

Mapping IEC 61131-3 Directly Represented Variables to CANopen

This paper

- Maps CANopen variables hierarchy to IEC-61131-3 “Directly Represented Variables”
- Could help in defining some uncovered “manufacturer specific” aspects of DS-405
- Proposes some algorithms for implementation

#	Name	Class	Type	Location
1	DigitalOut	Local	BYTE	%QB503254.0.16.25088
2	DigitalIn	Local	BYTE	%IB503254.0.32.24576.
3	AnalogOut1	Local	MYTYPE	%QW503254.0.16.2561
4	AnalogOut2	Local	MYTYPE	%QW503254.0.16.2561.
5	AnalogOut3	Local	INT	%QW503254.0.16.2561.
6	AnalogIn1	Local	INT	%IW503254.0.32.25601
7	AnalogIn2	Local	INT	%IW503254.32.25601.2



Directly represented variables

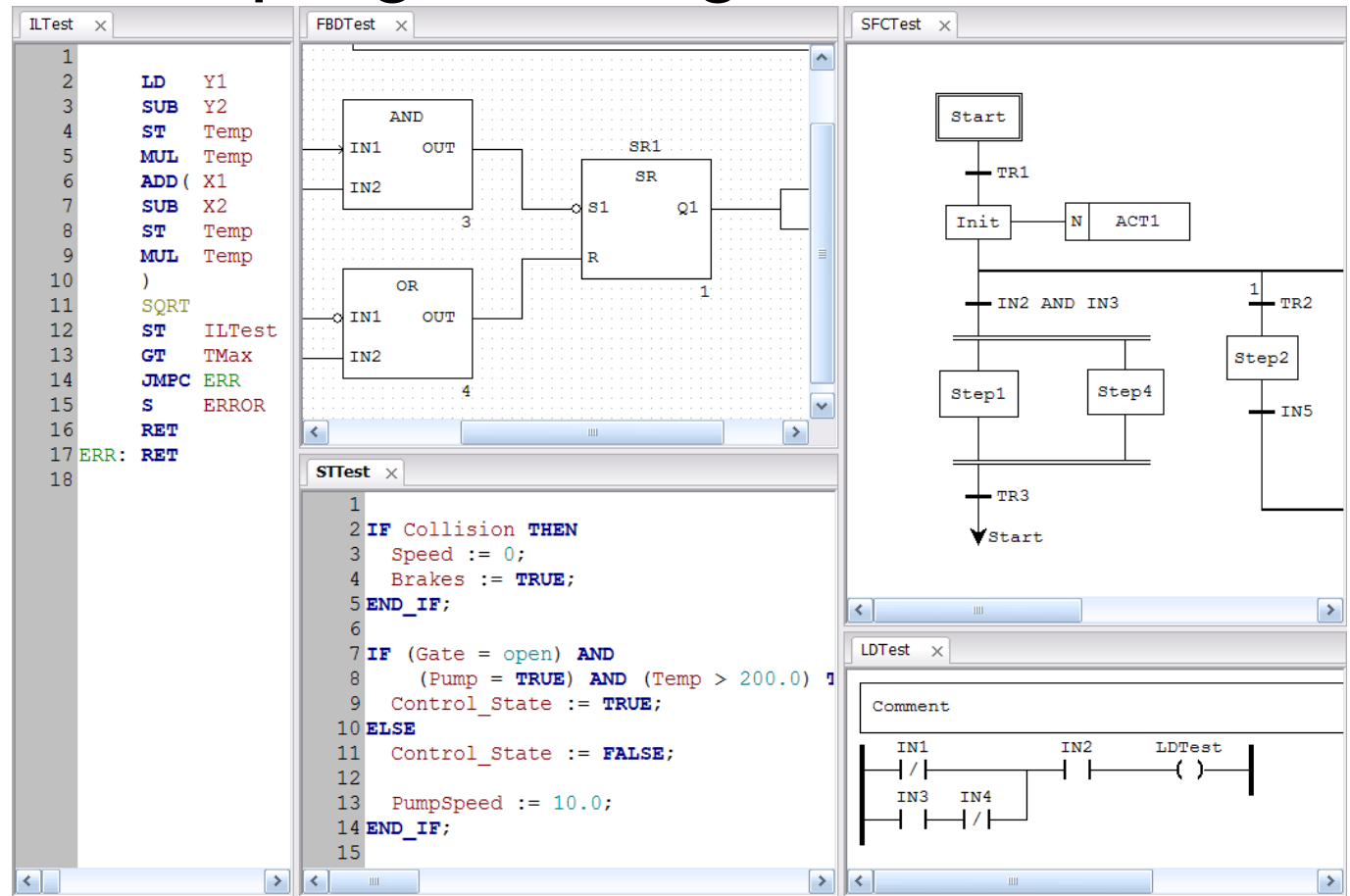
CANopen

About IEC 61131-3

• IEC 61131 rules PLC programming art

• Languages :

- 3 graphical
 - SFC
 - FBD
 - LD
- 2 textual
 - ST
 - IL



The screenshot displays four windows illustrating IEC 61131-3 languages:

- ILTest x**: Instruction List (IL) code.


```

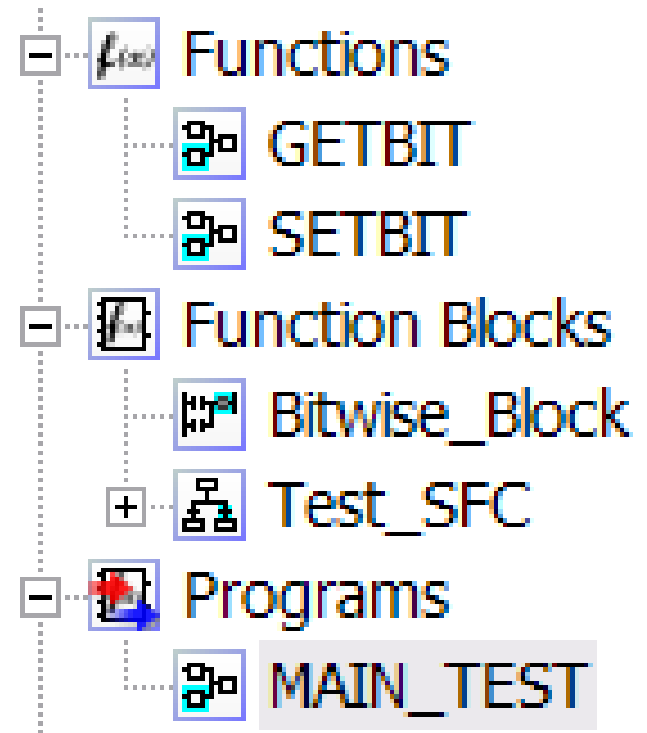
1 LD Y1
2 SUB Y2
3 ST Temp
4 MUL Temp
5 ADD ( X1
6 SUB X2
7 ST Temp
8 MUL Temp
9 )
10 SQRT
11 ST ILTest
12 GT TMax
13 JMP ERR
14 S ERROR
15 RET
16
17 ERR: RET
18
```
- FBDTest x**: Function Block Diagram (FBD) showing an AND block connected to an SR (Set Reset) block, and an OR block connected to the same SR block. The SR block has inputs S1 and R, and output Q1.
- SFCTest x**: Sequential Function Chart (SFC) showing a sequence of steps: Start, Init, Step1, Step4, Step2, and back to Start. Transitions include TR1, IN2 AND IN3, TR2, TR3, and IN5. An action ACT1 is associated with the transition from Init to Step1.
- STTest x**: Structured Text (ST) code.


```

1
2 IF Collision THEN
3   Speed := 0;
4   Brakes := TRUE;
5 END_IF;
6
7 IF (Gate = open) AND
8   (Pump = TRUE) AND (Temp > 200.0)
9   Control_State := TRUE;
10 ELSE
11   Control_State := FALSE;
12
13   PumpSpeed := 10.0;
14 END_IF;
15
```
- LDTest x**: Ladder Diagram (LD) showing a network with inputs IN1, IN2, IN3, and IN4, and output LDTest. The network consists of a normally open contact for IN1, a normally closed contact for IN2, and a normally open contact for IN3 in parallel with a normally open contact for IN4.

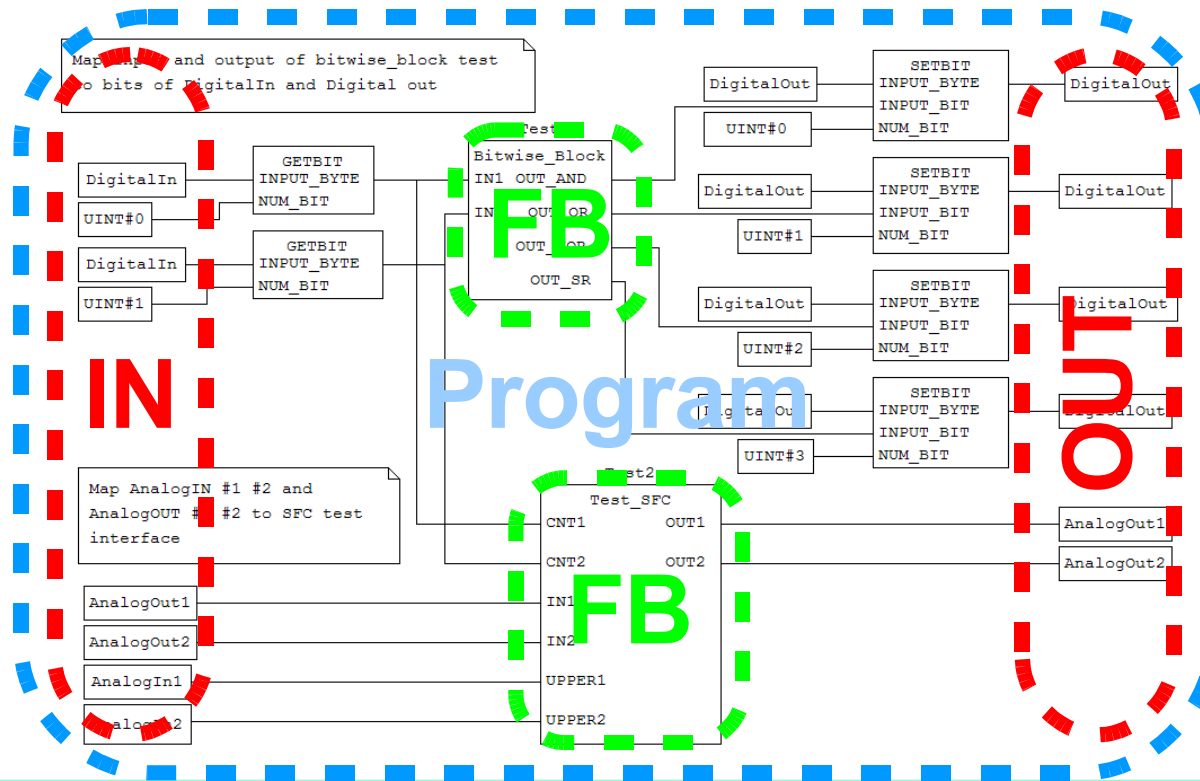
About IEC 61131-3

- PLC Programmers organize their work through “Program Organization Units” (POU)
- POU can be:
 - Functions (no instance, idempotents)
 - Function Blocks (instantiated, internal state)
 - Programs (instantiated, internal state, I/O access)



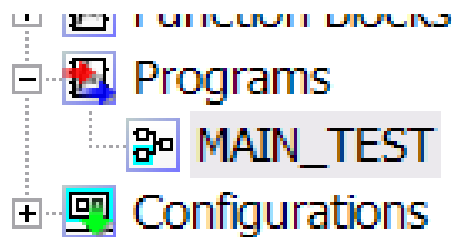
About IEC 61131-3

- Usually, function blocks are instantiated in Programs in order to be “connected” to I/Os.



Direct Representation

- “Directly represented variables” represent I/Os
- Locations are integers separated by dots



The screenshot shows a variable panel with the following data:

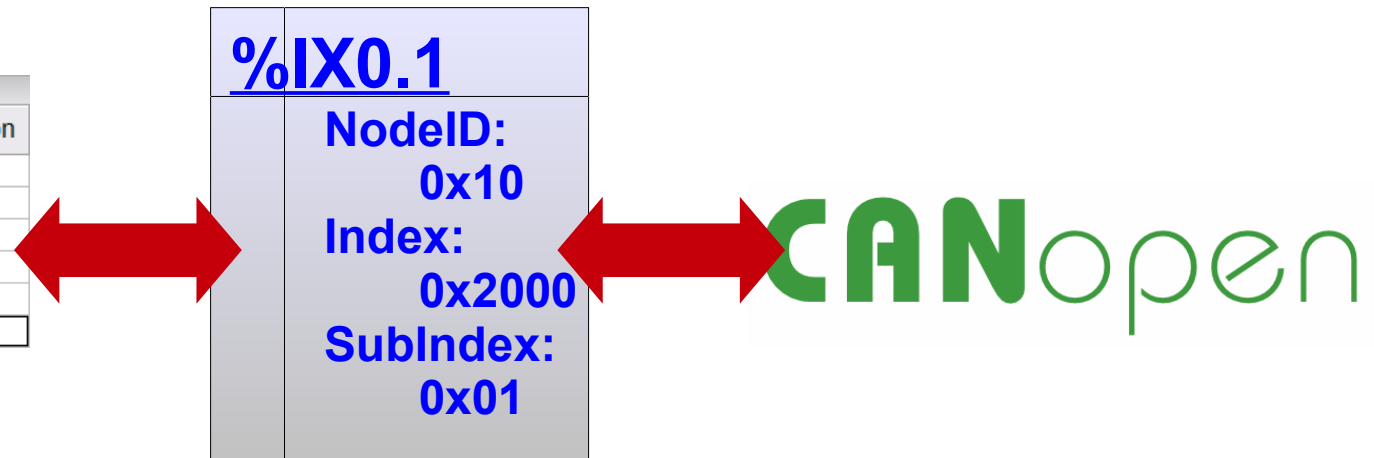
#	Name	Class	Type	Location
1	DigitalOut	Local	BYTE	%QB503254.0.16.25088.1
2	DigitalIn	Local	BYTE	%IB503254.0.32.24576.1
3	AnalogOut1	Local	MYTYPE	%QW503254.0.16.25617.1

Below the table, a ladder logic diagram is visible, showing a box labeled 'DigitalIn' connected to a 'GETBIT' function block. The function block has inputs 'INPUT_BYTE' and 'NUM BIT', and an output 'Bitw: IN1'.

IEC 61131-3 and CANopen

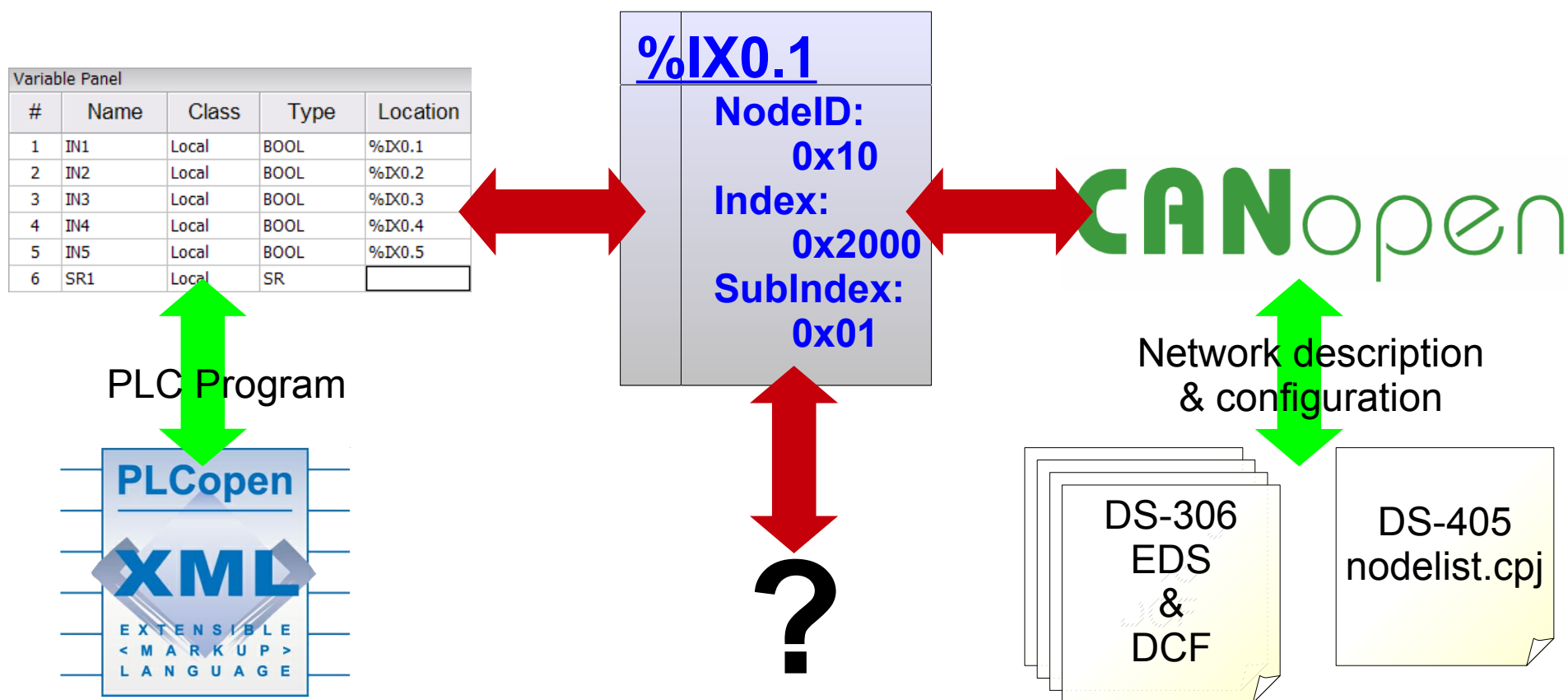
- CANopen PLC users have to define an intermediate nomenclature between Programs interface and CANopen.

#	Name	Class	Type	Location
1	IN1	Local	BOOL	%IX0.1
2	IN2	Local	BOOL	%IX0.2
3	IN3	Local	BOOL	%IX0.3
4	IN4	Local	BOOL	%IX0.4
5	IN5	Local	BOOL	%IX0.5
6	SR1	Local	SR	



IEC 61131-3 and CANopen

- File format for this nomenclature is not specified



Proposal

- Express nomenclature in variables locations

Variable Panel				
#	Name	Class	Type	Location
1	IN1	Local	BOOL	%IX0.1
2	IN2	Local	BOOL	%IX0.2
3	IN3	Local	BOOL	%IX0.3
4	IN4	Local	BOOL	%IX0.4
5	IN5	Local	BOOL	%IX0.5
6	SR1	Local	SR	

%IX0.1

NodeID:
0x10

Index:
0x2000

SubIndex:
0x01

CANopen

Variable Panel				
#	Name	Class	Type	Location
1	DigitalOut	Local	BYTE	%QB503254.0.16.25088
2	DigitalIn	Local	BYTE	%IB503254.0.32.24576.
3	AnalogOut1	Local	MYTYPE	%QW503254.0.16.2561
4	AnalogOut2	Local	MYTYPE	%QW503254.0.16.2561
5	AnalogOut3	Local	INT	%QW503254.0.16.2561
6	AnalogIn1	Local	INT	%IW503254.0.32.25601
7	AnalogIn2	Local	INT	%IW503254.32.25601.2

Directly represented variables

CANopen

Numbering scheme

- Variable location explicitly represent the path in the CANopen hierarchy to access any object.
- Example : Remote First DS-401 Read Input Byte

Direction : In
 Data Size : Byte
 Protocol : CanOpen
 Bus-ID : 0
 Node-ID : 2
 Transmit Type : 1
 Index : 6000h
 SubIndex : 1

%IB503254 . 0 . 2 . 1 . 24576 . 1

Numbering scheme

- Depending on count of integers, location may represent remote or local CANopen variables
- Example : Read access to local Error Register

Direction : In
Data Size : Byte
Protocol : CanOpen
Bus-ID : 0
Index : 1001h
SubIndex : 0

%IB503254.0.4097.0

Numbering scheme

- Local representations can also be used to implicitly declare arbitrary new OD entry
- Example : New Manufacturer Specific entry

Direction : In
 Data Size : Byte
 Protocol : CanOpen
 Bus-ID : 0
 Index : 2001h
 SubIndex : 0

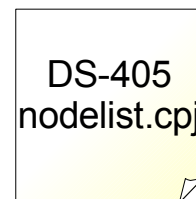
%IB503254 . 0 . 8193 . 0

Where it helps

- Avoids network reconfiguration steps when revamping CANopen machines
- Eases portage of PLC programs through different CANopen controller brands
- Provides long-term re-usability, independently of PLC solutions life time.



+

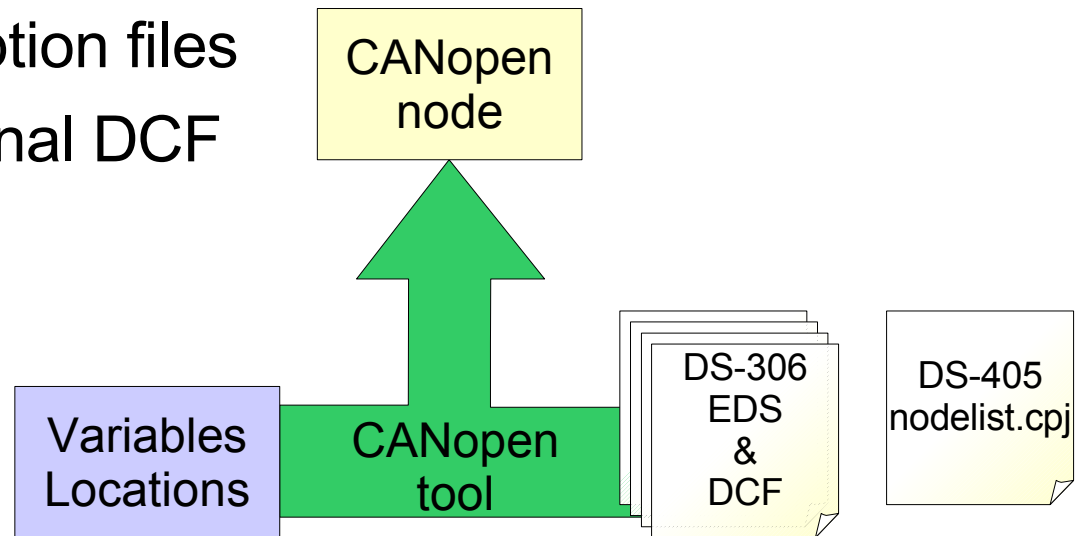


=>



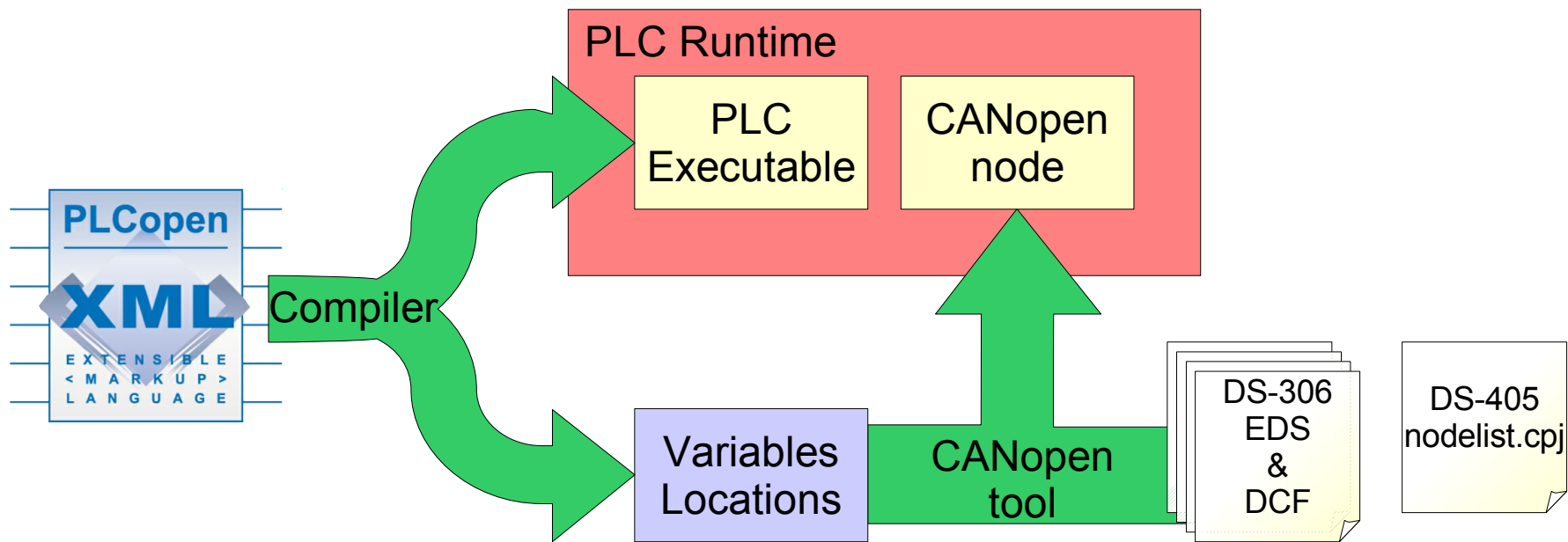
Implementation

- CANopen configuration tool
 - Change default variables location calculation
 - Generate PLC node configuration from
 - declared variable locations
 - network description files
 - optional additional DCF



Implementation

- PLC workbench
 - Extract variable locations at compile time
 - Pass locations list to CANopen configuration tool

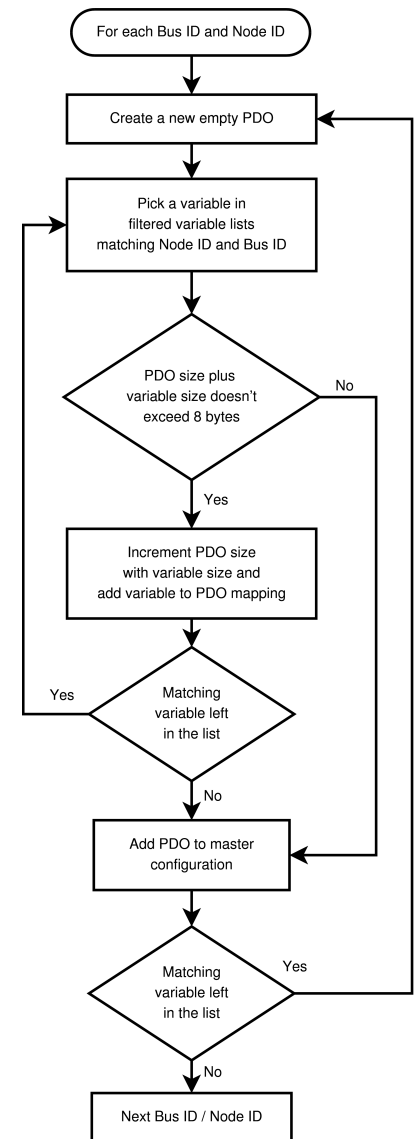


PDO mapping

- Remote variables access are mapped to PDO
- Establishing arbitrary PDO mapping implies to:
 - be a master node
 - slaves can only access/declare variables local OD
 - define new PDO mapping in each accessed node
 - may be stored in master's concise DCF
 - declare corresponding mapped PLC's OD entries
 - define corresponding PDO mapping in PLC's OD

PDO mapping

- More than one mapping is possible for one set of locations
 - Some side effect may appear on some implementations
- Some algorithm may be defined to fix:
 - Mapped variable layout in node's PDOs
 - Pre-configured PDO remapping politic
 - Mapped variables order in master OD



Ergonomic thoughts

- Compensate hard to read representation
 - Hexadecimal display option
 - %IB503254.0.16.1.24576.1 => %IB503254.0.10h.1.6000h.01h
 - Tooltip texts showing CANopen OD entry name
 - point and click variable to jump to configuration tool
- Smart manipulation of locations
 - variables substitution across programs
 - programs and EDS import and re-mapping

Questions

