

# Data Modelling with ASN.1 for Space Applications

*ESA/ESTEC frame contract n°4000104809*

*Thanassis Tsiodras, Dr.-Ing*

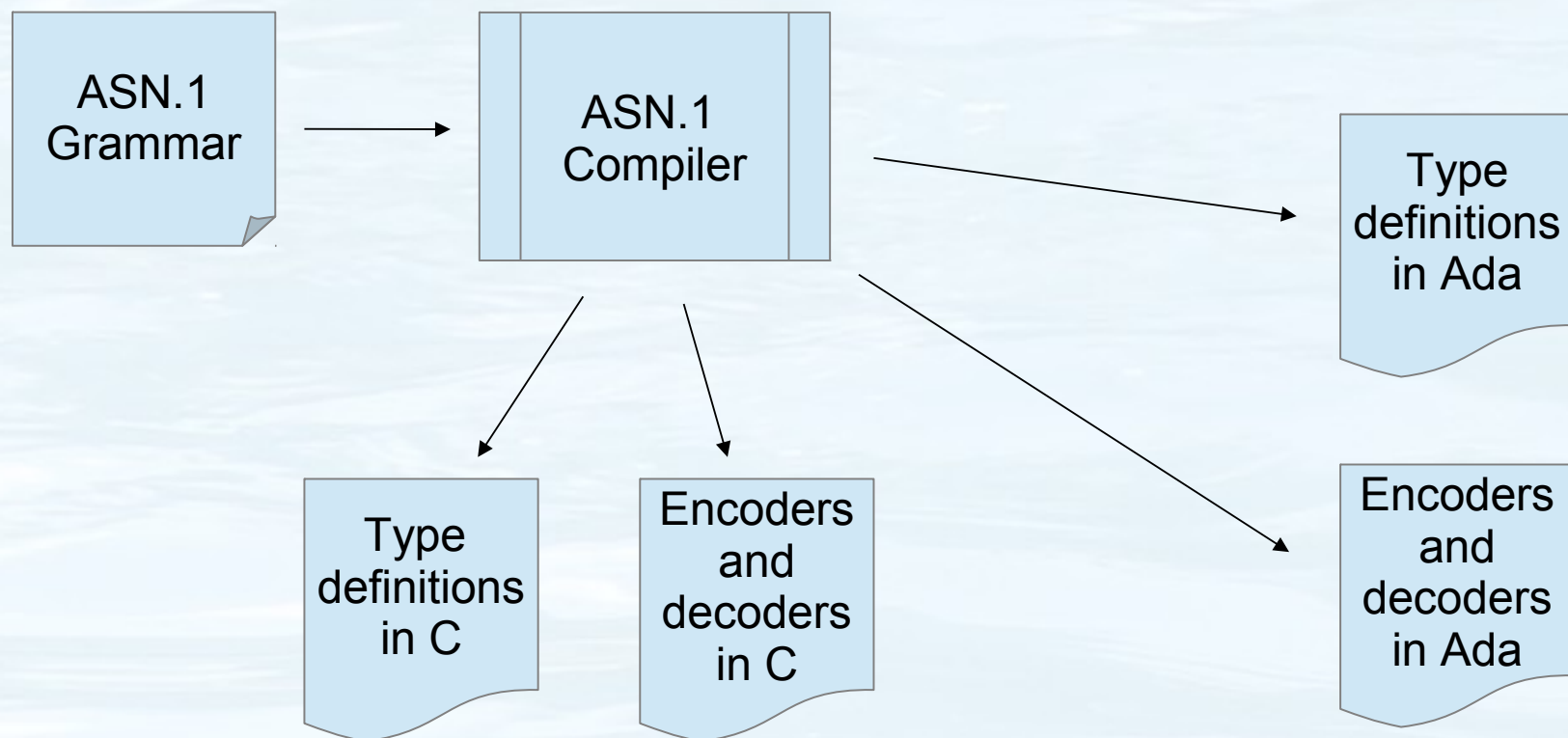
*NeuroPublic S.A.*

# ASN.1? What is that?

It's a "secret" weapon of the aeronautical, security and telecommunication domains - a simple language describing data structures, and offering multiple ways (trade-offs between CPU usage/space) to encode them:

```
DataView DEFINITIONS ::= BEGIN
T-THRUSTER-INDEX ::= INTEGER( 1 .. 10 ) -- Allowed: 1 to 10
T-REAL ::= REAL( -10000.0 .. 10000.0 ) -- Allowed value range
T-ACCELERATION ::= SEQUENCE (SIZE(3)) OF T-REAL -- Array of 3
T-ACS-CMD ::= SEQUENCE {
    set-thruster-on      BOOLEAN,
    set-thruster-index  T-THRUSTER-INDEX,
    set-thruster-data   T-ACCELERATION
}
END
```

# ASN.1? What is that?



ASN.1 compilers automatically generate your messages' type definitions, as well as your encoders and decoders – *you don't write (and test!) them manually*. You can also easily choose between encoding trade-offs (e.g. PER: packed encodes but more CPU, etc)



## Not so secret, really

- You all used it today. At least 100 times.
- OK, not you - your phone. “Dear local GSM cell tower, I am alive and well, you can find me here”.
- Your browser used it when you accessed any HTTPS-enabled site from your laptop / tablet / ...
- Your bank used it to send information about your account's balance yesterday.
- The local telecom provider in Noordwijk (Vodafone NL) used it to send your mobile phone's roaming charges to your home provider (e.g. Vodafone DEU)
- Etc... Billions of ASN.1 msgs exchanged per day...

## ASN.1 in space - ICDs (1/6)

- Airplanes use ASN.1 for Air Traffic Control => ESA became interested, and a study with Astrium back in 2008 showed many potential ASN.1 space applications...
- It's not just encoders and decoders. We can automatically generate \*many\* things from ASN.1 – because the type information “drives” many things.
- For starters: why write Interface Control Documents (ICDs) in Word/Excel? They can be automatically generated (with hyperlinks, too) from ASN.1 specs...

Message(SEQUENCE) <a href="#">ASN.1</a>				min = 14 bytes		max = 26 bytes	
No	Field	Comment	Optional	Type	Constraint	Min Length (bits)	Max Length (bits)
1	Preamble	Special field used by PER to indicate the presence/absence of optional and default fields. <ul style="list-style-type: none"> <li>bit1 == 1 ⇒ <i>isReady</i> is present</li> </ul>	No	Bit mask	N.A.	1	1
2	msgId		No	INTEGER	(0 .. 255)	8	8
3	myflag		No	INTEGER	(0 .. 1000)	10	10
4	value		No	REAL	(0.000000E+000 .. 9.999000E+003)	8	104 <a href="#">why?</a>
5	szDescription		No	OCTET-STRING	(SIZE(10))	80	80
6	isReady		Def	BOOLEAN	N.A.	1	1

## File : DataView.asn

```

MY-MODULE DEFINITIONS AUTOMATIC TAGS ::= BEGIN
Message ::= SEQUENCE {
    msgId INTEGER (0..255),
    myflag INTEGER (0..1000),
    value REAL (0.0..9999.0),
    szDescription OCTET STRING (SIZE(10)),
    isReady BOOLEAN DEFAULT TRUE
}

```



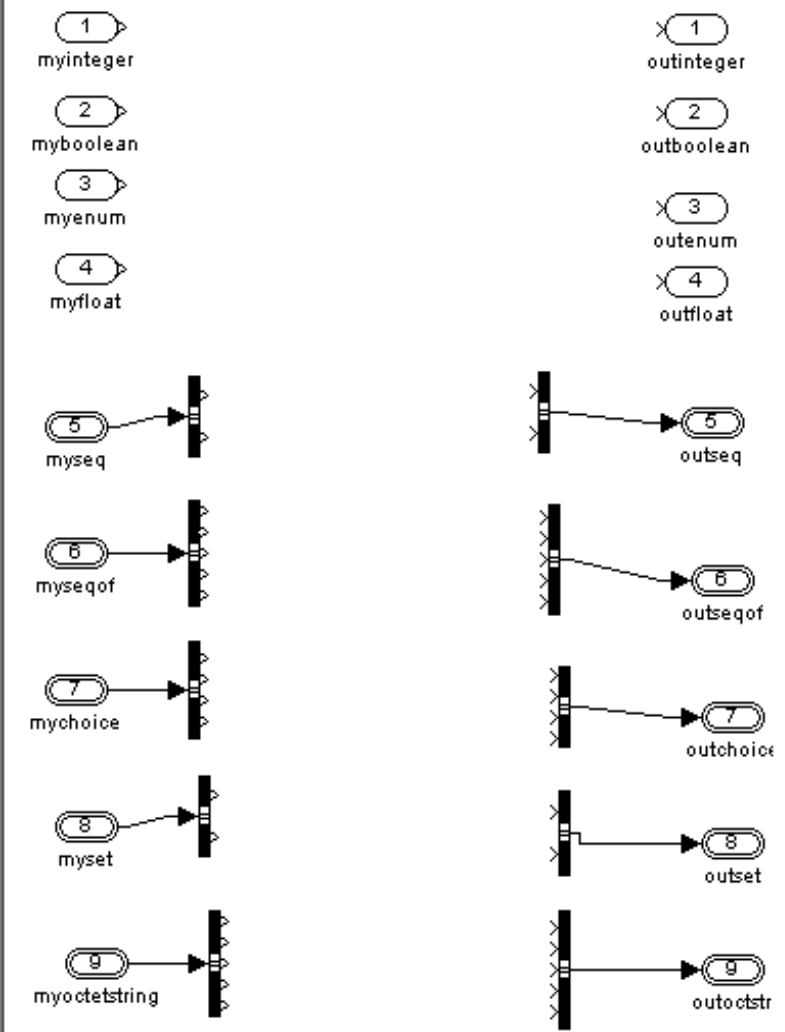
## ASN.1 in space - “glue” code (2/6)

- Automatically generated runtime “translators” of messages between code written in different languages (C, Ada, Python) and/or different tools (Simulink/RTW, SCADE, PragmaDev RTDS, etc)

```
typedef struct {  
    GU_RG_51_10 fd_height;  
    GU_RG_50_9 fd_latitude;  
    GU_RG_49_8 fd_longitude;  
    GU_SEQOF_52_11 fd_subtypearray;  
-} GU_T_POS;
```

```
typedef struct {  
    real_T longitude;  
    real_T latitude;  
    real_T height;  
    Subtypearray_type subtypearray;  
} T_POS;
```

- You write code in Simulink/RTW and “glue” it to your projects manually? There's no need to suffer that!



### Bus Types Editor

Bus types in base workspace

- T\_CHOICE
  - choiceIdx
  - boolchoice
  - enumchoice
  - intchoice
- T\_NESTED
- T\_OCTSTR
- T\_SEQOF
- T\_SEQUENCE
  - x
  - y
- T\_SET
- T\_SETOF

Bus elements

Name	Dimension	Data/Bus T...	Sample Time	Complexity	Sampling M...
choiceIdx	1	uint8	-1	real	Sample...
boolchoice	1	boolean	-1	real	Sample...
enumchoice	1	int32	-1	real	Sample...
intchoice	1	uint8	-1	real	Sample...

Bus name:  Header file that contains typedef for bus:

Bus description:

Loaded bus objects existing in base workspace

Help Close



RTDS - Project "orchestrator\_project.rdp" (modified)

File Edit View Element Generate Tools Windows Help

orchestrator\_project.rdp

- ASN1Types
  - RTDSdataView.asn
- orchestrator
- scheduled

Opening project... Done.

orchestrator\_project.rdp RTDSdataView.asn orchestrator

RTDS RTDSdataView.asn

File Edit Search Preferences Windows Help

RTDSdataView.asn

```
DataModel DEFINITIONS ::=
BEGIN

-- Input types
Digital-Inputs ::= SEQUENCE {
  sw-cmd BOOLEAN,
  sw-gripper BOOLEAN
}

Analog-Inputs ::= SEQUENCE (SIZE(16)) OF REAL (0.0 .. 6.0)

-- Output types
VR-Model-Output ::= SEQUENCE {
  x1 REAL (-1000 .. 1000),
  y1 REAL (-1000 .. 1000),
  z1 REAL (-1000 .. 1000),
  p1 REAL (-1000 .. 1000),
  x2 REAL (-1000 .. 1000),
  y2 REAL (-1000 .. 1000),
  z2 REAL (-1000 .. 1000),
  p2 REAL (-1000 .. 1000),
  x3 REAL (-1000 .. 1000),
  y3 REAL (-1000 .. 1000),
  z3 REAL (-1000 .. 1000),
  p3 REAL (-1000 .. 1000),
  j-rad SEQUENCE (SIZE(16)) OF REAL (-1000 .. 1000)
}

VR-Arm-Configuration ::= SEQUENCE {
  -- TAGS (1..15)

```

/home/assert/buildsupport/misc/robot/Inputs/models/orchestrator/R line 17 col 0 695 bytes

RTDS - Diagram "orchestrator" (modified)

Diagram Edit Search View Export Windows Help

100%

orchestrator

```
USE ASN1Types;
```

```
signal digital_command(Digital_Inputs);
signal analog_command(Analog_Inputs);
signal cyclic_activation;
signal vr_command(VR_Model_Output);
```

```
← taste_envl
[vr_command]
[digital_command,
analog_command,
cyclic_activation]
orchestrator_p
```

```
PROCEDURE control_law(
  IN in_analog Analog_Inputs,
  IN in_digital Digital_Inputs,
  IN/OUT out_vr VR_Model_Output
) EXTERNAL;
```

## ASN.1 in space – TM/TC GUIs, tests (3/6)

- TM/TC GUIs: we generate them automatically since 2010 (i.e. 0% manually written code), and they allow us to communicate with our running systems - and both monitor and control them easily.
- We can also watch (and graph) the TM/TC message exchanges in real-time MSC diagrams – again, the necessary code is generated automatically.
- The system's integration tests – why not write them in a nice scripting language? We automatically generate Python mappings for your ASN.1 messages – so system testing can be scripted, added to nightly regression-checks, etc.

# ASN.1 in space – TM/TC GUIs, tests (3/6)

The screenshot displays the 'taste' GUI interface for testing. It is divided into several sections:

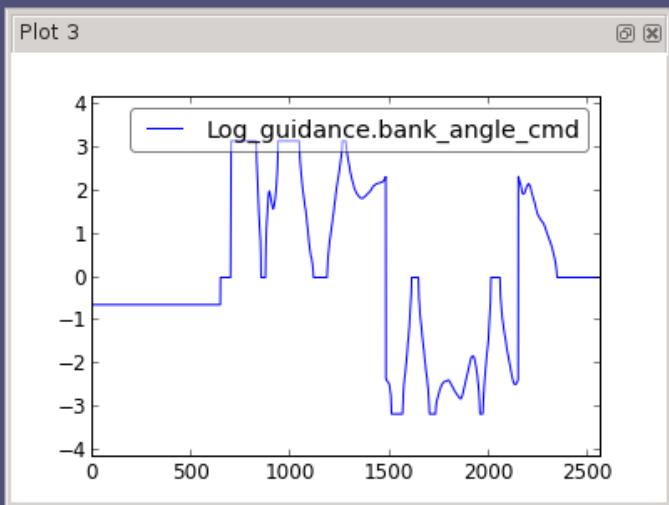
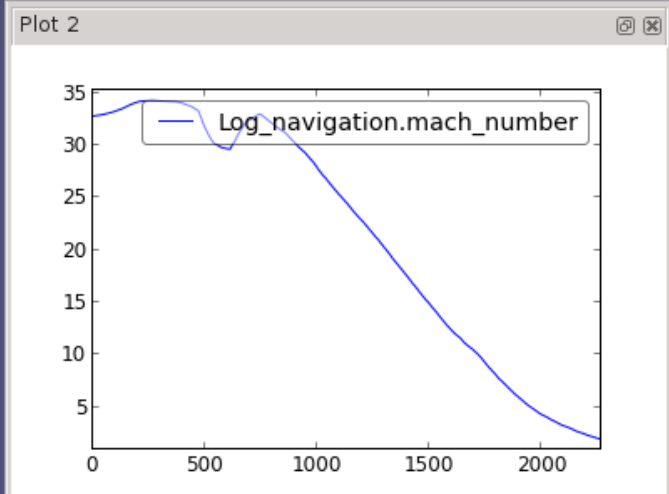
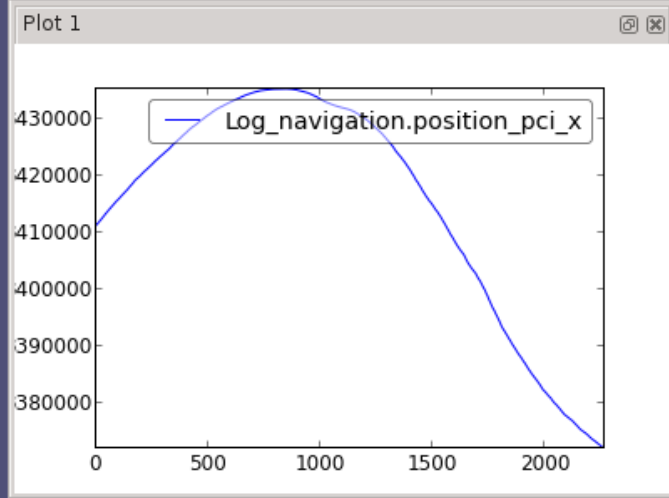
- Available test scripts:** A list of two scripts: `ground_segment_trace_201204101828.msc` and `ground_segment_trace_201204101822.msc`.
- data\_response Table:** A table showing the current values for the data\_response object.

Field	Value
data_response	
input_signal	0.08090173895
output_signal	4.60557413171
- start Table:** A table showing the current values for the start object.

Field	Value
start	
initial_value	0
control_law	integrator
input_data	ramp
- Plot 1:** A line graph showing the input and output signals over time. The x-axis represents time from 0 to 50, and the y-axis represents signal amplitude from -2 to 5. The blue line (input\_signal) shows a small oscillation around 0. The green line (output\_signal) shows a ramp that starts at -2 and increases to approximately 4.5.
- SM 2: data\_response.input\_signal:** A circular meter showing the current value of the input signal, which is 0. The range is -5 to 5, and the needle is at 0. A green checkmark indicates the value is within the range.
- SM 3: data\_response.output\_si...:** A circular meter showing the current value of the output signal, which is approximately 4.6. The range is -3 to 3, and the needle is at approximately 4.6. A green checkmark indicates the value is within the range.

At the bottom of the main window, there are buttons for 'Run', 'Load', 'Edit', 'Send TC', 'Load TC', and 'Save TC'. The text 'ENUMERATED' is visible at the bottom left.





dashboard

MSC

# taste

Available test scripts:

Run Load Edit

REAL (-999999999999.00..999999999999.00)

Log\_cont... Log\_guida... **Log\_navigati...** Log\_pl...

Log\_navigation

Field	Value
declination	0.0621092223528
longitude	0.092856522519
fp_speed	437.578872008
fp_inclination_gc	-0.256068594663
fp_azimuth_gc	0.668440715168
aoa	-0.294850578393
aos	-0.0147469120202
bank_angle	-0.000224468824261
omega_x_b_pci_b	0.000161159081628
omega_y_b_pci_b	0.00264147510894
omega_z_b_pci_b	5.02820288741e-05

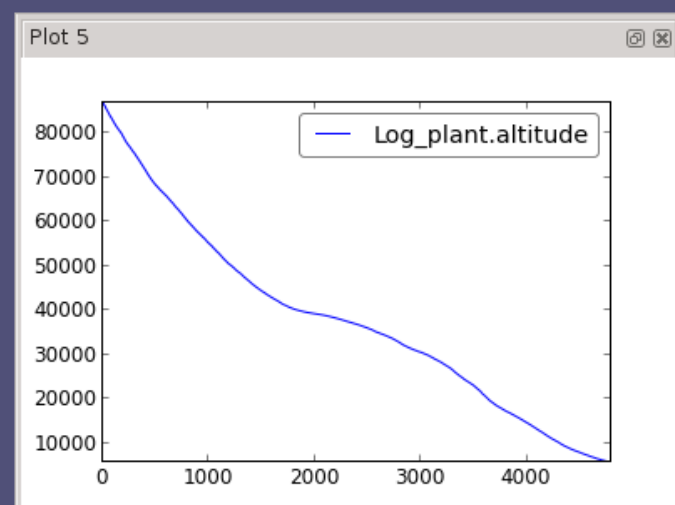
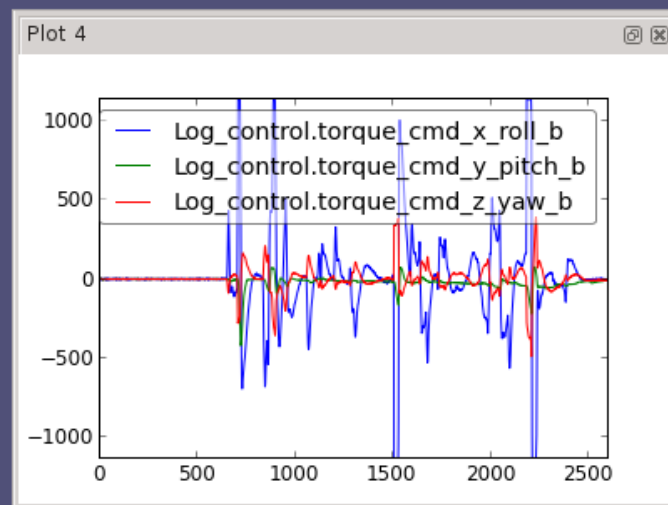
Plot Meter

start\_command stop\_command

start\_command

Field	Value
start_command	
torque_cmd_x_roll_b	0
torque_cmd_y_pitch_b	0
torque_cmd_z_yaw_b	0

Send TC Load TC Save TC




## ASN.1 in space - Databases (4/6)

- Databases – automatically store your ASN.1 messages (TM/TC, whatever)
- SQL mappers – they map your mission's ASN.1 messages to semantically identical database tables' definitions.
- But that's not all - we also generate runtime mappers for Python, addressing all major open-source database engines: If you use SQLite, MySQL or PostgreSQL, you can store and restore your message data with one-liners... No need for manual tinkering of any sort.

## ASN.1 in space – DB examples (4/6)

- The SQL mapper generates a table for each ASN.1 type, with a “sequence” (*autoincrement*-ed) primary key.
- For primitive types (INTEGER, REAL, etc) the table has a “data” field of the appropriate type, with the necessary constraints:

```
MyInt ::= INTEGER (0 .. 20)
```



```
CREATE TABLE "MyInt"  
(  
  iid serial NOT NULL,  
  data integer NOT NULL,  
  CONSTRAINT "MyInt_pkey" PRIMARY KEY (iid),  
  CONSTRAINT "MyInt_data_check" CHECK (data >= 0 AND data <= 20)  
)
```



## ASN.1 in space – DB examples (4/6)

- For SEQUENCE types (records), the mapper generates detail tables, and adds appropriate foreign keys.
- For SEQUENCE OF types (arrays), the mapper generates index fields as well.

```
MyInt ::= INTEGER (0 .. 20)
My2ndInt ::= MyInt ( 1 .. 18)
MySeq ::= SEQUENCE {
    anInt MyInt,
    anotherInt My2ndInt
}
```

```
CREATE TABLE "MySeq"
(
    iid serial NOT NULL,
    "fk_anInt_iid" integer NOT NULL,
    "fk_anotherInt_iid" integer NOT NULL,
    CONSTRAINT "MySeq_pkey" PRIMARY KEY (iid),
    CONSTRAINT "MySeq_fk_anInt_iid_fkey"
        FOREIGN KEY ("fk_anInt_iid")
        REFERENCES "MyInt" (iid),
    CONSTRAINT "MySeq_fk_anotherInt_iid_fkey"
        FOREIGN KEY ("fk_anotherInt_iid")
        REFERENCES "My2ndInt" (iid)
)
```

```
class TypeNested_SQL(Base):
    __tablename__ = 'TypeNested'
    __table_args__ = (UniqueConstraint('iid'),)
    iid = Column(Integer, primary_key=True)
```

```
@staticmethod
```

```
def loadFromDB(session, iid):
    return session.query(
        TypeNested_SQL).filter(TypeNested_SQL.iid == iid).first()
```

```
@property
```

```
def asn1(self):
    if hasattr(self, "_cache"):
        return self._cache
    pyObj = TypeNested()
    self.assignToASN1object(pyObj)
    self._cache = pyObj
    return pyObj
```

```
def assignToASN1object(self, pyObj):
    state = pyObj.GetState()
    pyObj.Reset(state)
    self.intVal.assignToASN1object(pyObj.intVal)
    pyObj.Reset(state)
    self.int2Val.assignToASN1object(pyObj.int2Val)
    pyObj.Reset(state)
    self.int3Val.assignToASN1object(pyObj.int3Val)
    pyObj.Reset(state)
    self.intArray.assignToASN1object(pyObj.intArray)
    pyObj.Reset(state)
    self.realArray.assignToASN1object(pyObj.realArray)
    pyObj.Reset(state)
    self.octStrArray.assignToASN1object(pyObj.octStrArray)
    pyObj.Reset(state)
    self.boolArray.assignToASN1object(pyObj.boolArray)
    pyObj.Reset(state)
    self.enumArray.assignToASN1object(pyObj.enumArray)
    pyObj.Reset(state)
    self.enumValue.assignToASN1object(pyObj.enumValue)
    pyObj.Reset(state)
```





SCHEMAS

- AType
- AType\_bIArray
- AType\_bIArray\_indexes
- My2ndAType
- My2ndArr
- My2ndArr\_indexes
- My2ndBool
- My2ndEnumerated
- My2ndInt
- My2ndInt\_1
- My2ndReal
- My2ndString
- My2ndTypeNested
- MyChoice
- MyChoice\_aReal
- MyInt
- MySeq**
  - Columns
    - iid
    - fk\_anInt\_iid
    - fk\_anotherInt\_iid

Object Info Session

Table: MySeq  
 Columns:  
 iid int(11) PK AI  
 fk\_anInt\_iid int(11)  
 fk\_anotherInt\_iid int(11)  
 Related Tables: MyInt (fk\_anInt\_iid → iid)  
 My2ndInt (fk\_anotherInt\_iid → iid)

SQL File 1 x Query 1 x

```

1
2 • SELECT * FROM test.MySeq;
3
4

```

Filter: Edit: Export: Autosize:

#	iid	fk_anInt_iid	fk_anotherInt_iid
1	1	2	2
2	2	4	3
3	3	5	4
*	NULL	NULL	NULL

MySeq 1 x Apply Revert

Action Output

	Time	Action	Message	Duration
✓ 1	19:12:38	SELECT * FROM test.MySeq LIMIT 0, 1000	3 row(s) returned	0.001 s



# ASN.1 in space – Safety Critical (5/6)

- But... is it safe? This is safety-critical code!
- Meet our compiler ( <https://github.com/ttsiodras/asn1scc> ): Open-source, written in F# (OCaml-derivative), a language where many programmer errors are caught at compile-time (option types, pattern matching on type forms, etc). *Translation:* a compiler where large categories of errors are impossible.
- It generates code for C/C++ and SPARK/Ada. Well, SPARK encoders and decoders come with code verification – the message encoders and decoders are **proven**:

```
PROCEDURE BitStream_AppendByte (  
    Strm      : IN OUT BitStream;  
    ByteValue :          Unsigned_8;  
    Negate    : IN      Boolean) IS  
  
    --# pre Strm.CurrentBit+8>Strm.Data'SIZE+1  
  
    ByteVal : Unsigned_8 := ByteValue;  
BEGIN  
  
    IF Negate THEN
```

## ASN.1 in space – Safety Critical (5/6)

- Automatically generated test cases for your ASN.1 grammars, that provide 100% coverage of the code (gcov)
- No dynamic memory (heap) or syscalls – portable code
- Run-time library (RTL) is open, too - no black boxes, minimal and optimal.
- In benchmarks, there's less than 10% speed difference with the top commercial ASN.1 compiler
- Legacy encodings? Compatibility with the past? Sure – ACN allows you to completely control the serialization format used in the binary streams



## ASN.1 in space - FPGAs (6/6)

- You work with FPGAs? We can help.
- Our code generators automatically map a system's ASN.1 grammar to ...
  - (a) VHDL and SystemC skeletons for your designs, with all the interface declarations ready-made for you (just fill in the body of the logic)
  - (b) device drivers that automatically interface with the FPGA, communicating with it at runtime. We prototyped this over a USB accessible Xilinx Spartan3, and then tried it on an FPGA accessible at runtime over Leon's PCI bus.



# ASN.1 in space - FPGAs (6/6)

## VHDL Skeleton

```
architecture archivhdl_aes of vhdl_aes is
begin
  process(clk_vhdl_aes,rst_vhdl_aes)
    variable run : std_logic;
  begin
    if rst_vhdl_aes='0' then -- Asynchronous reset
      finish_vhdl_aes <= '0';
      -- write your resets here
      run := '1';

    elsif clk_vhdl_aes'event and clk_vhdl_aes='1' then
      if start_vhdl_aes = '0' then
        finish_vhdl_aes <= '0';
        run := '1';
      elsif run = '1' then
        -- write your logic to compute outputs from inputs here
        -- and when your results are ready, set...
        --
        -- run := '0';
        -- finish_vhdl_aes <= '1';
      end if;
    end if;
  end process;
end archivhdl_aes;
```

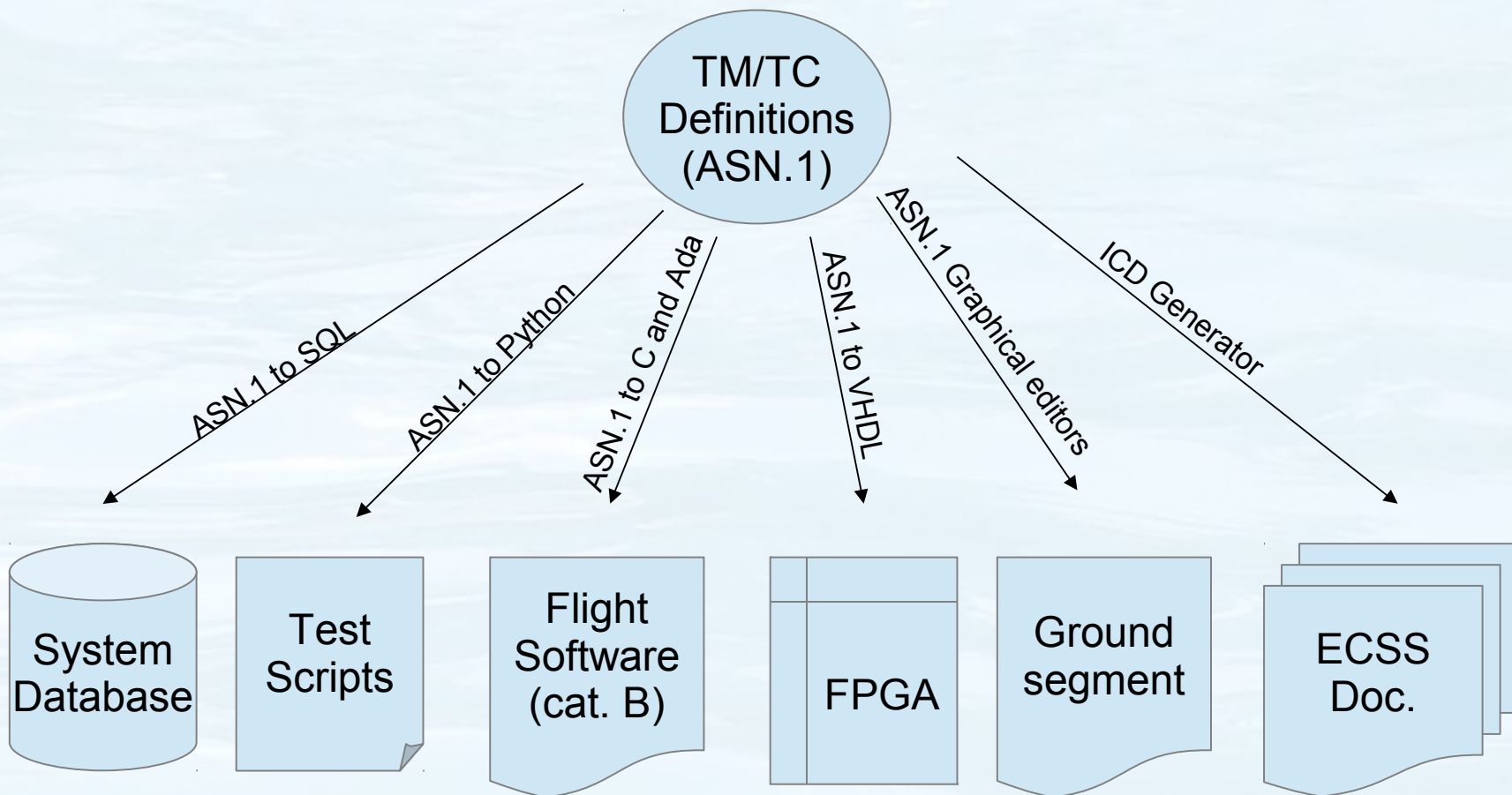
# ASN.1 in space - FPGAs (6/6)

Automatically generated device driver:

```
if (var_T_VHDL_Arg.kind == t_vhdl_aes_arg_set_key_PRESENT) {
    unsigned tmp = 1;
    ESAWriteRegister(BASE_ADDR + 0x4, tmp);
    {
        unsigned tmp = var_T_VHDL_Arg.u.t_vhdl_aes_arg_set_key.t_arg_key_length;
        ESAWriteRegister(BASE_ADDR + 0x8, tmp);
    }
    {
        unsigned tmp = 0;
        tmp |= ((unsigned)var_T_VHDL_Arg.u.t_vhdl_aes_arg_set_key.t_arg_key_content.arr[0]) << 0;
        tmp |= ((unsigned)var_T_VHDL_Arg.u.t_vhdl_aes_arg_set_key.t_arg_key_content.arr[1]) << 8;
        tmp |= ((unsigned)var_T_VHDL_Arg.u.t_vhdl_aes_arg_set_key.t_arg_key_content.arr[2]) << 16;
        tmp |= ((unsigned)var_T_VHDL_Arg.u.t_vhdl_aes_arg_set_key.t_arg_key_content.arr[3]) << 24;
        ESAWriteRegister(BASE_ADDR + 0xc + 0, tmp);
    }
}
```



# Summary: a single data definition to ensure consistency everywhere





## Meet TASTE ([taste.tuxfamily.org](http://taste.tuxfamily.org))

- There's much more. These were just the highlights!
- You are cordially invited to download the [TASTE VM](#), a Virtual Machine that contains all the tools you saw - built over the last 6 years. Use this VM with the free VMWARE Player (Windows, Linux) or VirtualBox (Linux, OS/X)
- The VM auto-updates itself, once you boot it, via a simple “UpdateTASTE.sh” script – or a simple double-click on a desktop icon :-) You are always up to date in terms of our tools.
- The technology is quite mature – join us! We can build a mission together, even up to a complete satellite.

<http://taste.tuxfamily.org>

**Questions?**

<http://taste.tuxfamily.org>

**Questions?**