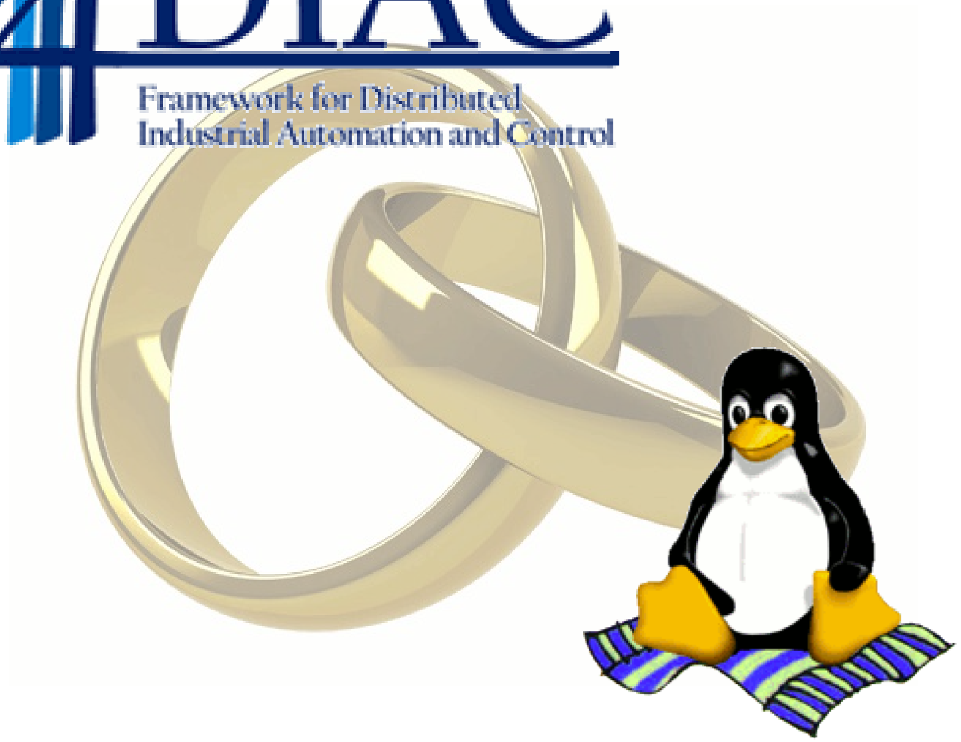
On Adding to 4DIAC Better Support for the IEC61131-3 Languages

4DIAC
Framework for Distributed Industrial Automation and Control



Mário de Sousa

U. PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

(Uni. Porto - Portugal)
(Engineering Faculty)

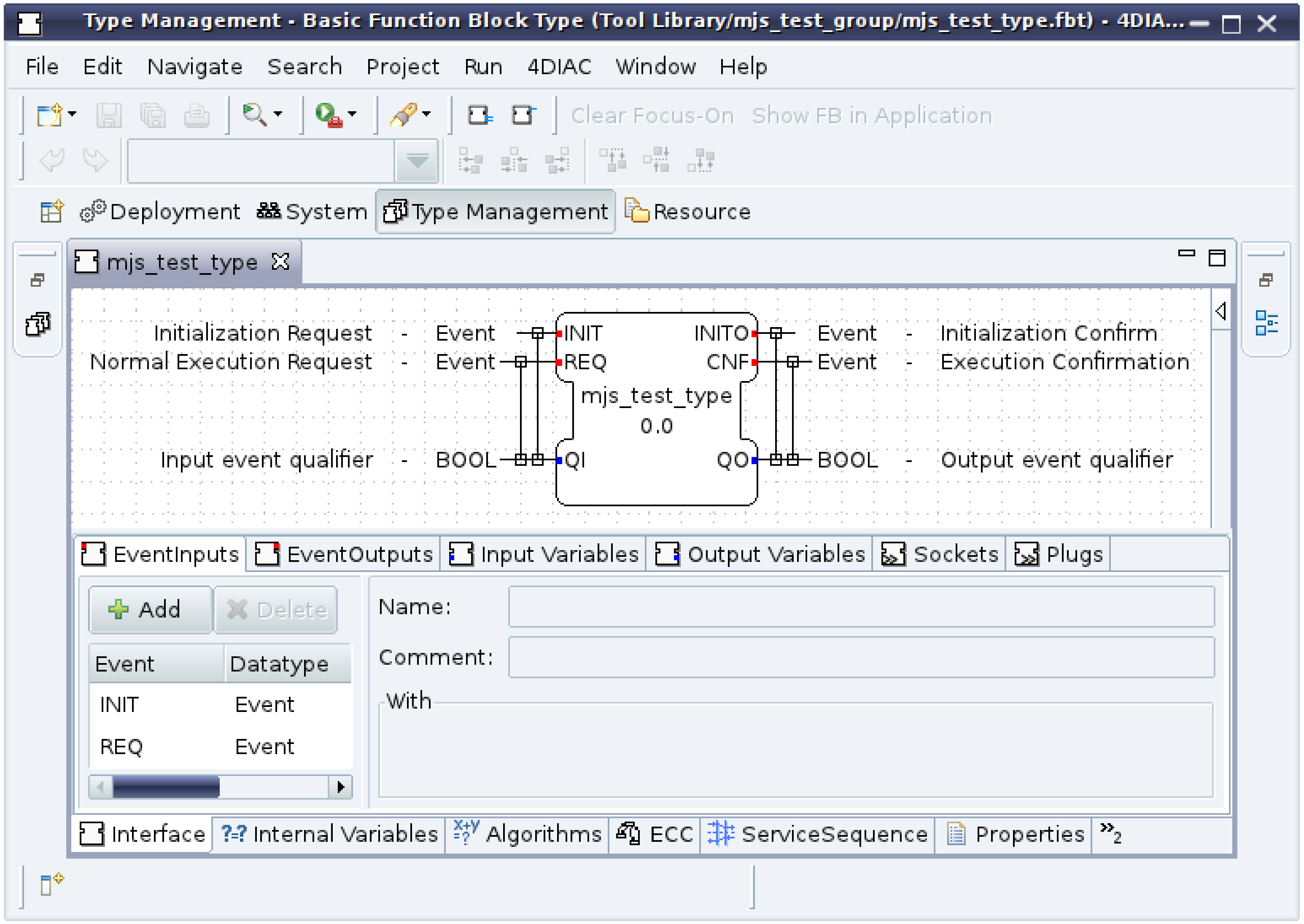
What is the problem?
(a.k.a. motivation)

4DIAC supports IEC 61131-3 **ST** programming language, but:

- Does not do semantic verification
- Does not allow calling standard Functions nor FBs

4DIAC does not support IEC 61131-3 **IL** language!

Examples...



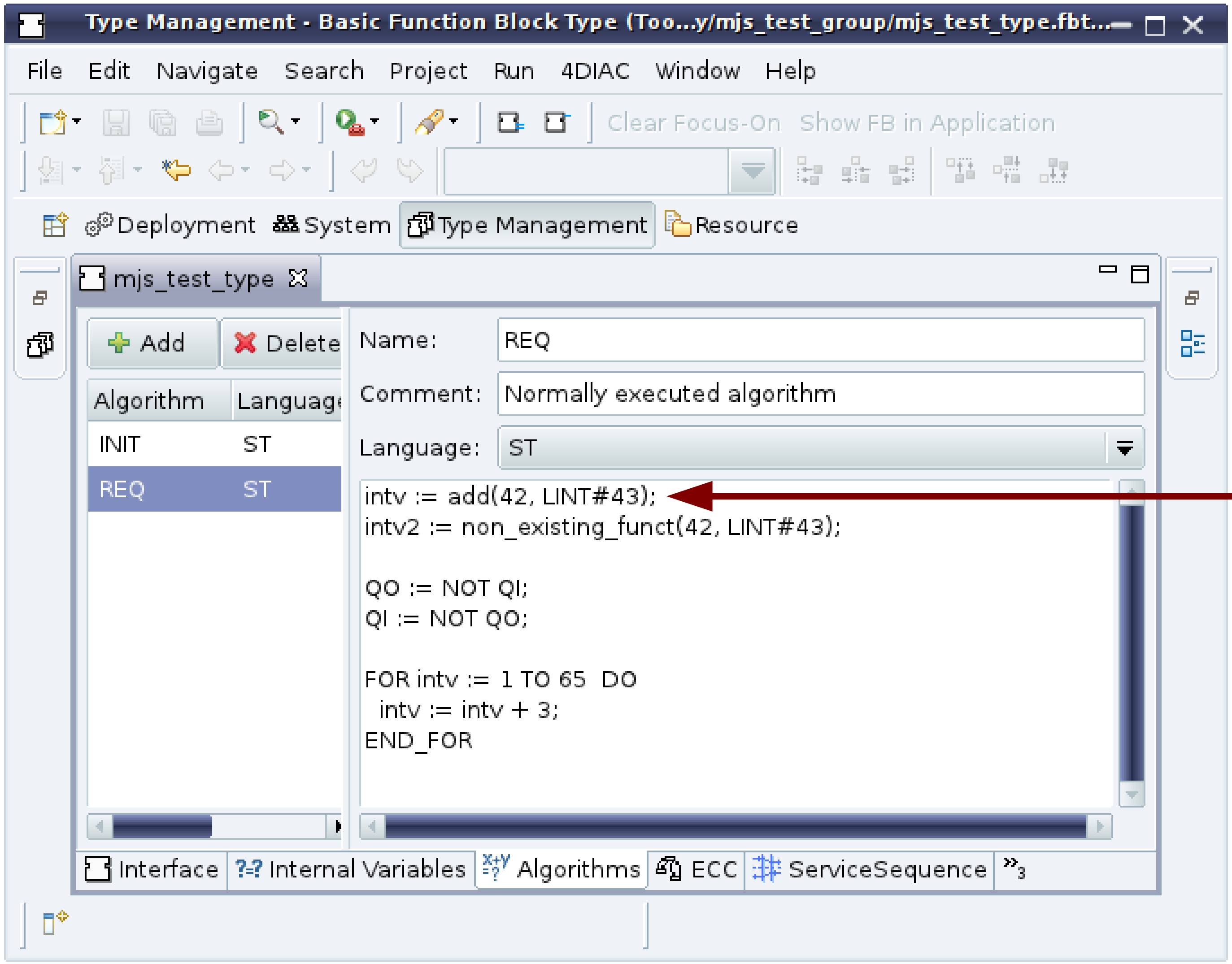
Type Management - Basic Function Block Type (Tool Library/mjs_test_group/mjs_test_type.fbt) - 4DIA...
File Edit Navigate Search Project Run 4DIAC Window Help
Clear Focus-On Show FB in Application
Deployment System Type Management Resource
mjs_test_type
Initialization Request - Event INIT
Normal Execution Request - Event REQ
Input event qualifier - BOOL QI
INITO Event - Initialization Confirm
CNF Event - Execution Confirmation
QO BOOL - Output event qualifier
mjs_test_type
0.0
EventInputs EventOutputs Input Variables Output Variables Sockets Plugs
Add Delete
Name:
Comment:
With
Interface Internal Variables Algorithms ECC ServiceSequence Properties

Event	Datatype
INIT	Event
REQ	Event

Just a simple 61499 Basic FB,

With an additional intv: INT Internal variable.

Examples...


Type Management - Basic Function Block Type (Too...y/mjs_test_group/mjs_test_type.fbt...
File Edit Navigate Search Project Run 4DIAC Window Help
Clear Focus-On Show FB in Application
Deployment System Type Management Resource
mjs_test_type
Add Delete
Interface Internal Variables Algorithms ECC ServiceSequence

Algorithm	Language
INIT	ST
REQ	ST

Name: REQ

Comment: Normally executed algorithm

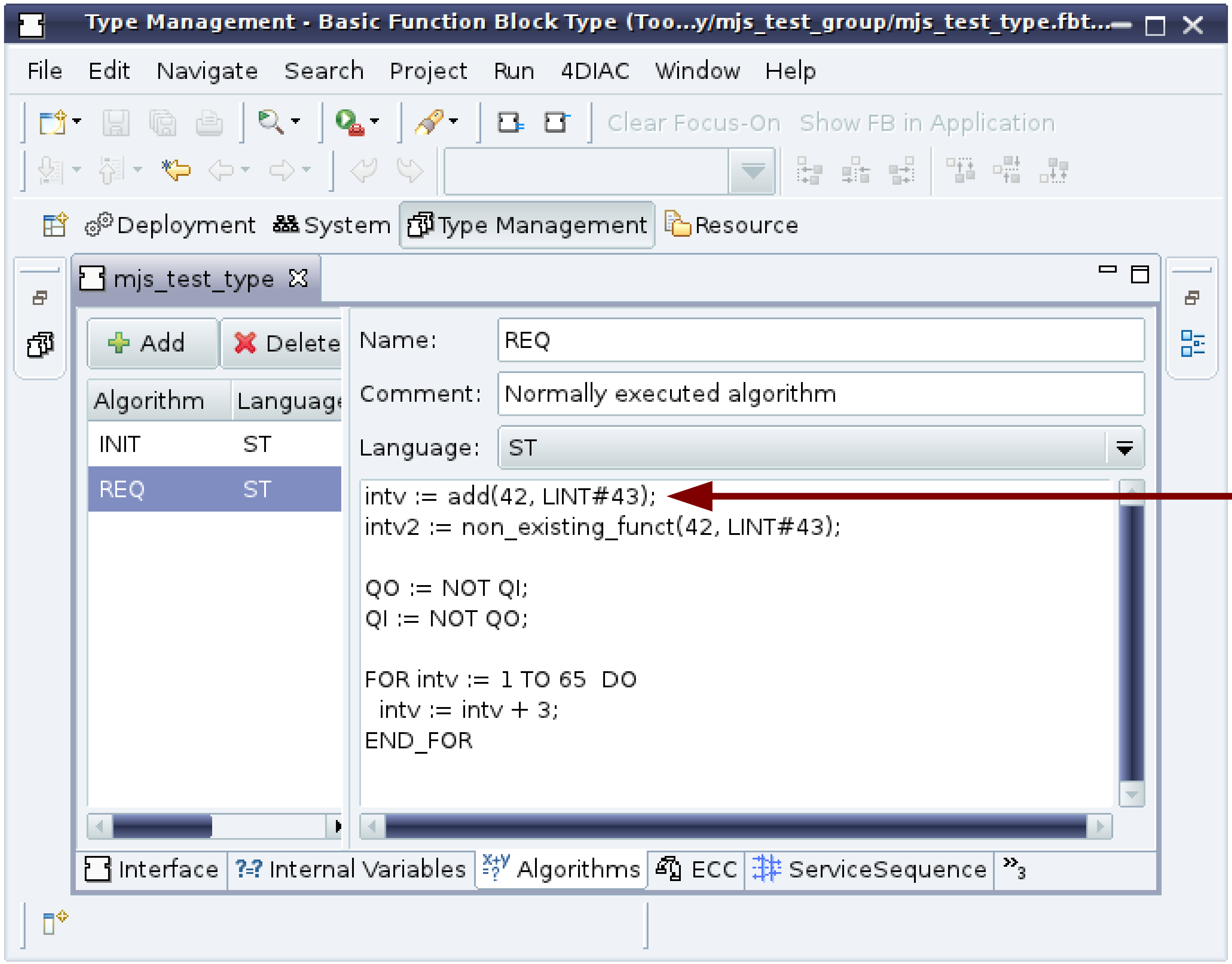
Language: ST

```
intv := add(42, LINT#43);
intv2 := non_existing_funct(42, LINT#43);

QO := NOT QI;
QI := NOT QO;

FOR intv := 1 TO 65  DO
  intv := intv + 3;
END_FOR
```

Use of literal with explicit data type LINT#43

Legal in ST code, but not supported in 4DIAC.

Examples...


Type Management - Basic Function Block Type (Too...y/mjs_test_group/mjs_test_type.fbt...
File Edit Navigate Search Project Run 4DIAC Window Help
Clear Focus-On Show FB in Application
Deployment System Type Management Resource
mjs_test_type
Add Delete
Interface Internal Variables Algorithms ECC ServiceSequence

Algorithm	Language
INIT	ST
REQ	ST

Name: REQ

Comment: Normally executed algorithm

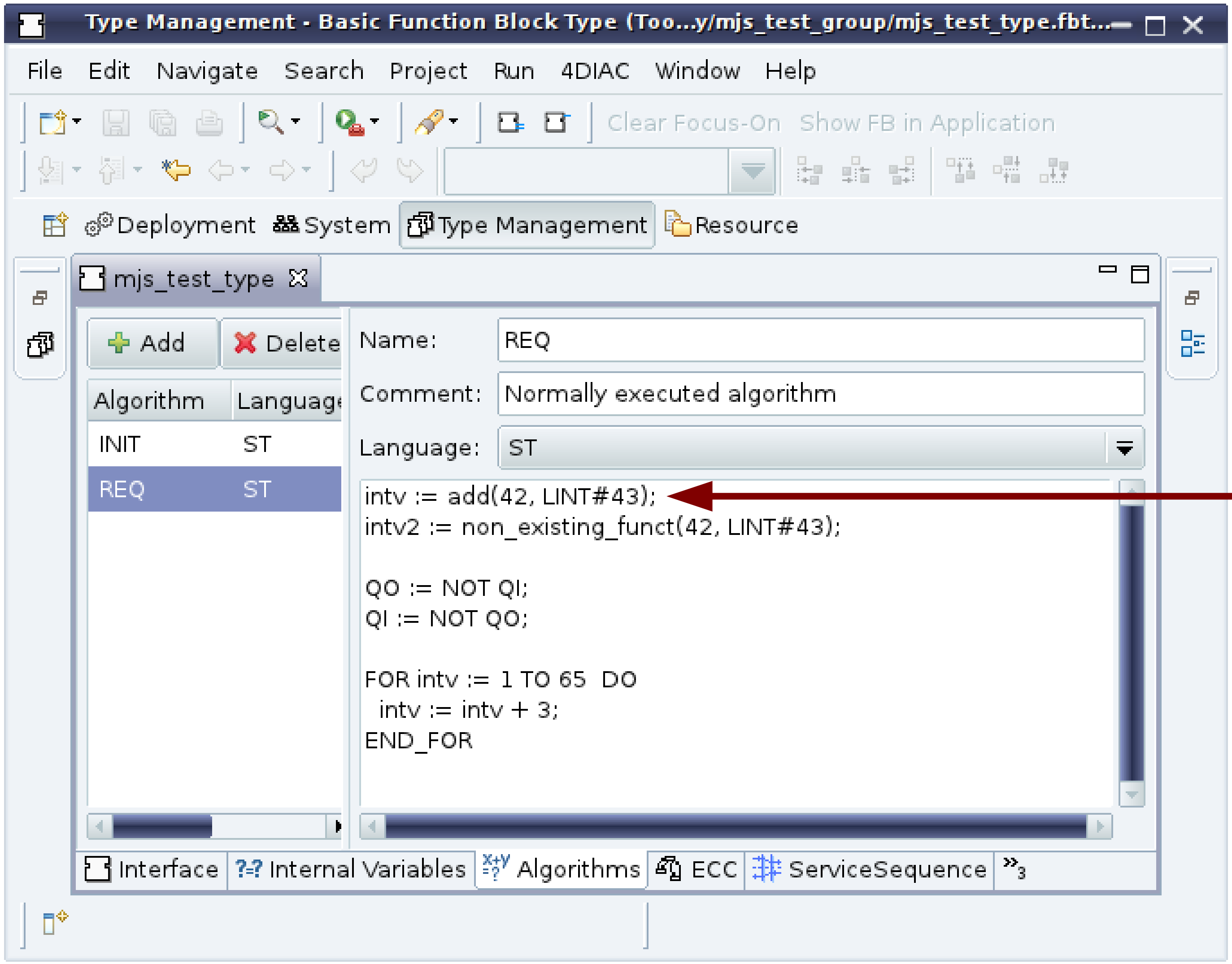
Language: ST

```
intv := add(42, LINT#43);
intv2 := non_existing_funct(42, LINT#43);

QO := NOT QI;
QI := NOT QO;

FOR intv := 1 TO 65  DO
  intv := intv + 3;
END_FOR
```

Calling the standard function add()

Legal in ST code, but not supported in 4DIAC.

Examples...


Type Management - Basic Function Block Type (Too...y/mjs_test_group/mjs_test_type.fbt...
File Edit Navigate Search Project Run 4DIAC Window Help
Clear Focus-On Show FB in Application
Deployment System Type Management Resource
mjs_test_type
Add Delete
Interface ?-? Internal Variables Algorithms ECC ServiceSequence

Algorithm	Language
INIT	ST
REQ	ST

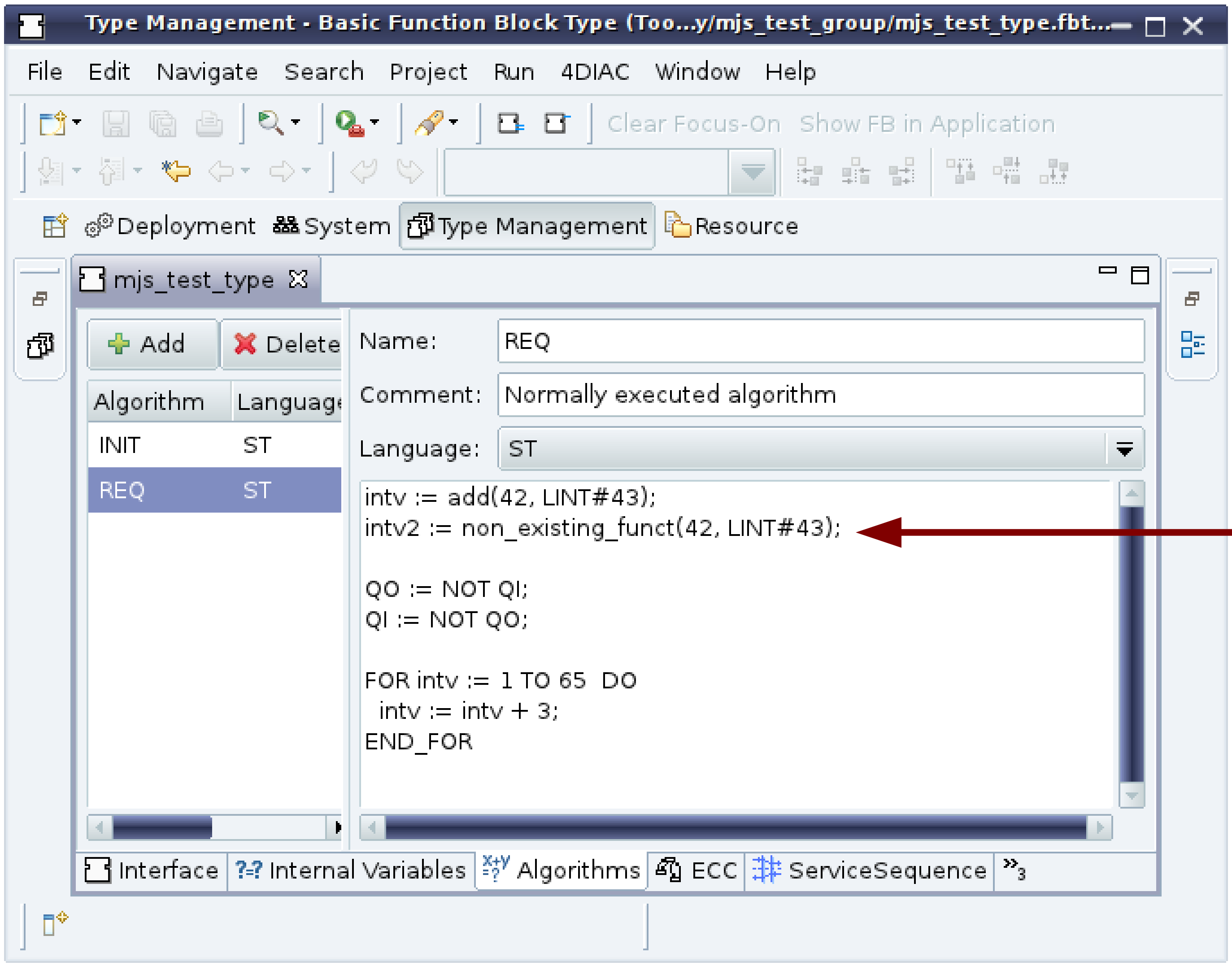
Name: REQ
Comment: Normally executed algorithm
Language: ST

```
intv := add(42, LINT#43);
intv2 := non_existing_funct(42, LINT#43);

QO := NOT QI;
QI := NOT QO;

FOR intv := 1 TO 65  DO
  intv := intv + 3;
END_FOR
```

Attribution of LINT data to an INT variable.

Illegal in ST code.

Examples...


Type Management - Basic Function Block Type (Too...y/mjs_test_group/mjs_test_type.fbt...
File Edit Navigate Search Project Run 4DIAC Window Help
Clear Focus-On Show FB in Application
Deployment System Type Management Resource
mjs_test_type
Add Delete
Interface ?:? Internal Variables Algorithms ECC ServiceSequence »3

Algorithm	Language
INIT	ST
REQ	ST

Name: REQ

Comment: Normally executed algorithm

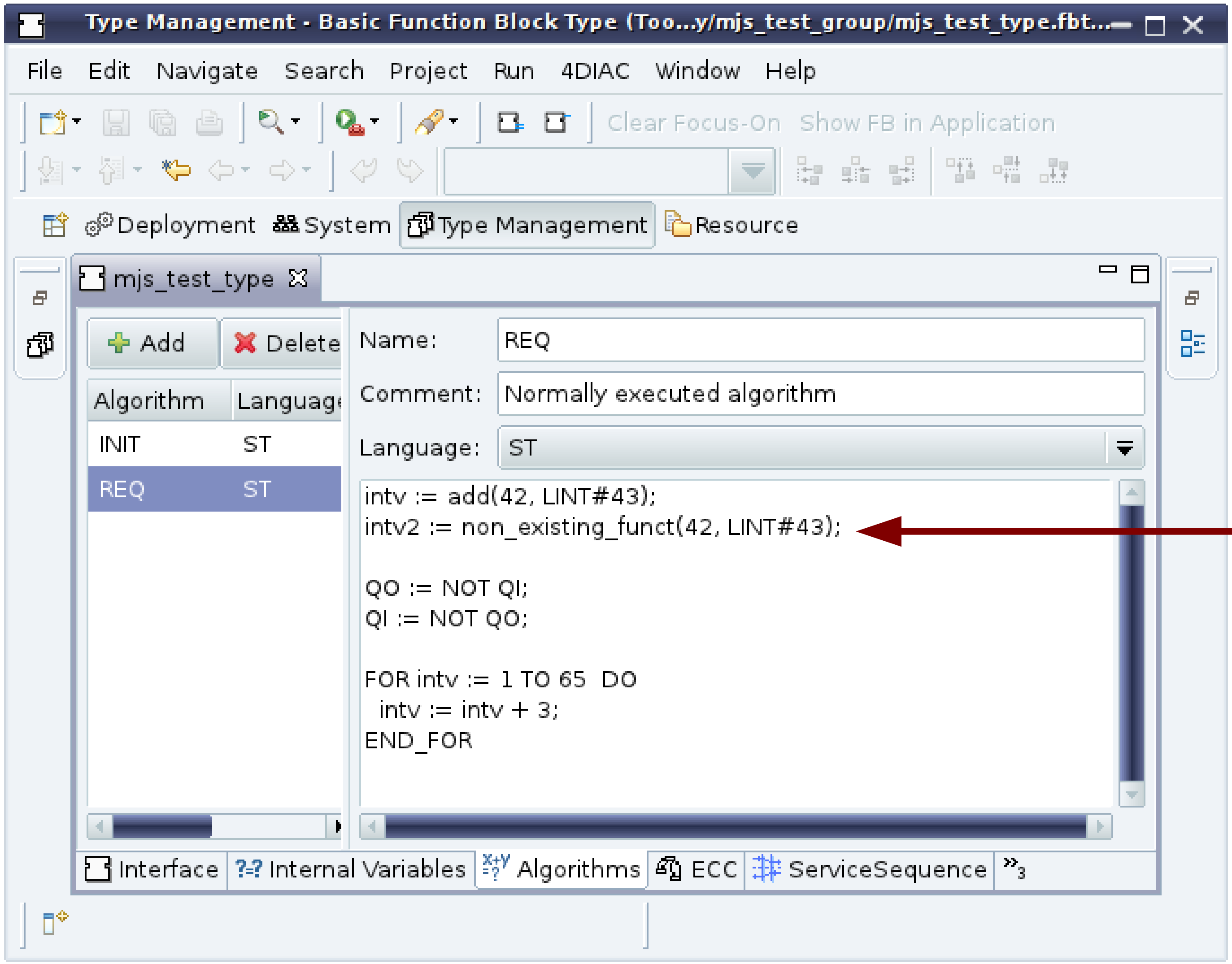
Language: ST

```
intv := add(42, LINT#43);
intv2 := non_existing_funct(42, LINT#43);

QO := NOT QI;
QI := NOT QO;

FOR intv := 1 TO 65  DO
  intv := intv + 3;
END_FOR
```

Calling a non existing function

Illegal in ST code.

Examples...


Type Management - Basic Function Block Type (Too...y/mjs_test_group/mjs_test_type.fbt...
File Edit Navigate Search Project Run 4DIAC Window Help
Clear Focus-On Show FB in Application
Deployment System Type Management Resource
mjs_test_type
Add Delete
Interface Internal Variables Algorithms ECC ServiceSequence

Algorithm	Language
INIT	ST
REQ	ST

Name: REQ

Comment: Normally executed algorithm

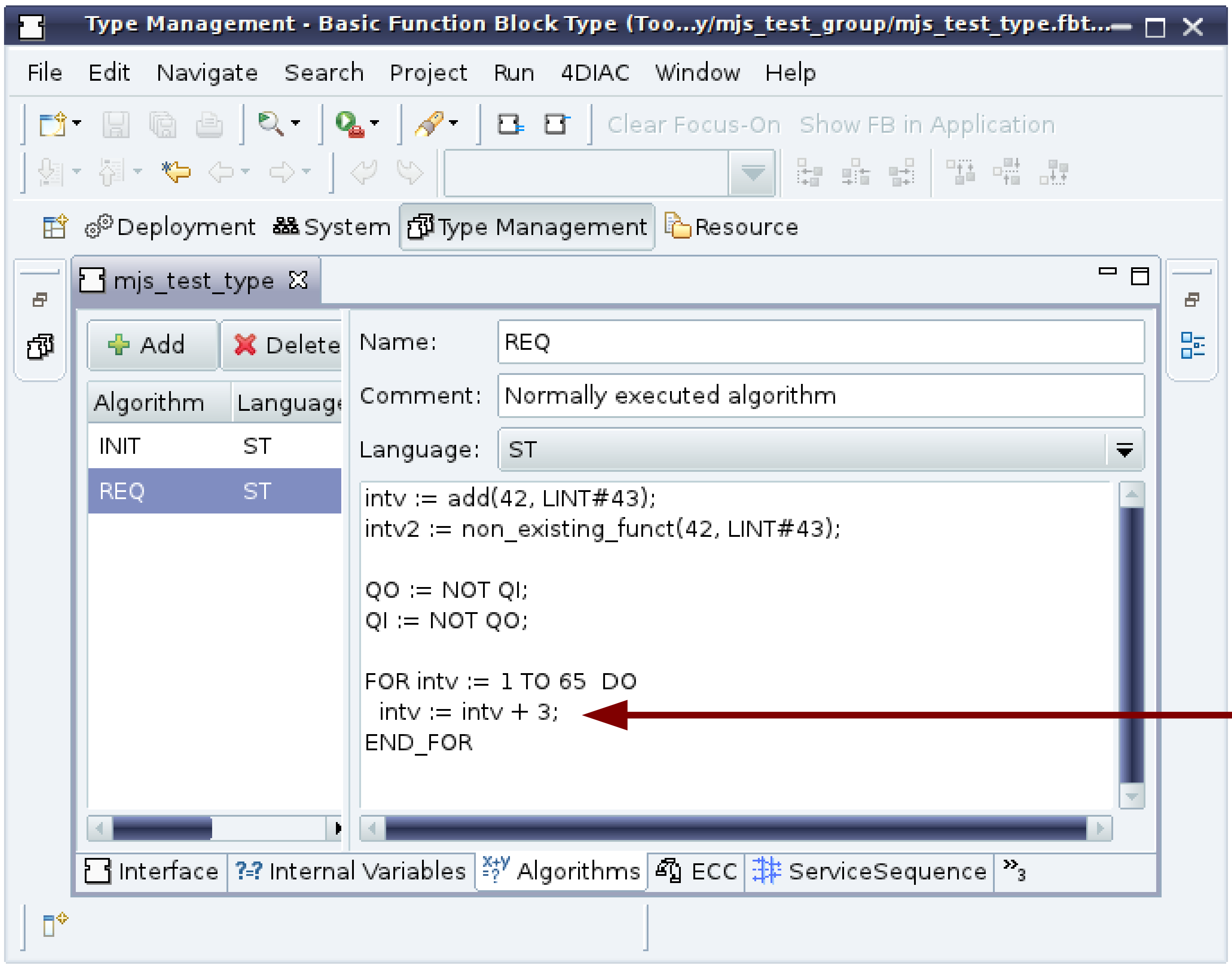
Language: ST

```
intv := add(42, LINT#43);
intv2 := non_existing_funct(42, LINT#43);

QO := NOT QI;
QI := NOT QO;

FOR intv := 1 TO 65  DO
  intv := intv + 3;
END_FOR
```

Writing to non existing variable intv2.

Illegal in ST code.

Examples...


Type Management - Basic Function Block Type (Too...y/mjs_test_group/mjs_test_type.fbt...
File Edit Navigate Search Project Run 4DIAC Window Help
Clear Focus-On Show FB in Application
Deployment System Type Management Resource
mjs_test_type
Add Delete
Interface Internal Variables Algorithms ECC ServiceSequence

Algorithm	Language
INIT	ST
REQ	ST

Name: REQ

Comment: Normally executed algorithm

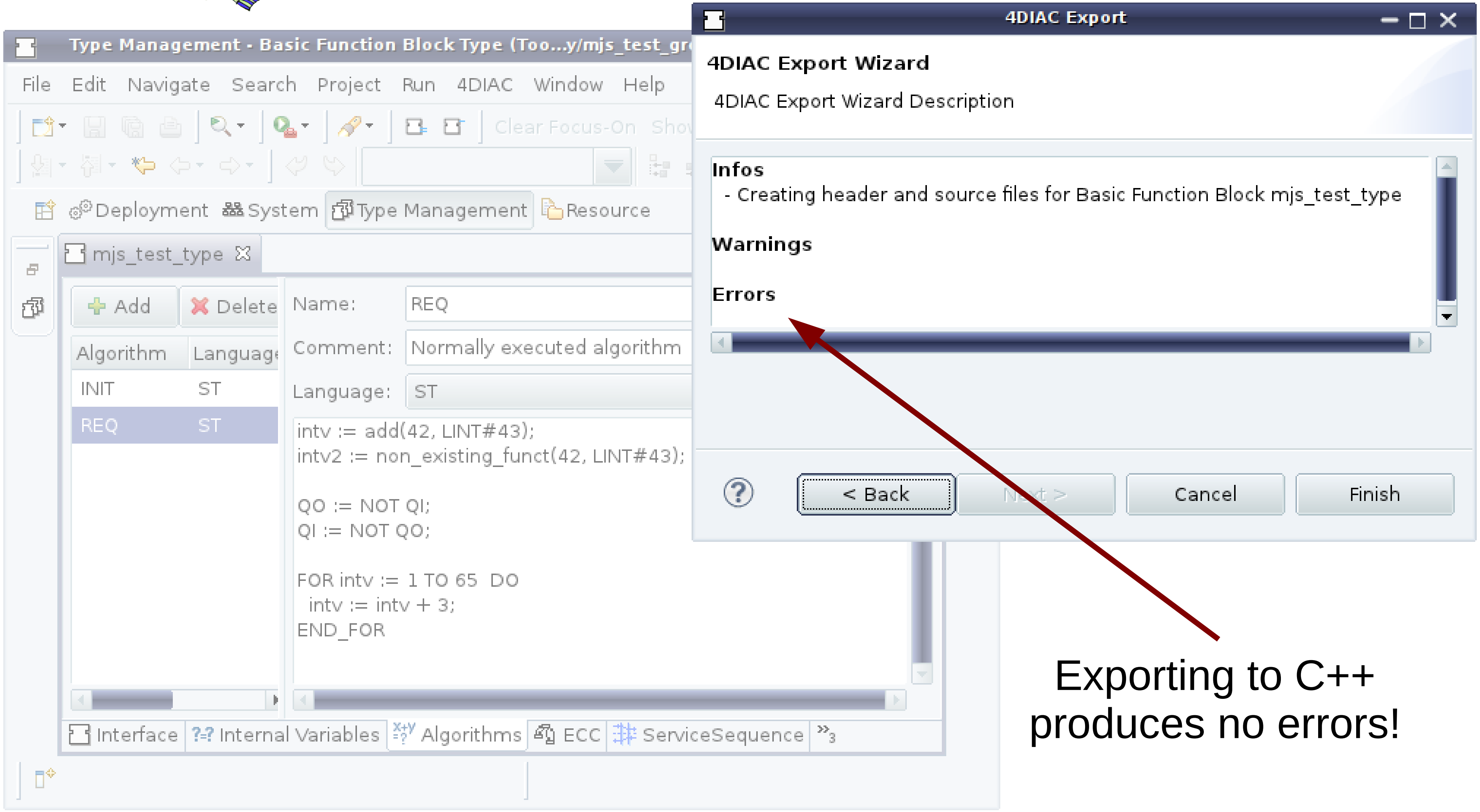
Language: ST

```
intv := add(42, LINT#43);
intv2 := non_existing_funct(42, LINT#43);

QO := NOT QI;
QI := NOT QO;

FOR intv := 1 TO 65  DO
  intv := intv + 3;
END_FOR
```

Writing to a loop control variable.

Illegal in ST code.

Examples...


Type Management - Basic Function Block Type (Too...y/mjs_test_gr

File Edit Navigate Search Project Run 4DIAC Window Help
Clear Focus-On
Deployment System Type Management Resource
mjs_test_type
Add Delete
Interface Internal Variables Algorithms ECC ServiceSequence

Algorithm	Language
INIT	ST
REQ	ST

Name: REQ
Comment: Normally executed algorithm
Language: ST

```
intv := add(42, LINT#43);
intv2 := non_existing_funct(42, LINT#43);

QO := NOT QI;
QI := NOT QO;

FOR intv := 1 TO 65  DO
  intv := intv + 3;
END_FOR
```

4DIAC Export

4DIAC Export Wizard

4DIAC Export Wizard Description

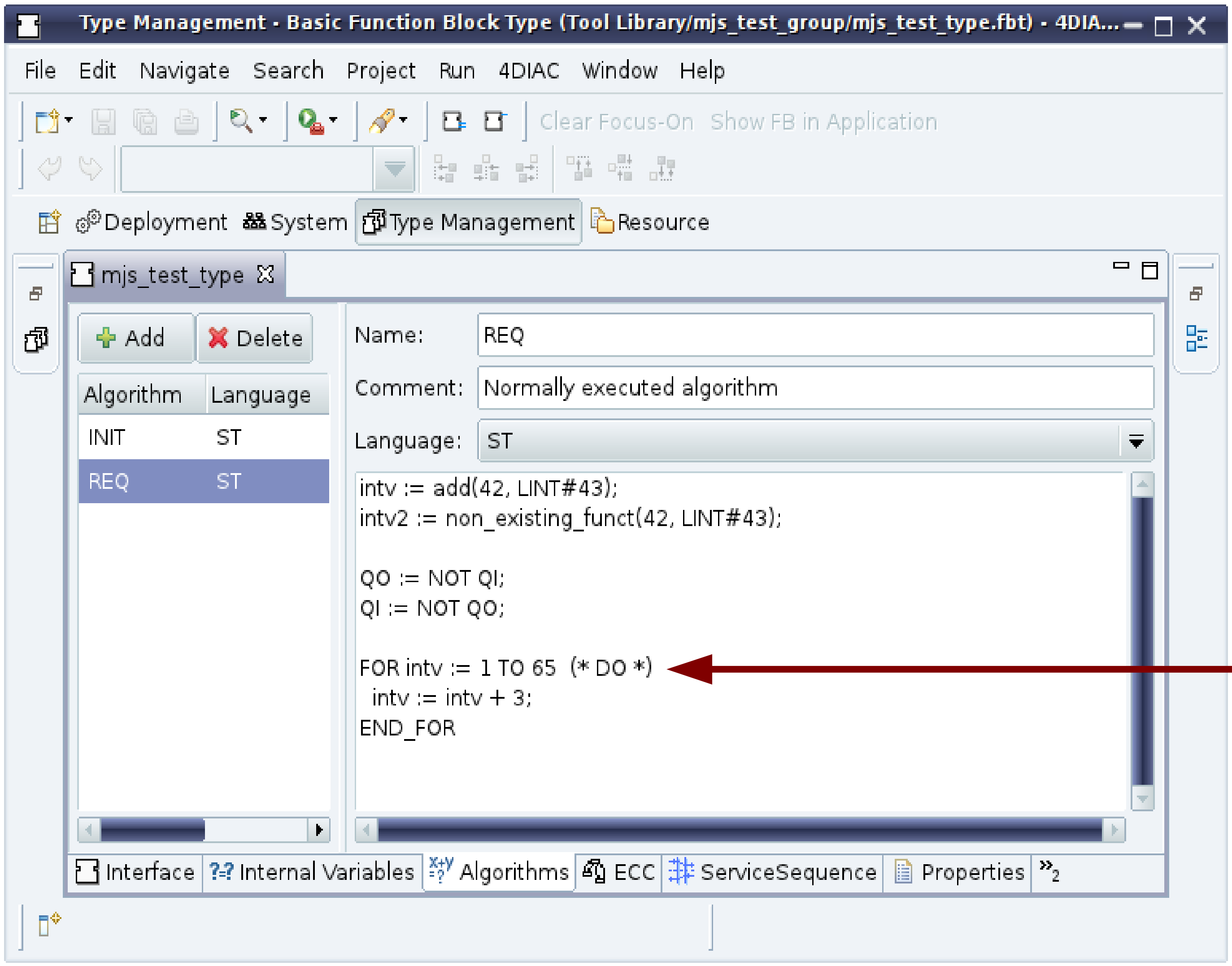
Infos
- Creating header and source files for Basic Function Block mjs_test_type

Warnings

Errors

Exporting to C++ produces no errors!

Examples...


Type Management - Basic Function Block Type (Tool Library/mjs_test_group/mjs_test_type.fbt) - 4DIA...
File Edit Navigate Search Project Run 4DIAC Window Help
Clear Focus-On Show FB in Application
Deployment System Type Management Resource
mjs_test_type
Add Delete
Interface Internal Variables Algorithms ECC ServiceSequence Properties

Algorithm	Language
INIT	ST
REQ	ST

Name: REQ

Comment: Normally executed algorithm

Language: ST

```
intv := add(42, LINT#43);
intv2 := non_existing_funct(42, LINT#43);

QO := NOT QI;
QI := NOT QO;

FOR intv := 1 TO 65  (* DO *)
  intv := intv + 3;
END_FOR
```

If we add a syntax error to ST code...

Exporting simply skips the 'confirmation' window, and no error is reported.

How can we fix this?

The easiest solution seems to be to use an existing IEC 61131-3 compiler ---> matiec

- Open source (GPL v3)
- May be executed as an external tool, or from the 'code export script' (eventual license incompatibility is no longer an issue)
- Supports IEC 61131-3 ST, IL and SFC (in textual format).
- Already supports most of the necessary semantic checks:
  - **Data type checking**
  - **Constant folding**
  - **...**

Example - continued

- Let's place the ST code inside a 61131-3 function
- and run it through matiec...

```
 1 FUNCTION FBtypename_algorithmname : BOOL
 2   VAR_INPUT    QI : BOOL;   END_VAR
 3   VAR_OUTPUT   QO : BOOL;   END_VAR
 4   VAR        intv : INT;    END_VAR
 5
 6   (* body *)
 7   intv  := add(42, LINT#43);
 8   intv2 := non_existing_func(42, LINT#43)
 9
10   QO := NOT QI;
11   QI := NOT QO;
12
13   FOR intv := 1 to 65 BY 6 DO
14     intv := intv + 3;
15   END_FOR
16
17 END_FUNCTION
```

```
test/iec61499.st:8: error:
invalid variable before ':=' in
ST assignment statement.

test/iec61499.st:8: error:
';' missing at the end of
statement in ST statement.

test/iec61499.st:8: error:
invalid statement in ST
statement.
```

Example - continued

- Let's place the ST code inside a 61131-3 function
- and run it through matiec...

```
 1 FUNCTION FBtypename_algorithmname : BOOL
 2   VAR_INPUT     QI : BOOL;   END_VAR
 3   VAR_OUTPUT    QO : BOOL;   END_VAR
 4   VAR         intv : INT;    END_VAR
 5
 6    (* body *)
 7   intv  := add(42, LINT#43);
 8   intv  := sub(42, LINT#43);
 9
10   QO := NOT QI;
11   QI := NOT QO;
12
13   FOR intv := 1 to 65 BY 6 DO
14     intv := intv + 3;
15   END_FOR
16
17 END_FUNCTION
```

```
test/iec61499.st:7-16..7-17:
error: Data type
incompatibility for value
passed in position 1 when
invoking function 'add'

test/iec61499.st:7-20..7-26:
error: Data type
incompatibility for value
passed in position 2 when
invoking function 'add'

test/iec61499.st:7-11..7-27:
error: Incompatible data types
for ':=' operation.
```

Example - continued

- Let's place the ST code inside a 61131-3 function

```
 1 FUNCTION FBtypename_algorithmname : BOOL
 2   VAR_INPUT     QI : BOOL;   END_VAR
 3   VAR_OUTPUT    QO : BOOL;   END_VAR
 4   VAR         intv : INT;    END_VAR
 5
 6   (* body *)
 7   intv  := add(42, LINT#43);
 8   intv  := sub(42, LINT#43);
 9
10   QO := NOT QI;
11   QI := NOT QO;
12
13   FOR intv := 1 to 65 BY 6 DO
14     intv := intv + 3;
15   END_FOR
16
17 END_FUNCTION
```

- and run it through matiec...

```
test/iec61499.st:8-16..8-17:
error: Data type
incompatibility for value
passed in position 1 when
invoking function 'sub'

test/iec61499.st:8-20..8-26:
error: Data type
incompatibility for value
passed in position 2 when
invoking function 'sub'

test/iec61499.st:8-5..8-27:
error: Incompatible data types
for ':=' operation.

test/iec61499.st:14-6..14-9:
error: Assignment to FOR
control variable is not
allowed.
```

Example - continued

- Let's place the ST code inside a 61131-3 function...
  …and compile it with matiec

```
FUNCTION FBtypename_algorithmname : BOOL
  VAR_INPUT     QI : BOOL;   END_VAR
  VAR_OUTPUT    QO : BOOL;   END_VAR
  VAR         intv : INT;    END_VAR

   (* body *)
  intv   := add(42, 43);
  intv   := sub(42, 43);

  QO := NOT QI;
  QI := NOT QO;

  FOR intv := 1 to 65 BY 6 DO
    QO := NOT QI;
  END_FOR

END_FUNCTION
```

Example - continued

- Let's place the ST code inside a 61131-3 function...
  …and compile it with matiec. The result is...

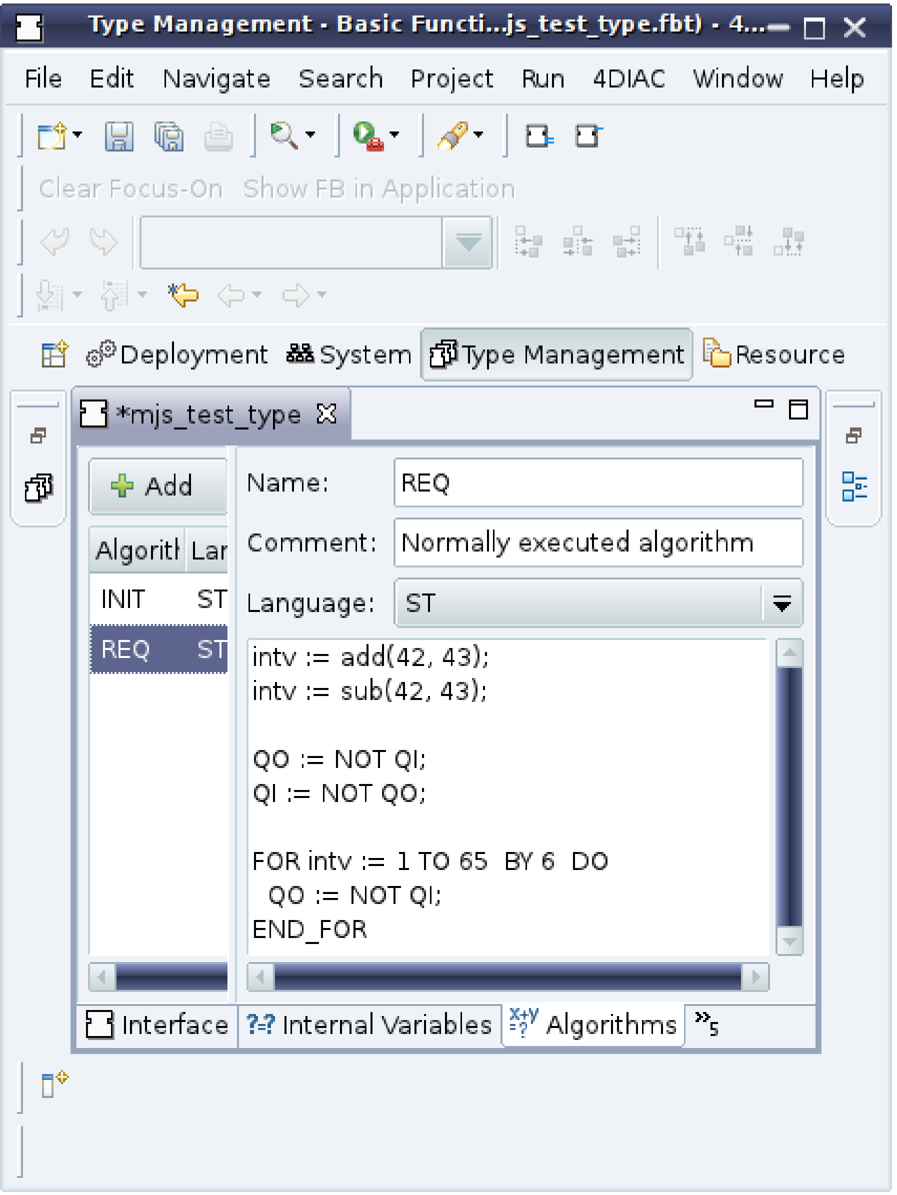
```
BOOL FBTYPENAME_ALGORITHMNAME(BOOL EN, BOOL *__ENO, BOOL QI, BOOL *__QO) {
  BOOL ENO = __BOOL_LITERAL(TRUE);
  BOOL QO = __BOOL_LITERAL(FALSE);
  INT INTV = 0;
  BOOL FBTYPENAME_ALGORITHMNAME = __BOOL_LITERAL(FALSE);

  // Control execution
  if (!EN) {
    if (__ENO != NULL) {*__ENO = __BOOL_LITERAL(FALSE);}
    return FBTYPENAME_ALGORITHMNAME;
  }
  // Body
  ...

__end:
  if (__ENO != NULL) {*__ENO = ENO;}
  if (__QO  != NULL) {*__QO  = QO; }
  return FBTYPENAME_ALGORITHMNAME;
}
```

Example - continued

- Let's place the ST code inside a 61131-3 function...
  …and compile it with matiec. The result is...

```
// Body
INTV = ADD__INT__INT((BOOL)__BOOL_LITERAL(TRUE), NULL,
                     (UINT)2, (INT)42, (INT)43);
INTV = SUB__INT__INT__INT((BOOL)__BOOL_LITERAL(TRUE), NULL,
                     (INT)42, (INT)43);

QO = !(QI);
QI = !(QO);

for(INTV = 1; ((6) > 0)? (INTV <= (65)) : (INTV >= (65)); INTV += (6)) {
  QO = !(QI);
};
```

Examples...


Type Management - Basic Functi...js_test_type.fbt) - 4...
File Edit Navigate Search Project Run 4DIAC Window Help
Clear Focus-On Show FB in Application
Deployment System Type Management Resource
*mjs_test_type
Add
Interface ?-? Internal Variables Algorithms »5

Algorit	Lar
INIT	ST
REQ	ST

Name: REQ
Comment: Normally executed algorithm
Language: ST

```
intv := add(42, 43);
intv := sub(42, 43);

QO := NOT QI;
QI := NOT QO;

FOR intv := 1 TO 65  BY 6  DO
  QO := NOT QI;
END_FOR
```

- Exported by 4DIAC results in...

```
void FORTE_mjs_test_type::alg_REQ(void){
intv() = add((42), (43));
intv() = sub((42), (43));

QO() = !QI();
QI() = !QO();

  {
   bool isintv_Up = ((6) > 0);
   intv() = 1;
   while(!(((isintv_Up) && (intv() > (65))) ||
           ((!isintv_Up) && (intv() < (65))))){
      QO() = !QI();

      if(((isintv_Up) && ((6) > 0)) ||
         ((!isintv_Up) && ((6) < 0))) {
        intv() = intv() + (6);
      } else {
        intv() = intv() - (6);
      }
    }
  }
}
```

Mário de Sousa – 2012/09/21

Examples...

How can we merge these two code snippets ??

Example - continued

- Matiec supports both extensible and overloaded functions.
  (only for standard functions, e.g. add(1.1, 2.2), add(1,2), add(1,2,3,4))
- Because of the way the calling the above functions are handled/called, it is best to let matiec generate the source code for the body.
- We need to change matiec so that:
  - Variable names are not printed in capitals
  - Variable names are printed followed by ' () '
  - Only the 'body' is generated.

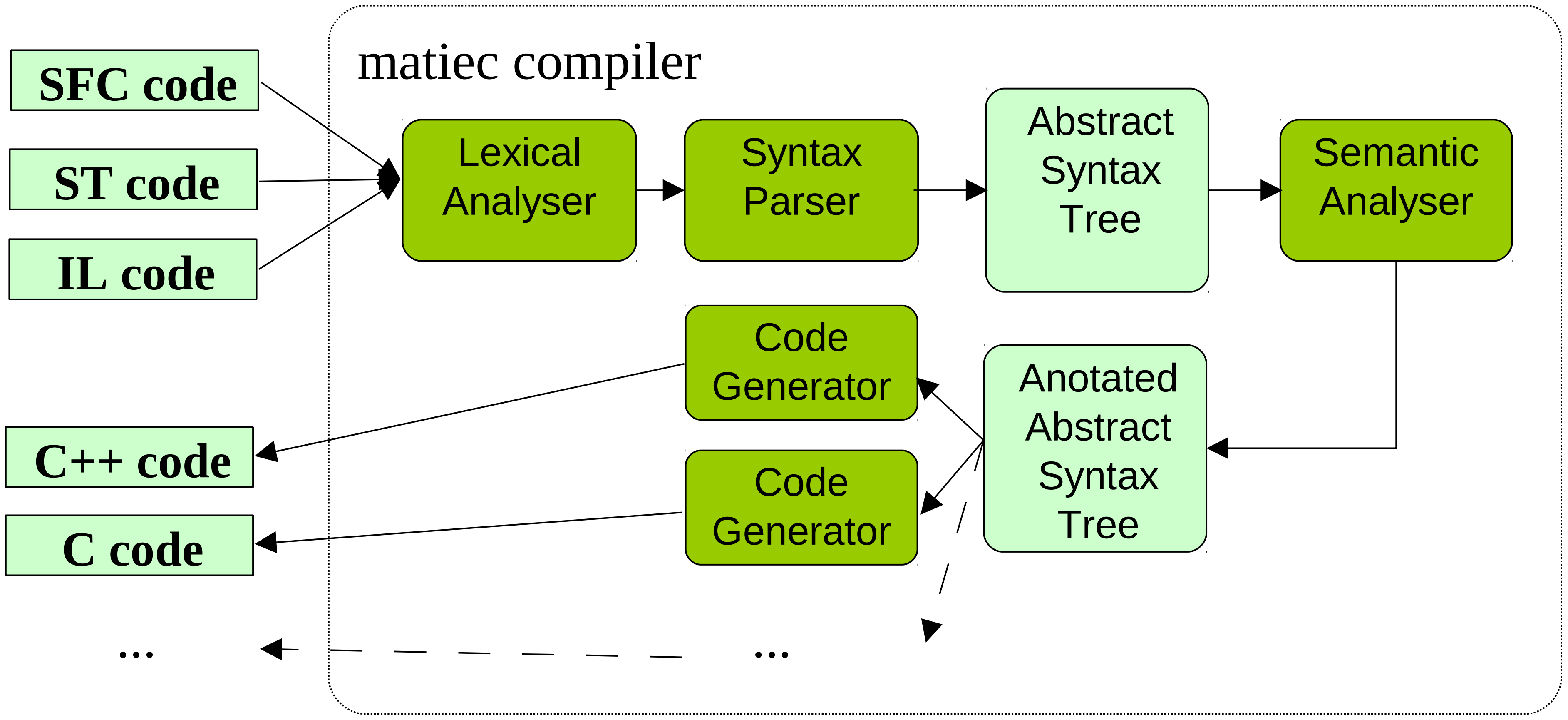
```
// Body
intv()=ADD__INT__INT((BOOL)__BOOL_LITERAL(TRUE),NULL,(UINT)2, (INT)42, (INT)43);
intv()=SUB__INT__INT__INT((BOOL)__BOOL_LITERAL(TRUE),NULL, (INT)42, (INT)43);
QO() = !(QI());
QI() = !(QO());
for(intv() = 1; ((6) > 0)? (intv() <= (65)) : (intv() >= (65)); intv() += (6)) {
  QO() = !(QI());
};
```

Example - continued

- We need to change 4DIAC so that:
  - It uses the 'body' code generated by matiec.
  - The elementary data types INT, SINT, etc.. are defined
  - The standard function library that comes with matiec is included and linked.

```
void FORTE_mjs_test_type::alg_REQ(void){
// Body
intv()=ADD__INT__INT((BOOL)__BOOL_LITERAL(TRUE),NULL,(UINT)2, (INT)42, (INT)43);
intv()=SUB__INT__INT__INT((BOOL)__BOOL_LITERAL(TRUE),NULL, (INT)42, (INT)43);
QO() = !(QI());
QI() = !(QO());
for(intv() = 1; ((6) > 0)? (intv() <= (65)) : (intv() >= (65)); intv() += (6)) {
  QO() = !(QI());
};
```

Matiec...


SFC code
ST code
IL code
matiec compiler
Lexical Analyser
Syntax Parser
Abstract Syntax Tree
Semantic Analyser
Code Generator
Anotated Abstract Syntax Tree
Code Generator
C++ code
C code
...
...

Matiec...

- Lexical Parser → flex
- Syntax Analyser → bison
- Semantic Analyser → C++ code (visitor pattern)
  - **flow_control_analysis**
  - **constant_folding (constant propagation still missing)**
  - **type_safety**
  - **lvalue_check**
  - **array_range_check**
- Code Generator → C++ code (visitor pattern)
  - **C code generator**
  - **IEC 61131-3 code generator (for debugging purposes only)**

Matiec...

- Constant Folding
  - Determine the result of every expression in which only constant values are used.
  - Every entry in the abstract syntax tree that represents a fixed (constant) value gets annotated with the result. (e.g.: boolv := (314159 / 42) = 666;)
  - We do not yet support constant values of enumeration data types (maybe in the near future?).
  - We do not yet support constant propagation. (e.g.: x := 42 + 1;   y := x -3; → value of y is constant)

Matiec...

- Flow Control Analysis
  - **Analyse the possible control flow when the program executes**
  - **ST code is rather straight forward**
    - check boolean values of 'if', 'while', 'repeat', and possible constant values of 'for' and 'case', to find unreachable code.
  - **IL is more complex –**
    - analyse JMP, JMPC, and JMPCN instructions to determine all the entry points for each labeled instruction (take into account constant values when considering JMPC and JMPCN).
    - Find any labeled instruction with no entry points (i.e. unreachable code!)

Matiec...

- Type Safety
  - **Determine the data type of each expression/instruction**
  - **Since IEC 61131-3 allows overloaded functions that only differ in the returned data type (e.g. LEN(ANY_STRING): ANY_INT), we use the algorithm:**
    - Fill candidate data types
    - Narrow candidate data types
    - Print data type inconsistency error messages
  - **IL is more complex, since it may have JMP going forward and/or back.**
    - We need to run the above algorithm twice!

Matiec...

- Lvalue Check
  - **Check whether the 'values' (variables, really) on the 'left' hand side of expressions are valid**
    - Also consider variables passed to OUT & IN_OUT in Function/FB/Program invocations.
  - **'Left' hand values (variables) may not**
    - Have been declared CONSTANT
    - Be FOR loop control variables
    - Be OUTPUT variables, when directly accessing the variables of a FB using the syntax of a structured data type.
      (e.g. Timer1.Q := TRUE;)
    - Be an expression (may occur in function invocations, when passing values to IN_OUT parameters!)
      (e.g. foo(add(42, 4)) - & foo has a single IN_OUT param)

Matiec...

- Array Range Check
  - **Check whether the number of subscript values is correct**
    (e.g. A: ARRAY [1..3] OF INT;   … A[x, y] := 0)
  - **Check whether array subscript values fall within the allowed range (check only done for constant values in the indexes)**
    (e.g. A: ARRAY [1..3] OF INT;   … A[1+3] := 0)

My Questions

- Are you interested in this approach? Would it be helpful?
  - **Main advantages I can see...**
    - Adds support for IL
    - Adds support for standard IEC 61131-3 functions
    - Adds semantic/syntax error verification and error messages.
- Would anybody like to help me implement this approach?
  I will focus on matiec. Help mainly needed on 4DIAC side.
- Any other suggestion?

4DIAC
Framework for Distributed Industrial Automation and Control

Questions?

(preferably in English)

Kysymyksiä?

Questions?

Otázky?

Questions?

질문?

Въпроси?

質問ですか？

Spørgsmål?

Domande?

Vragen?

الأسئلة؟

Spørsmål?

Perguntas?

Pitanja?

¿Preguntas?

Fragen?

问题？

Ερωτήσεις;

Frågor?

Pytania?

Întrebări?

Вопросы?