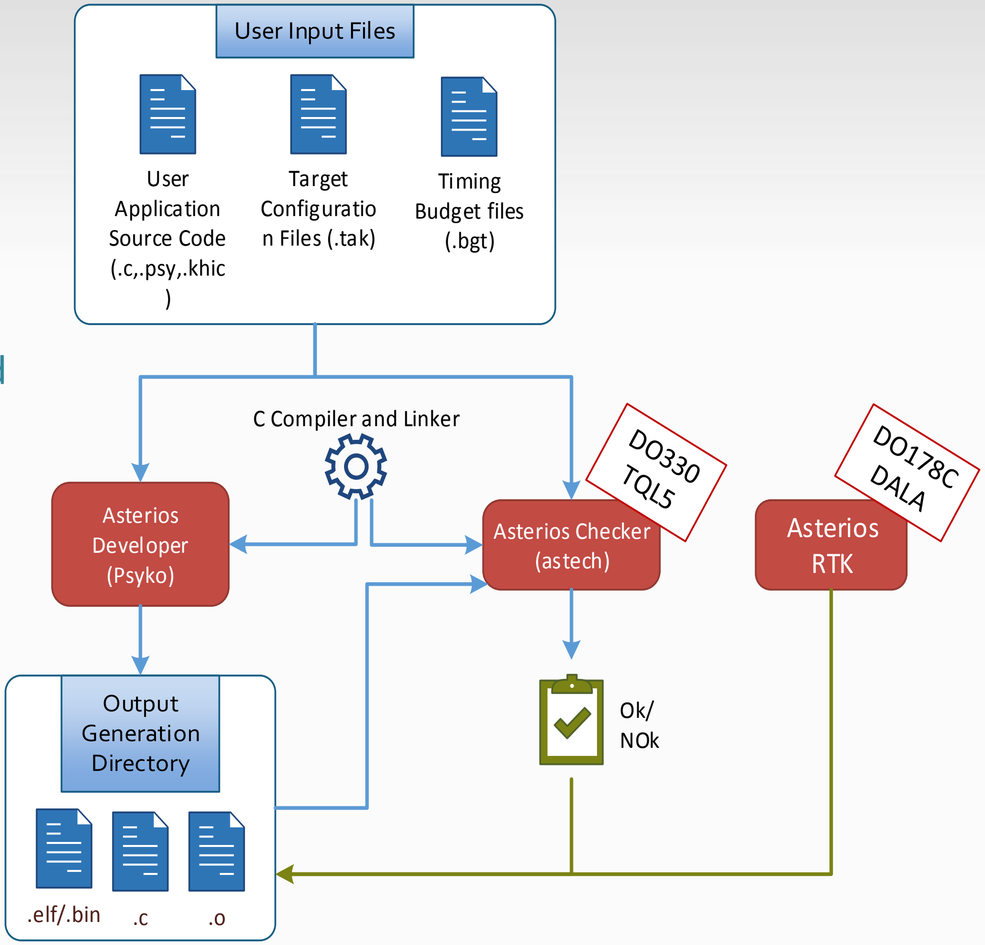
Test environment


PC testbench
Space Link TM/TC test application
MIL1553 test application
Sensor/ actuator emulation
UDP over Ethernet
UDP over Ethernet
Apalis ARM board under test
ASTERIOS Eagle application

- Leverage existing SVF environment for Eagle Eye for Space link TM/TC and MIL1553 sensor/actuator test applications

Qualification

- Aeronautic Domain
  - Target DO178C-DALA
- ASTERIOS RTK: DO-178C
  - Target : DAL A
- ASTERIOS Developer Tool chain
  - Strategy: Tool Chain outputs checked by an independent tool chain
  - Target : QA
- ASTERIOS Checker
  - Target : DO330 TQL5
- Reuse this approach for ECSS-E-ST-40C qualification process (criticality level A)


User Input Files
User Application Source Code (.c,.psy,.khic)
Target Configuration Files (.tak)
Timing Budget files (.bgt)
C Compiler and Linker
Asterios Developer (Psyko)
Asterios Checker (astech)
DO330 TQL5
Asterios RTK
DO178C DALA
Output Generation Directory
.elf/.bin
.c
.o
Ok/ NOk

Qualification

- Main benefits of ASTERIOS usage for for ECSS-E-ST-40C qualification process
  - Formal description of the dynamic architecture (Psy language to express timing and partitioning requirements) of the space application
  - Automatic generation of runtime tables (scheduling and memory configuration) based on the dynamic architecture requirements and the unitary functions maximum execution times
  - Independent checker tool to validate automatically that the generation process of the runtime tables is correct (replacement of manual review/verification activities)
  - Generation of a deterministic by construction application due to time triggered concepts and data flow consistency
  - Reuse of artifacts/documents produced for DO-178B certification of ASTERIOS kernel for the application of ECSS-E-ST-40C development process for ASTERIOS kernel (considered as a COTS)

Lessons learned

- Dynamic architecture design built by reverse engineering of the Eagle Eye for ARM application current implementation (Xtratum partitions + RTEMS tasks)
  - ⇒ Using timing, partitions and data flow requirements (end to end functional constraints, system response time, IO physical capability..) as inputs to build the dynamic architecture is a better practice for a full benefit of ASTERIOS technology usage
- Learning curve and support needed to adapt test environment for the HW/SW setup of Eagle Eye application running with ASTERIOS on ARM target
  - Needs adaptations of SVF environment in order to exchange TM/TC data using UDP over UDP
  - Needs adaptations of SVF environment in order to emulate sensors/actuators data due to the lack of 1553 hardware on chosen ARM board.

KRONO-SAFE
Safe design in real-time

Conclusion

- Successful port of a representative control/command application (Eagle Eye) for a space system using ASTERIOS technology to run on a ARM reference board
- ASTERIOS technology allows to guarantee the determinism by construction of an application on single core or multicore ARM platforms
  - Logical Execution Time (Time Triggered applications)
  - Data flow consistency and determinism
  - A formal language to express parallelism, sequencing and synchronizations
  - Automatic generation of optimal scheduling and memory configuration from specifications
  - A performant real-time kernel (lock free/wait free, small footprint, constant time execution, native multicore)
- ASTERIOS technology have positive impact when developing applications according to space standards ECSS-E-ST-40C qualification
  - Qualification strategy with an automatic checker of the main toolchain
  - Reuse artifacts from DO-178C Level A/DO330 certification/qualification

Thank your for your attention…

…any questions ?

Erwan Coz – Senior FAE

Erwan.coz@krono-safe.com

www.krono-safe.com

KRONO-SAFE