

AOCS S/W Development Process for SWARM, Sentinel-2 and EarthCare (contribution to NAVVA workshop)

Stand: 10.09.2010

Astrium Satellites, ASG62
Domenico Reggio

All the space you need



Topics

- Introduction
- S/W Development & Specification
 - S/W Architecture
 - AOSE – AOCS Offline Simulator
 - Algorithm Development
 - Executable Specification
- Formal Verification
 - Reinjection of Flight Code

This document is the property of Astrium. It shall not be communicated to third parties without prior written agreement. Its content shall not be disclosed.

Introduction

■ Abstract

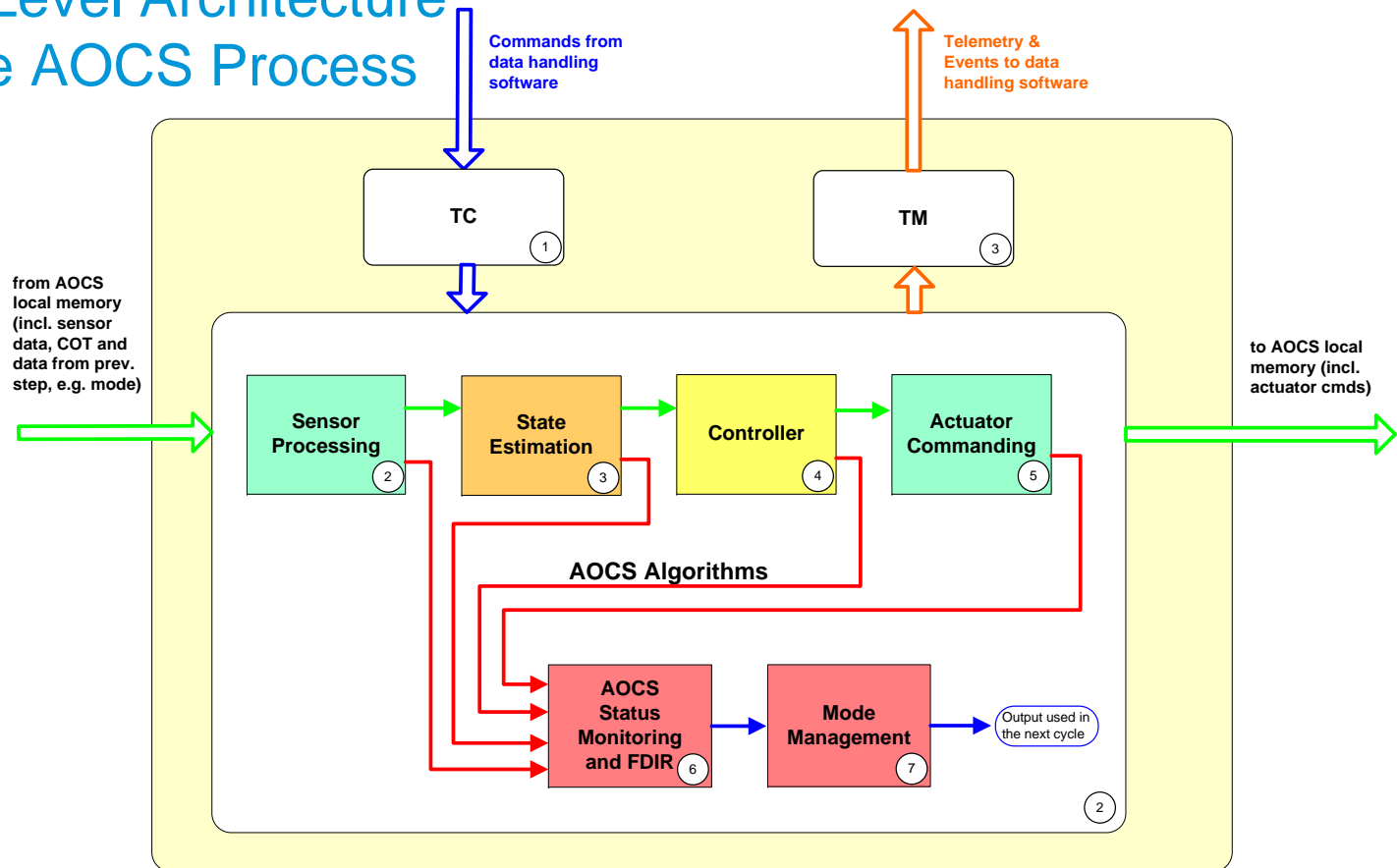
- Using the Matlab/Simulink environment and a dedicated AOCS offline simulator (AOSE) we are able to develop, define and test AOCS algorithms very early in the project lifetime. The algorithms specification is taken directly from the AOSE. After SW coding by the SW supplier the AOCS part of the overall onboard software is reinjected into the AOSE for performance verification. This all assures a lean development process.

■ Background Information

- Activities for „Detailed“ Algorithms Specification Start in middle of Phase B
- Big part of AOCS Offline Simulator (AOSE) already available from Phase A
- The process has been applied on SWARM and is applied on Sentinel-2 and EarthCare

AOCS S/W Architecture

■ Top Level Architecture of the AOCS Process



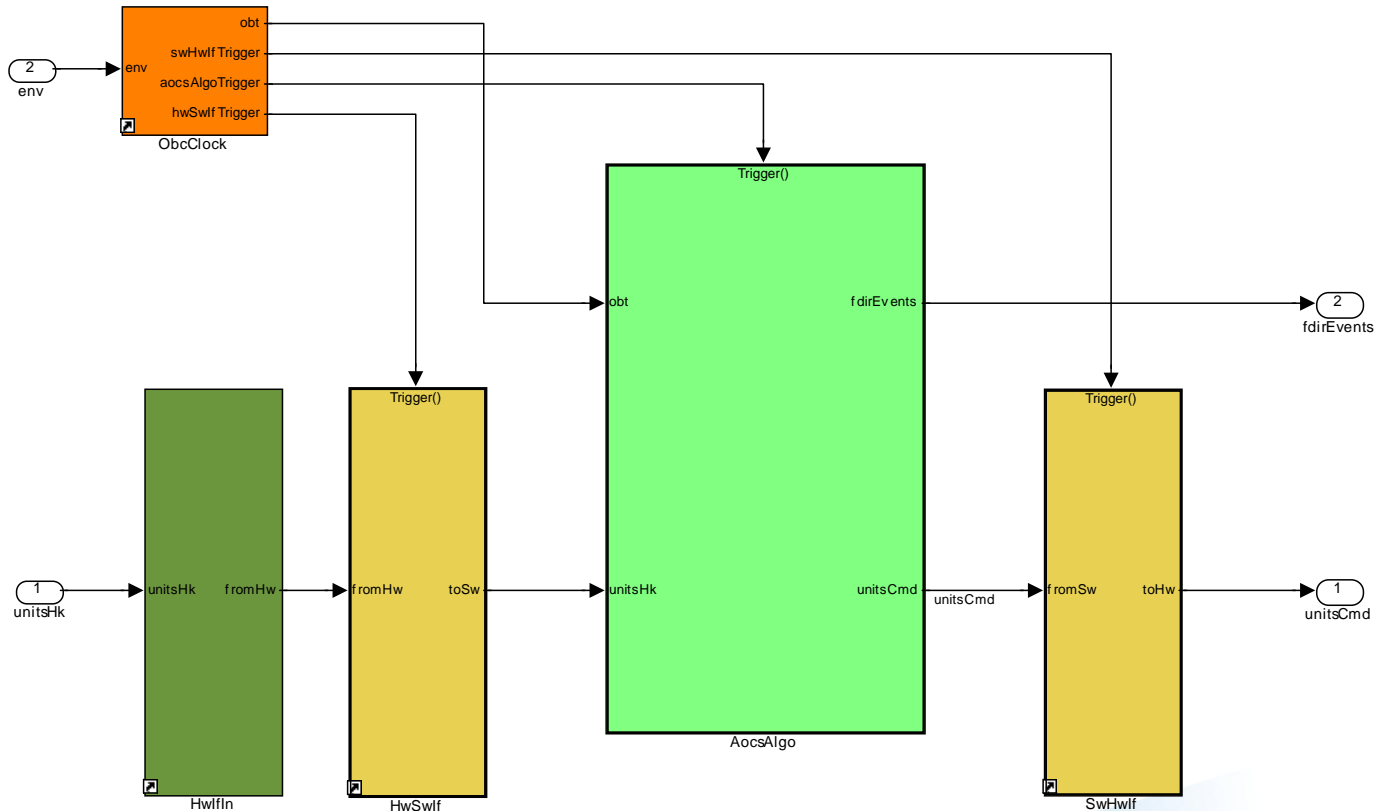
(In-house standard AOCS SW functional architecture)

AOSE – AOCS Offline Simulator

- The AOCS „Offline“ Simulator
 - Simulates
 - Physical environment around the S/C (Gravity, Magnetic Field, Eclipse, ...)
 - Unit models (all Sensors, Actuators, OBC and PCDU)
 - S/C itself (incl. geometry, mass properties, ...)
 - Development environment for
 - AOCS S/W Algorithms
 - Verification tasks
 - AOCS S/W unit tests
 - Open loop tests
 - Closed loop tests
 - Supports mission analysis tasks (e.g. simulation of calibration manoeuvres)
 - Based on STAR (In-house library for Dynamics and Environment Models)

AOSE – AOCS Offline Simulator (cont'd)

■ OBC



- Converts analog to digits
- selects the cold redundant units
- data corresponds to data pool
- OBT generation
- further proc. not included in AOCS Algo

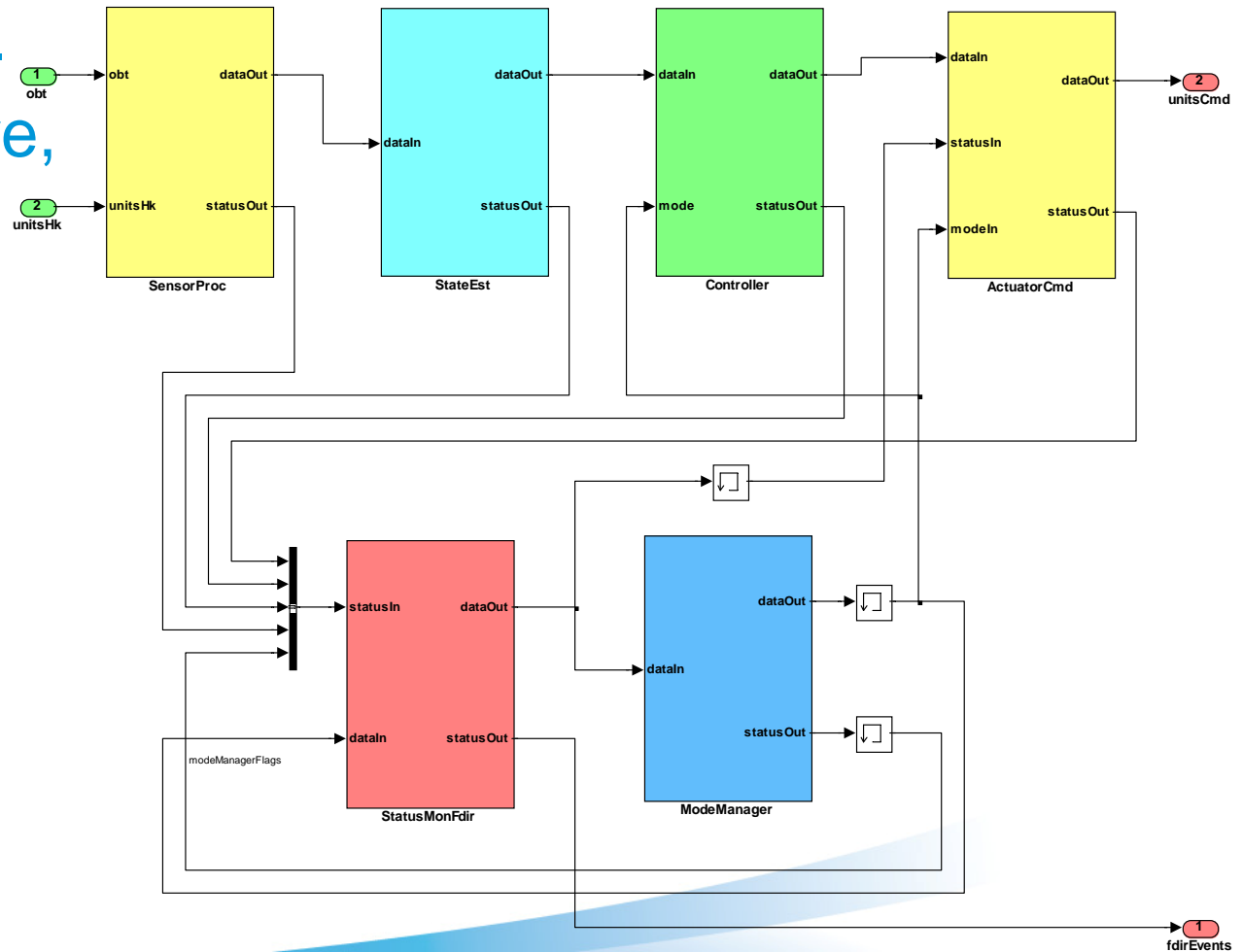
AOSE– AOCS Offline Simulator (cont'd)

■ AocsAlgo

- Block contains all S/W algorithms for AOCS
- Interfaces to outer world are minimised
- S/W architecture is split into levels 1-3
 - Level 1 – Top Level Architecture (project independant)
 - Level 2 – „Unit“ Level Architecture
 - Level 3 – „Function“ Level Architecture, connects „Building Blocks“
 - Architecture always implemented in Simulink
- „Buildings Blocks“ contain the algorithms
 - Big reuse of In-House verified „Building Blocks“ library, collected into „AOCS Functions“
 - Implemented in Embedded Matlab (EM)
- „Support Functions“
 - Support „Building Blocks“ with basic functionality
 - E.g. math library, filter, integrator, signal handling, ...

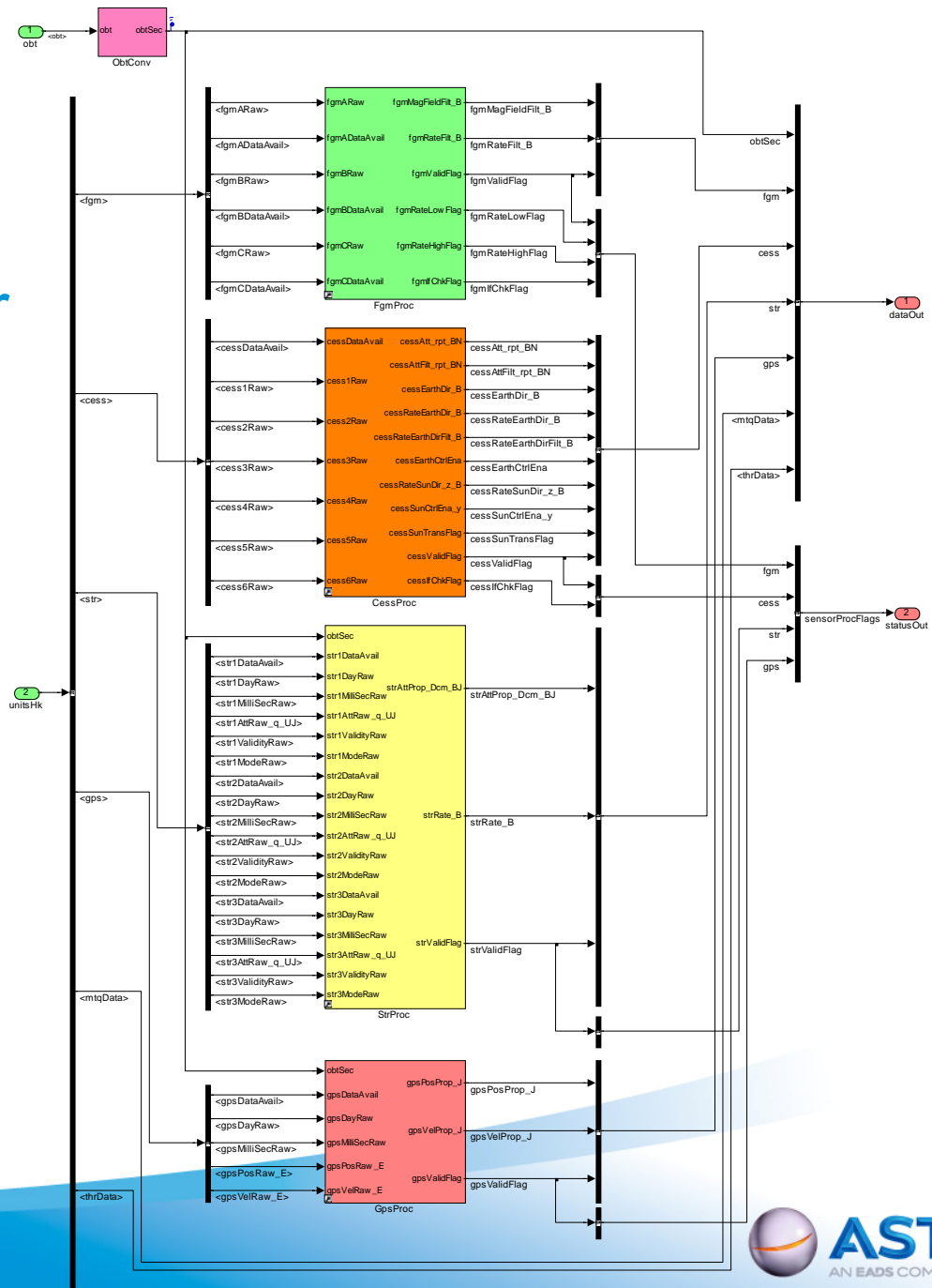
AOSE – AOCS Offline Simulator (cont'd)

■ AocsAlgo - Architecture, Level 1



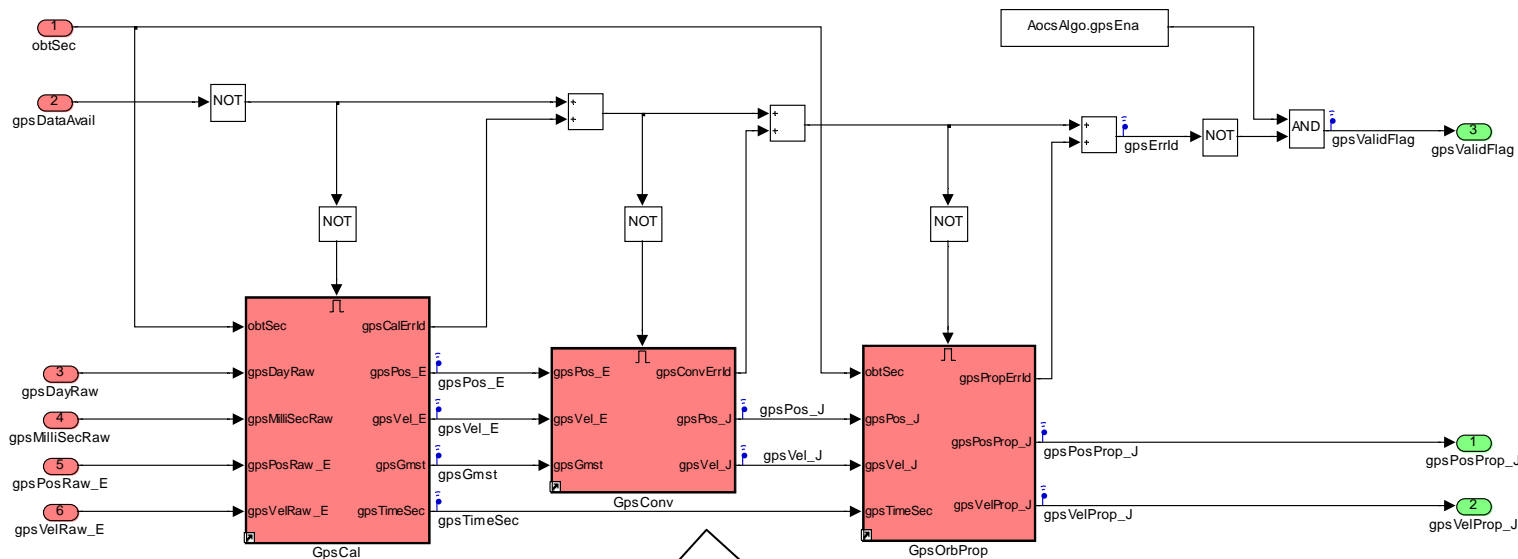
AOSE – AOCS Offline Simulator (cont'd)

- AocsAlgo – SensorProc, Level 2



This document is the property of Astrium. It shall not be communicated to third parties without prior written agreement. Its content shall not be disclosed.

Algorithm Development



Level 3 + Building Block
 Example: GPS data processing
 Architecture and data flow implemented using Simulink. Detailed algorithms implemented in Embedded Matlab (EM) function blocks.

```

Embedded MATLAB Editor - Block: ans/Obr/AocsAlgo/SensorProc/GpsProc/GpsConv/GpsConv
File Edit Text Debug Tools Window Help
1 function [gpsConvErrId, gpsPos_J, gpsVel_J] = ...
2 GpsConv(gpsPos_E, gpsVel_E, gpsGmst, satPosWLim, satPosUpLim, ...
3 satVelWLim, satVelUpLim, satPosVelCosLim, earthAngRate, dbzEps, dcm_JM)
4
5
6 % 1) The following variables shall be initialised
7
8 gpsConvErrId = uint16(0);
9
10 %-----
11 % 2) The transformation matrix from earth fixed frame to J2000 shall be computed
12 % considering a constant precession matrix.
13
14 dcm_JE = multDcms(dcm_JM, rot2Axis(-2*pi/86400*gpsGmst));
15
16 %-----
17 % 3) The position and velocity shall be converted to the inertial frame
18
19 gpsPos_J=multDcmVec(dcm_JE, gpsPos_E);
20 gpsVel_J= multDcmVec(dcm_JE, gpsVel_E+cross([0 0 earthAngRate]',gpsPos_E));
    
```



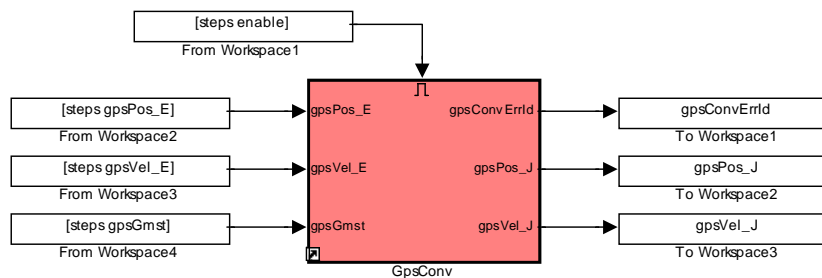
Algorithm Development (cont'd)

■ DSC-File (DSC: Descriptor)

- used for description, interfaces, masking, test setup, ... (see later)

■ Testing

- Each algorithmic BB is formally tested



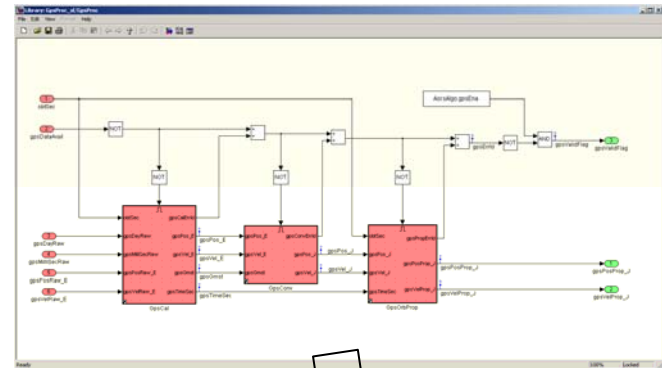
- Tests consists of one Simulink model and one or many test m-files
- Time series (not constant) for inputs
- Time series of simulated output is checked against time series of expected output with specified evaluation criteria
=> Test passed or failed

```
Editor - MS\AOC\S\AOS\AocsAlgoLib\BuildingBlocks\GpsConv\testGpsConv.m
File Edit View Tools Desktop Window Help
12 testRunName='('testGpsConvModel')';
13
14 %% Simulation Parameters
15 % (used for settings in the "Simulation Parameters" menu of the simulink model)
16
17 numSteps = 9; % Number of steps for the run
18 steps = [1:numSteps]'; % Step numbers (required by "from workspace" blocks)
19
20 %% Constants used
21 % Every constant must be explicitly defined here
22 % Never use constants directly defined in STAR (loadConstants.m)
23 %<constant1= value>
24
25 %% Model Parameters
26
27 % if a model parameter is a constant define it here
28 dcm_3X = [1 0 0 0 1 0 0 0 1];
29 satPosLowLim = 6578.e3 % Alt = 200 km
30 satPosUpLim = 6978.e3 % Alt = 600 km
31 satVelLowLim = 7.3e3 % Alt = 600 km
32 satVelUpLim = 7.8e3 % Alt = 200 km
33 satPosCoLim = 8.7e-2 % 5 deg
34 earthAngRate = 7.292115146700000e-005;
35 dbzEps = 1.e-14;
36
37 %% Model Inputs
38
39 % (e.g. for 2nd alternative)
40 enable = logical(ones(size(steps)));
41 enable(9) = logical(0);
42
43 gpsPos_E = [6700.e3 0 0;
44 0 6700.e3 0;
45 0 0 6700.e3;
46 1 0 0;
47 6700.e3 0 0;
48 6700.e3 0 0;
49 0 0 0;
50 6700.e3 0 0;
51 6700.e3 0 0];
52 gpsVel_E = [0 7.0e3 0;
53 0 0 7.4e3;
54 ]
```

This document is the property of Astrium. It shall not be communicated to third parties without prior written agreement. Its content shall not be disclosed.

“Executable” Specification

- sum of all Dsc-files
- + all Simulink architecture models
- + all Embedded Matlab code
- = AOCS S/W Requirements Specification



```

Embedded MATLAB Editor - Block: aocs/Obc/AocsAlgo/SensorProc/GpsProc/GpsConv/GpsConv
File Edit Text Debug Tools Window Help
1 function [gpsConvErrId, gpsPos_J, gpsVel_J] = ...
2   GpsConv(gpsPos_E, gpsVel_E, gpsGsmst, satPosLowLim, satPosUpLim, ...
3   satVelLowLim, satVelUpLim, satPosVelCosLim, earthAngRate, dbzEps, dcm_JH)
4
5 %-----
6 % 1) The following variables shall be initialised
7
8 gpsConvErrId = uint16(0);
9
10 %-----
11 % 2) The transformation matrix from earth fixed frame to J2000 shall be computed
12 %    considering a constant precession matrix.
13
14 dcm_JE = multDcms(dcm_JH, rotZAxis(-2*pi/86400*gpsGsmst));
15
16 %-----
17 % 3) The position and velocity shall be converted to the inertial frame
18
19 gpsPos_J = multDcmVec(dcm_JE, gpsPos_E);
20 gpsVel_J = multDcmVec(dcm_JE, gpsVel_E+cross([0 0 earthAngRate]', gpsPos_E));

```



SWARM Doc No: SW REQ EAD SV 0008

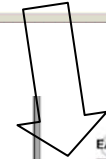
EADS ASTRIUM OBC S/W AOCS Requirements Specification

| Variable Name | Description | Data Type | Size | Phys. Unit |
|-----------------|---|-----------|------|------------|
| dcm_JE | Transformation matrix considering precession | double | 1 | - |
| satPosLowLim | Lower limit for norm of position vector | double | 1 | m |
| satPosUpLim | Upper limit for norm of position vector | double | 1 | m |
| satVelLowLim | Lower limit for norm of velocity vector | double | 1 | m |
| satVelUpLim | Upper limit for norm of velocity vector | double | 1 | m |
| satPosVelCosLim | Limit for cosine between position and velocity vector | double | 1 | - |
| earthAngRate | Earth angular rate | double | 1 | rad/s |
| dbzEps | Threshold for divisions by zero detection | double | 1 | -1 |

States: None

Traceable internal variables: None

Table 3.3-38: GpsConv IF definition



SWARM Doc No: SW REQ EAD SV 0008

EADS ASTRIUM OBC S/W AOCS Requirements Specification

3.3.3 Orbit Propagation of GPS Navigation Data (GpsObsProp)

An update of the navigation data is available from GPS every TRD sec. For further on-board usage, this function propagates the GPS navigation data (available in J2000 inertial coordinate system) to the COT every time when an update is received.

ASW 2127/T

The algorithm shall be as follows:

General Function Informations

| Function | GpsObsProp |
|------------------|---|
| Output | Propagates the GPS orbit state vectors to actual on-board time |
| Description | This function propagates the GPS position and velocity to the actual on-board time (OBT). |
| Enabled Function | Yes, bold States and error Outputs, enabled by: <code>gpsObsToC</code> |
| Instances | One instance, with same name as the function itself |

Table 3.3-39: GpsConv algorithm

3.3.3.3 Orbit Propagation of GPS Navigation Data (GpsObsProp)

An update of the navigation data is available from GPS every TRD sec. For further on-board usage, this function propagates the GPS navigation data (available in J2000 inertial coordinate system) to the COT every time when an update is received.

ASW 2127/T

The algorithm shall be as follows:

General Function Informations

| Function | GpsObsProp |
|------------------|---|
| Output | Propagates the GPS orbit state vectors to actual on-board time |
| Description | This function propagates the GPS position and velocity to the actual on-board time (OBT). |
| Enabled Function | Yes, bold States and error Outputs, enabled by: <code>gpsObsToC</code> |
| Instances | One instance, with same name as the function itself |

Embedded Matlab code is directly used in the written AOCS S/W specification.

(Generated through DOORS)

“Executable” Specification (cont’d)

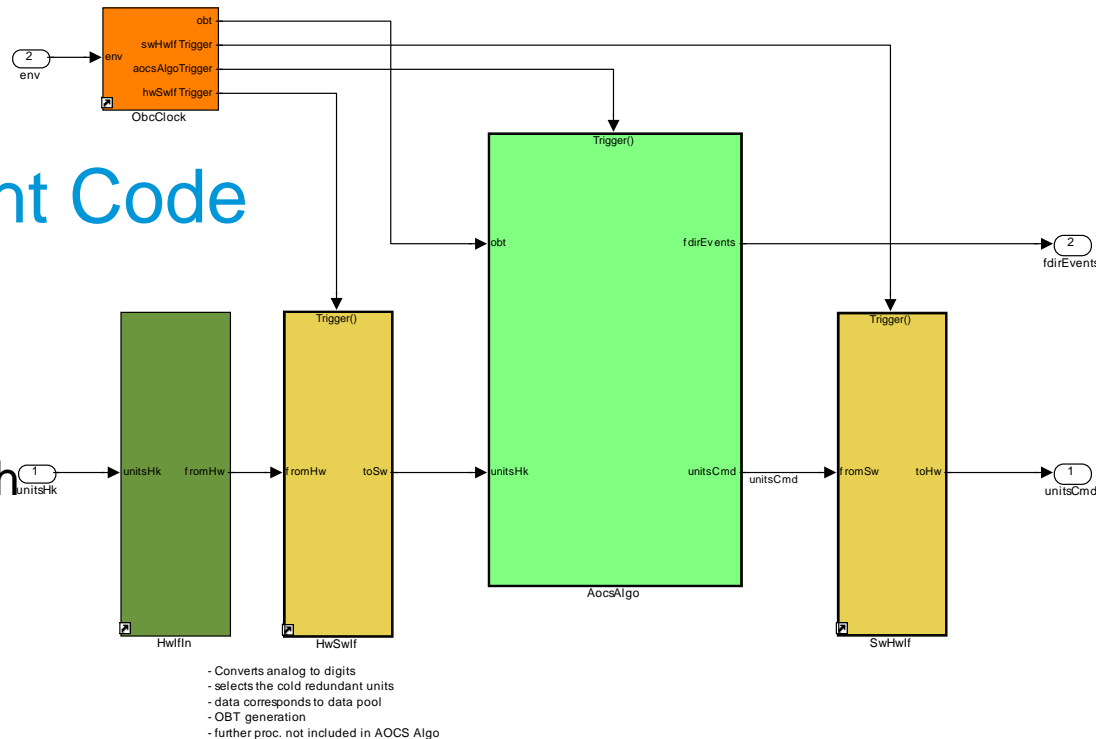
- **Delivery to code supplier**
 - Specification Document
 - Simulink and Embedded Matlab models (informal)
 - Unit test files for regression tests on BB level with
 - Input time series
 - Expected output time series
 - Evaluation criteria
 - Test Models in Simulink and Matlab (informal)

This document is the property of Astrium. It shall not be communicated to third parties without prior written agreement. Its content shall not be disclosed.

Reinjection of Flight Code

In principal

1. Take AOSE
2. Replace AocsAlgo with S-function incl. supplier Ada-code
3. That's all



Required to / with code supplier

- Clear interfaces for I/O, parameters, states, logging (TM), commanding (TC)
- Clear scheduling of AOCS S/W items
- Easy cut-out of AOCS S/W part from overall OBSW Ada code
- No direct packet handling and no direct PUS-services in AOCS S/W part
- Naming conventions wrt. Parameter (Variables) to be reused identically in SDB (TM/TC)

Reinjection of Flight Code (cont'd)

- Advantages to have AOSE as precursor
 - Environment, Dynamics and Satellite Modelling are ready and verified
 - Equipment Models are ready and verified
 - During development of AOSE
 - Single mode tests
 - Multi mode tests
 - FDIR testsare debugged and executed
 - Big part of these tests can be reused for formal
 - Open Loop testing on supplier premises
 - Closed Loop testing with FVB
 - Test scripts including plotting and evaluation can be reused without modifications
 - New tests can easily be created by change of parameterisation files