KOLLMORGEN®

Because Motion Matters™

CNC AKC-PPC

PLC Programming Manual

SOUZA, RENATO

KOLLMORGEN
Because Motion Matters™

Contents

1.0 Introduction

To build the PLC program in the AKC-PPC CNC we will use the PLC editor of the AKC-PPC IDE. The programming language used for this is ST which is defined in the IEC61131-3 standard
There are many books and training on PLC Programming defined in the IEC61131-3 standard.
The purpose of this documentation is not to reproduce what is already widely defined in the standard and other publications, but to convey a basic understanding of the structured text language in the ACK- PPC IDE and AKC-PPC CNC environment.

2.0 ST Language

Structured text is the PLC programming language defined by "PLCOpen" in the IEC 61131-3 standard. The programming language is text-based, unlike "Ladder" which is graphical.
It may seem better to use a graphical programming language for PLC programming, but depending on the size of the program that will be developed this can be exactly the opposite.
For those who have programmed in high-level languages the structured text will seem familiar.
The syntax of structured text is designed to resemble the syntax of a high-level programming language with similar loops, variables, conditions, and operators.

2.1 Program Flow

PLC program execution has a well-defined peculiarity.
The PLC program runs as if it were an infinite loop program, that is, it runs once and at the end of the execution it returns to the beginning and does this again each time.
As long as the CNC is on, the PLC is active and always running.
In the CNC we have a parameter to determine the cycle time of the PLC program.
It is parameter 100 and is located in the "General" tab of the parameter editor. In this parameter we can choose the following execution times:

0 = 16ms
1 = 4ms
2 = 8ms

If when running the PLC program the CNC realizes that the time provided is not enough, a message will be displayed to identify this.
This issue can happen because of the size of the program developed, or because of some faulty logic.
This can be solved by increasing the runtime of the PLC or by checking and optimizing the PLC program.

2.2 Data Types

When adding variables in the PLC program you must enter the data type of the variable. The CNC supports the following types, described in the table below.

Type	Description	Possible Values
BOOL	Boolean (bit)	FALSE or TRUE
SINT	8-bit integer with sign	-128 a +127
USINT	Unsigned 8-bit integer	0 a 255
INT	16-bit integer with sign	-32768 a + 32767
UINT	Unsigned 16-bit integer	0 a +65535
DINT	32-bit integer with sign	-2147483648 a +2147483647
UDINT	Unsigned 32-bit integer	0 a +4294967295
LINT	64-bit integer with sign	-(2^63) a +(2^63) - 1
ULINT	Unsigned 64-bit integer	0 a +(2^64) - 1
REAL	32-bit floating-point real	-3.4E38 to 3.4E38 and -3.4E-38 to 3.4E-38 (6 to 7 digits precision)
ARRAY	Vector of INT or REAL values	According to the type used

To add, edit or delete variables refer to the AKC-PPC IDE User Manual and check the PLC editor section.

2.3 Operators and Expressions

Operators are commands that we apply to expressions that we write in our programs to perform a certain action. In the following we will describe the operators and expressions and how they should be used when programming the PLC for the AKC-PPC CNC.

2.3.1 Arithmetic Operators

Arithmetic operators perform mathematical operations and return a value as the result. If we need to write larger or more complex expressions, we can combine these operators and create them. This allows us to perform any kind of calculation that is needed.
In the following table we see all the arithmetic operators available for building the PLC program.

Operator	Description
+	Add
-	Subtraction
*****	Multiplication
/	Division
******	Potentiation (Used only in REAL numbers)
SQRT()	Square Root (Used only in REAL numbers)
ABS()	Absolute Value (Used only for REAL numbers)
SIN()	Sine (Used only for REAL numbers)
COS()	Cosine (Used only in numbers of type REAL)
TAN()	Tangent (Used only with REAL numbers)
ATAN()	Arc Tangent (Used only with REAL numbers)

2.3.2 Relational Operators

All relational operands parse 2 operands and return the BOOL type, regardless of the type of comparison being performed.
Relational operators always evaluate two operands of the same type.
In the following table, we see all the relational operators available for building the PLC program.

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
=	Same as
<>	Different

2.3.3 Logical Operators

Logical operators allow us to create larger logical expressions by joining two or more expressions. All logical operands return the same type in an expression. For this reason, in expressions with these operators you should not use variables of different types.
In the following table we see all the logical operators available for building the PLC program.

Operator	Description
AND	And Boolean
OR	OR Boolean
XOR	OR Exclusive
NOT	No

2.3.4 Assignment Operator

The assignment operator is used to set the value of a variable.
In the following table, we see the assignment operator used to construct the PLC program.

Operator	Description
:=	Assignment

In its use, the operand on the left represents the variable to which we want to assign the value entered in the expression (or variable) on the right.

2.3.5 Conditional control structures and repetitions

The structures of conditional controls and repetitions allow us to make important decisions in the execution of the PLC program according to a given expression.

2.3.5.1 IF

IF is a conditional control structure that performs the given actions if the conditional expression is satisfied. You can also chain with the ELSIF command and the ELSE

command. Syntax of the IF control structure:

IF only

```
IF (Expression 1) THEN
    Actions to be performed if "Expression 1" is satisfied
END_IF;
```

IF in conjunction with ELSE

```
IF (Expression 1) THEN
    Actions to be performed if "Expression 1" is satisfied
ELSE
    Actions to be performed if "Expression 1" is not satisfied
END_IF;
```

IF in conjunction with ELSIF and ELSE

```
IF (Expression 1) THEN
    Actions to be performed if "Expression 1" is satisfied
ELSIF (Expression 2) THEN
    Actions to be performed if "Expression 1" is not satisfied, but "Expression 2" is
is satisfied ELSE
    Actions to be performed if "Expression 1" and "Expression 2" are not satisfied
END_IF;
```

Example of use:

```
IF var > 10 THEN
    value := 1000;
    onGreenLight := TRUE;
    onYellowLight := FALSE;
    onRedLight := FALSE;
ELSIF (var > 0) AND (var <= 10) THEN
    value := 100;
    onGreenLight := FALSE;
    onYellowLight := TRUE;
    onRedLight := FALSE;
ELSE
    value := 1;
    onGreenLight := FALSE;
    yellowLight on := FALSE; redLight on
    := TRUE;
END_IF;
```

2.3.5.2 CASE

CASE is a conditional control structure that performs only one block of actions.
CASE validation is done by comparing the value of the control variable with the value of the registered indexes. After the comparison the index that has the same value as the control variable will be executed. Below is the format of how this control structure is used.
You cannot use a control variable of type REAL in CASE, only variables of type INT.

CASE control structure syntax:
```
CASE
control variable OF 0:
    Actions to be performed if the "control variable" is equal to 0
1:
    Actions to be performed if the "control variable" is equal to 1
2:
    Actions to be performed if the "control variable" is equal to 2
.
.
.
N:
    Actions to be performed if the "control variable" is equal to N
END_CASE;
```

Usage example:
```
CASE
varControl OF 0:
    value := var * varControl;
    onGreenLight := TRUE;
    onYellowLight := FALSE;
    onRedLight := FALSE;
1:
    value := var * varControl;
    onGreenLight := FALSE;
    onYellowLight := TRUE;
    onRedLight := FALSE;
2:
    value := var * varControl;
    onGreenLight := FALSE;
    onYellowLight := FALSE;
    onRedLight := TRUE;
END_CASE;
```

2.3.5.3 WHILE

WHILE is a repetition structure that executes a block of actions as long as the validation condition is true. The structure always tests the validation before performing the actions.

Syntax of the WHILE control structure:

```
WHILE (Expression 1) DO
    Actions to be performed while "Expression 1" is satisfied
END_WHILE;
```

Example of use:

```
WHILE (varTemp > varTarget) DO
    varTemp := varTemp / 2;
    varCount := varCount + 1;
END_WHILE;
```

One must be very careful when using this structure, since depending on the amount of repetitions the PLC.

2.3.5.4 FOR

FOR is a repetition structure that executes a block of actions as long as the validation condition is true. The structure always tests the validation before performing the actions.
The validation condition in this structure is controlled by the structure itself
It uses a variable to control the count and increment.

Syntax of the FOR control structure:

```
FOR Index := Initial TO Final BY Increment DO
  Actions to be performed while the "Index" value is less than the "End" value END_FOR;
```

Always when entering this structure the "Index" will assume the "Initial" value and at each repetition the "Index" will be
incremented according to the "Increment" used. At the beginning of each repetition a check is made if the value of the "Index" has reached the "Final" value, if this has occurred the repeats of this structure are finished.

Example of use:

```
FOR Index := 0 TO 9 BY 1 DO
     varCount := varCount + 1;
END_FOR;
```

2.3.6 Expressions

Expressions are defined as a set of functions and variables that produce a result.
The variable that will receive the value of the expression must be the same type as the variables and functions used in the expression. Therefore, you cannot add two variables of type INT and store them in a variable of type REAL or vice versa. If this type of action is necessary, the variables will require conversion to the type being used.

Example of use:

```
varValue := varRep + 1;
varValue := COS( 30 ) * varLength;
```

2.3.6.1 Logical expressions

Boolean expressions work by comparing operators and generating a Boolean result (TRUE or FALSE). It is important to remember that the assignments and returns of the boolean type are TRUE and FALSE, it is not allowed to assign "1" to a boolean variable.
The use of parentheses is very important to organize and allow a better understanding of the expression. Comparisons can only be made with variables of the same type, although in an expression this can contain several comparisons with different types.

Example of use:

```
varValue := (varA > varB) AND ((varC <= varD) OR (varE == varF));
```

The result of this expression is TRUE only if all conditions are true.
Note that "varA" and "varB" must be of the same type. The same is true for "varC" and "varD".
But "varA" and "varC" don't have to be of the same type.

3.0 System Variables

To facilitate PLC programming, the CNC has several variables defined.
We call these variables "System Variables", since the user is not allowed to edit them. All these variables start with the prefix "sv".
Just by typing "sv" the PLC editor's intelisense will show all existing "System Variables".

3.1 General

3.1.1 svMemFirstCycle

Type:
BOOL
Description:
Reading Only.
Memory first PLC cycle (at power-up).

3.1.2 svOnInitialization

Type:
BOOL
Description:
Reading Only.
Indicates that a rising edge (from 0 to 1) has occurred at CNC initialization

3.1.3 svBlockInitialization

Type:
BOOL
Description:
Reading / Writing.
Locks the CNC startup.
If the CNC receives the initialization code (1000) and this variable is in the "TRUE" state, the initialization will not be made.

3.1.4 svSoftkeyBlockInitialization

Type:
BOOL
Description:
Reading Only.
Softkey lock bit 1, associated with the startup softkey.

3.1.5 svCncInitialized

Type:
BOOL
Description:
Reading Only.
Indicates that the CNC has already booted, exiting startup mode.

3.1.6 svEmergencyCnc

Type:
BOOL
Description:
Reading Only.
Indicates emergency status for the CNC.

3.1.7 svEmergencyOutput

Type:
BOOL
Description:
Reading Only.
State of the emergency exit.

3.1.8 svEmergencyState

Type:
BOOL
Description:
Reading Only.
Internal emergency status of the CNC.

3.1.9 svInitializationMode

Type:
BOOL
Description:
Read-only. Boot
mode.

3.1.10 svNewPLC

Type:
BOOL
Description:
Reading Only.
Indicates that a new PLC has been sent to the CNC.

3.1.11 svParametersUpdated

Type:
BOOL
Description:
Reading Only.
Variable that indicates whether there has been a parameter update, new PLC, or that the first PLC cycle has been executed.

3.1.12 svUserLevel

Type:

BOOL

Description:

Reading Only.

Tells you the current user level.

3.1.13 svVoid

Type:

BOOL

Description:

Reading Only.

Auxiliary variable for the return of all functions that do not need a return.

3.1.14 svTransition500ms

Type:

BOOL

Description:

Reading Only.

It stays in TRUE for 1 PLC cycle every 500ms.

3.1.15 svBlink500ms

Type:

BOOL

Description:

Reading Only.

It flashes 500ms in a "TRUE" state and 500ms in a "FALSE" state.

3.1.16 svIntDraft1

Type:

BOOL

Description:

Reading Only.

Used as a draft for any logic.

3.1.17 svIntDraft2

Type:

BOOL

Description:

Reading Only.

Used as a draft for any logic.

3.1.18 svRealDraft1

Type:

BOOL

Description:

Reading Only.

Used as a draft for any logic.

3.1.19 svRealDraft2

Type:

BOOL

Description:

Reading Only.

Used as a draft for any logic.

3.1.20 svSaveMemory

Type:

BOOL

Description:

Reading Only.

Requests to save values in the retention-protected area

3.1.21 svAxisInHoming

Type:

BOOL

Description:

Variable that indicates whether the axis is referencing.

3.1.22 svAxisLagError

Type:

BOOL

Description:

Indicates the lag error of the axis.

3.1.23 svClearLagError

Type:

BOOL

Description:

Reading / Writing.

Clears tracking error at the CNC.

3.1.24 svFeedholdCnc

Type:

BOOL

Description:

Reading Only.

Indicates general feedhold for CNC.

3.1.25 svReferenceMode

Type:
BOOL
Description:
Read Only. Reference
Mode.

3.1.26 svSelectOrigin

Type:
BOOL
Description:
Reading Only.
Source selection for the preset.

3.2 ISO Program Execution Commands

3.2.1 svStartCnc

Type:

BOOL

Description:

Reading Only.

Start" command for the CNC.

3.2.2 svStopCnc

Type:

BOOL

Description:

Reading Only.

Stop" command for the CNC.

3.2.3 svOnKeyStart

Type:

BOOL

Description:

Reading Only.

Indicates that a rising edge (from 0 to 1) has occurred on the CNC start key.

3.2.4 svOnKeyStop

Type:

BOOL

Description:

Reading Only.

Indicates that a rising edge (from 0 to 1) has occurred on the CNC stop key.

3.2.5 svBlockStart

Type:

BOOL

Description:

Reading / Writing.

Blocks the start of execution of an ISO program.

If the CNC receives the ISO program start code (400) or the system variable "svStartCnc" goes to the "TRUE" state and this variable is in the "TRUE" state the execution of the selected ISO program will not start.

3.2.6 svAbortCnc

Type:

BOOL - Reading/Writing

Description:

If you assign the value "TRUE" to this variable, you abort the execution of the ISO program on the CNC. Under normal conditions keep the value of this variable at "FALSE".

3.3 ISO Program Execution

3.3.1 svCncInElementSimulation

Type:

BOOL

Description:

Reading Only.

Indicates that the CNC is operating in element simulation mode.

3.3.2 svCncInSimulatedMode

Type:

BOOL

Description:

Reading Only.

Indicates that the CNC is operating in simulation mode.

3.3.3 svExecutionError

Type:

BOOL

Description:

Reading Only.

Program execution error.

3.3.4 svExecutionHold

Type:

BOOL

Description:

Reading Only.

Execution aborted or awaiting authorization to execute.

3.3.5 svExecutionInEmergency

Type:

BOOL

Description:

Reading Only.

Execution in a state of emergency.

3.3.6 svExecutionInInitialState

Type:

BOOL

Description:

Reading Only.

Execution in initial state.

3.3.7 svExecutionInWaitState

Type:

BOOL

Description:

Read Only. Run
Waiting.

3.3.8 svExecutionPause

Type:

BOOL

Description:

Reading Only.
Indicates that program execution has been paused.

3.3.9 svExecutionRun

Type:

BOOL

Description:

Reading Only.
Indicates that the CNC is running a program.

3.3.10 svNewCodeM

Type:

BOOL

Description:

Reading Only.
New M function for PLC.

3.3.11 svNewCodeS

Type:

BOOL

Description:

Reading Only.
New S function for PLC.

3.3.12 svNewCodeT

Type:

BOOL

Description:

Reading Only.
New T function for PLC.

3.3.13 svAnyFunctionMST

Type:

BOOL - Read Only

Description:

Constant for M, S and T functions to use when you want to know if an M, S or T function was executed regardless of the code.

3.4 ISO Program Selection

3.4.1 svSelectDirNumber

Type:

BOOL

Description:

Reading Only.

Directory number of the program that will be selected by the PLC

0 - Directory DATAPROGRAMS

1 - Directory DATAPROGRAMS\SUB.DIR

1 - Directory DATAPROGRAMS\CYC.DIR

3.4.2 svSelectProgramNumber

Type:

BOOL

Description:

Reading Only.

Program number to be selected by the PLC

3.4.3 svKeySelectProgram

Type:

BOOL

Description:

Reading Only.

Select program code. The program and directory numbers must be programmed in Select_Program_Number and Select_Dir_Number.

3.4.4 svKeyPgmCall

Type:

BOOL

Description:

Reading Only.

Calls the execution of program #1, CYC.DIR.

3.5 Potentiometers

3.5.1 svPotOverridePlcControl

Type:
BOOL
Description:
Reading / Writing.
If the state of this variable is "TRUE" it determines that the potentiometer will be controlled by the PLC.

3.5.2 svPotOverride

Type:
BOOL
Description:
Reading / Writing.
Percent of advance potentiometer, which will be used for speed control of all linear and rotary axis movements.

3.5.3 svPotSpindle

Type:
BOOL
Description:
Reading / Writing.
Percent of spindle potentiometer, which will be used for spindle speed control.

3.6 Errors

3.6.1 svCncError

Type:
BOOL
Description:
Reading Only.
Indicates the number of the error that is occurring at the CNC.
If the value of this variable equals "0", the CNC has no errors.

3.6.2 svStoFault

Type:
BOOL
Description:
Reading Only.
Indicates that svCncError is between 50001 and 50003.

3.6.3 svExternalClear

Type:
BOOL
Description:
Reading Only.
Used to clear by external button.

3.7 Key Codes

3.7.1 svNewCncKey

Type:

BOOL

Description:

Reading Only.

New key pressed by the user.

3.7.2 svRequestKeyPlc

Type:

BOOL

Description:

Reading Only.

Indicates which key is to be passed to the CNC.

3.7.3 svSendKeyToCnc

Type:

BOOL

Description:

Reading Only.

Command to send a key-code signal to the CNC.

3.7.4 svLastCncKey

Type:

BOOL

Description:

Reading Only.

Last key pressed or received from the CNC.

3.7.5 svOnKeyCE

Type:

BOOL

Description:

Reading Only.

Indicates that a rising edge (from 0 to 1) has occurred on the CE key.

3.8 Screens

3.8.1 svCurrentScreen

Type:
BOOL
Description:
Reading Only.
Code of the current screen in focus.

3.8.2 svLastScreen

Type:
BOOL
Description:
Reading Only.
Previous focus to the current one in focus.

3.9 Softkey Trees

3.9.1 svCurrentHorizontalState

Type:
BOOL
Description:
Reading Only.
Current state of the horizontal softkeys.

3.9.2 svCurrentVerticalState

Type:
BOOL
Description:
Reading Only.
Current status of vertical softkeys.

3.9.3 svCncNewSoftkeyState

Type:
BOOL
Description:
Reading Only.
Indicates that a new softkey state has been called by the user.

3.9.4 svCncNewHorizontalState

Type:
BOOL
Description:
Reading Only.
New state of horizontal softkeys called by user.

3.9.5 svPlcNewHorizontalState

Type:
BOOL
Description:
Reading Only.
Indicates which horizontal softkey state you want to call up.

3.9.6 svCncNewVerticalState

Type:
BOOL
Description:
Reading Only.
New vertical softkey state called by the user.

3.9.7 svPlcNewVerticalState

Type:

BOOL

Description:

Reading Only.

Indicates which vertical softkey state you want to call up.

3.9.8 svRequestNewSoftkeyState

Type:

BOOL

Description:

Reading Only.

Requests that the CNC sets a new softkey state.

4.0 Native Functions

Native functions have this name because they are created directly in the PLC/CNC firmware and cannot be edited or customized.
Native functions have no prefix for identification, but you can identify them in another way. These functions are described with capital letters.
Native functions can have more than one input data, but they have the characteristic of only one output data (only for read functions).
The native write functions or functions that are used for some kind of display do not return any value.

4.1 Messages

The functions described here will display messages at the top of the AKC-PPC CNC.
These messages are displayed according to their type.
The messages shown here must be properly registered in the AKC-PPC IDE's Alarm Editor. For this please refer to the AKC-PPC IDE user manual.

4.1.1 Alarm - ALARM

4.1.1.1 Operation

This function will continuously display an "Alarm" message on the screen when the PLC memory that is linked to the "BOOL_STATE" input receives an up transition, i.e., changes its value from the "FALSE" state to the "TRUE" state.
When the PLC memory that is linked to the "BOOL_STATE" input changes to the "FALSE" state the message will continue to be displayed.
This type of message is only removed from the screen when the code for clearing errors is sent to the CNC (code 333).
When this function is called the background color of the message displayed is red.
Up to 64 alarms can be used in the PLC application - that is, the indices range from 0 to 63.

4.1.1.2 Entries

INT_NUMBER:
Alarm index - set in the alarm editor.
BOOL_STATE:
Enables the message display when in the TRUE state.

5.1.1.3 Function Call

ALARM(INT_NUMBER, BOOL_STATE);

4.1.2 Blinking - BLINK_MESSAGE

4.1.2.1 Operation

This function will display a "Blinking" message on the screen at 2s intervals when the PLC memory that is linked to the "BOOL_STATE" input is in the "TRUE" state.
When the PLC memory that is linked to the "BOOL_STATE" input changes to the "FALSE" state the message will no longer be displayed.
When this function is called the background color of the displayed message will be yellow.
Up to 32 messages can be used in the PLC application - that is, the indexes range from 0 to 31.

4.1.2.2 Entries

INT_NUMBER:
Alarm index - set in the alarm editor.
BOOL_STATE:
Enables the message display when in the TRUE state.

4.1.2.3 Function Call

BLINK_MESSAGE(INT_NUMBER, BOOL_STATE);

4.1.3 Normal - MESSAGE

4.1.3.1 Operation

This function will continuously display a "Blinking" message on the screen when the PLC memory that is linked to the "BOOL_STATE" entry is in the "TRUE" state.
When the PLC memory that is linked to the "BOOL_STATE" input changes to the "FALSE" state the message will no longer be displayed.
When this function is called the background color of the displayed message will be yellow.

Up to 128 messages can be used in the PLC application - that is, the indexes range from 0 to 127.

4.1.3.2 Entries

INT_NUMBER:

Alarm index - set in the alarm editor.

BOOL_STATE:

Enables the message display when in the TRUE state.

4.1.3.3 Function Call

MESSAGE(INT_NUMBER, BOOL_STATE);

4.1.4 Timed - TIMED_MESSAGE

5.1.4.1 Operation

Displays a yellow message at the top of the CNC screen. This message is visible for 4 seconds after the function receives a rising edge (from FALSE to TRUE) on the "BOOL_STATE" input. Up to 32 different messages can be used in the PLC application - that is, the indices range from 0 to 31.

4.1.3.1 Operation

This function will display on the screen for 4s a "Timed" message when the PLC memory that is linked to the "BOOL_STATE" input receives an up transition, i.e. changes its value from the "FALSE" state to the "TRUE" state. After the display time has elapsed (4s) the message will no longer be displayed until a climb transition occurs again in the PLC memory that is linked to the "BOOL_STATE" entry.

When this function is called the background color of the displayed message will be yellow.

Up to 32 messages can be used in the PLC application - that is, the indexes range from 0 to 31.

4.1.3.2 Entries

INT_NUMBER:

Alarm index - set in the alarm editor.

BOOL_STATE:

Enables the message display when in the TRUE state.

4.1.3.3 Function Call

TIMED_MESSAGE(INT_NUMBER, BOOL_STATE);

4.2 Transition

4.2.1 Climb Transition - R_EDGE

4.2.1.1 Operation

This function detects an ascent transition, that is, the transition from the FALSE state to the TRUE state of a Boolean variable or expression.
Returns TRUE for one PLC cycle.

4.2.1.2 Entries

BOOL:
Boolean variable or expression that you want to detect the up transition.

4.2.1.3 Outputs

This function returns a boolean that is TRUE for one PLC cycle, indicating that an up transition has occurred in the input boolean.

4.2.1.4 Function Call

R_EDGE(BOOL);

4.2.2 Downhill Transition - F_EDGE

4.2.2.1 Operation

This function detects a drop transition, that is, the transition from the TRUE state to the FALSE state of a Boolean variable or expression.
Returns TRUE for one PLC cycle.

4.2.2.2 Entries

BOOL:
Boolean variable or expression that you want to detect the drop transition.

4.2.2.3 Outputs

This function returns a boolean that is TRUE for one PLC cycle, indicating that a downward transition has occurred in the input boolean.

4.2.2.4 Function Call

F_EDGE(BOOL);

4.2.3 Any Transition - RF_EDGE

4.2.3.1 Operation

This function detects any state transition (up and down) of a variable or Boolean expression. It returns TRUE for one PLC cycle.

4.2.3.2 Entries

BOOL:
Boolean variable or expression that you want to detect the transition.

4.2.3.3 Outputs

This function returns a boolean that is TRUE for one PLC cycle, indicating that a transition has occurred in the input boolean.

4.2.3.4 Function Call

RF_EDGE(BOOL);

4.3 Parameters, H Variables, and Reserved Variables

4.3.1 Read Parameter - R_PARAM

4.3.1.1 Operation

Returns the value of the CNC parameter specified in the function input.

4.3.1.2 Entries

INT:

Integer for the index of the parameter that will be read. Accepts values from 0 to 4199. General parameters: 0-499

Axis parameters: 1000-4199 (parIndex= 1000 + ((axis-1)*100)+parNum) PLC parameters: 900-999

Indexes from 500 to 899 are not used.

4.3.1.3 Outputs

Returns the value of the parameter indicated in the "INT" argument. The returned value is of type REAL.

4.3.1.4 Function Call

R_PARAM(INT);

4.3.2 Write in Parameter - W_PARAM

4.3.2.1 Operation

Changes the value of the parameter indicated in the function input.

4.3.2.2 Entries

INT:

Integer for the index of the parameter that will be read. Accepts values from 0 to 4199. General parameters: 0-499

Axis parameters: 1000-4199 (parIndex= 1000 + ((axis-1)*100)+parNum) PLC parameters: 900-999

Indexes from 500 to 899 are not used.

REAL:

New value of type REAL to be assigned to the parameter indicated in the "INT" argument.

4.3.2.3 Outputs

This function has no return.

4.3.2.4 Function Call

W_PARAM(INT, REAL);

4.3.3 Read H Register - R_HVAR

4.3.3.1 Operation

Returns the value of the H register indicated in the function input.

4.3.3.2 Entries

INT:

Number of the H register you want to read. Accepts values from 0 to 999.

4.3.3.3 Outputs

Returns the value of the H register indicated in the "INT" argument. The returned value is of type REAL.

4.3.3.4 Function Call

R_HVAR(INT);

4.3.4 Write to H Register - W_HVAR

4.3.4.1 Operation

Changes the value of the H register indicated in the function input.

4.3.4.2 Entries

INT:

Number of the H register you want to change the value of. Accepts values from 0 to 999.

REAL:

New value to be assigned to the H register indicated in the "INT" argument.

4.3.4.3 Outputs

This function has no return.

4.3.4.4 Function Call

W_HVAR(INT, REAL);

4.3.5 Read Reserved Variable - R_COMMAND_VAR

4.3.5.1 Operation

Returns the value of the reserved variable specified in the function input.
The list of existing reserved variables can be seen in the "Reserved Variables" manual.

4.3.5.2 Entries

INT:

Number of the reserved variable that you want to read. It accepts values from 10000 to 19999, which is the number of existing reserved variables. Not all reserved variable numbers perform some action.

4.3.5.3 Outputs

Returns the value of the reserved variable indicated in the "INT" argument. The returned value is of type REAL.

4.3.5.4 Function Call

R_COMMAND_VAR(INT);

4.3.6 Write to Reserved Variable - W_COMMAND_VAR

4.3.6.1 Operation

Changes the value of the reserved variable indicated in the function input.
The list of existing reserved variables can be seen in the "Reserved Variables" manual.

4.3.6.2 Entries

INT:

Number of the reserved variable that you want to change the value of, values from 10000 to 19999 can be programmed, which are the numbers of existing reserved variables.

REAL:

New value to be assigned to the reserved variable indicated in the "INT" argument.

4.3.6.3 Outputs

This function has no return.

4.3.6.4 Function Call

W_COMMAND_VAR(INT, REAL);

4.4 Trigonometric/Mathematical

4.4.1 Sine - SIN

4.4.1.1 Operation

Returns the sine value of the entered angle.

4.4.1.2 Entries

***REAL*:**

The actual value (in degrees) of the angle you want to calculate the sine.

4.4.1.3 Outputs

Returns the sine value of the angle that was entered into the "REAL" input.

4.4.1.4 Function Call

SIN(REAL)

4.4.2 Cosine - COS

4.4.2.1 Operation

Returns the value of the cosine of the entered angle.

4.4.2.2 Entries

***REAL*:**

The actual value (in degrees) of the angle you want to calculate the cosine of.

4.4.2.3 Outputs

Returns the value of the cosine of the angle that was entered into the "REAL" input.

4.4.2.4 Function Call

COS(REAL)

4.4.4 Tangent - TAN

4.4.4.1 Operation

Returns the value of the tangent of the entered angle.

4.4.4.2 Entries

***REAL*:**

The actual value (in degrees) of the angle you want to calculate the tangent.

4.4.4.3 Outputs

Returns the value of the tangent of the angle that was entered in the "REAL" input.

4.4.4.4 Function Call

TAN(REAL)

4.4.5 Tangent Arc - ATAN

4.4.5.1 Operation

Returns the arc tangent value of the entered value.

4.4.5.2 Entries

***REAL*:**

Actual value that you want to calculate the arc tangent.

4.4.5.3 Outputs

Returns the arc tangent value of the number that was entered in the "REAL" input.

4.4.5.4 Function Call

ATAN(REAL)

4.4.6 Value of the Pi - PI Constant

4.4.6.1 Operation

Returns the value of the mathematical constant PI.

4.4.6.2 Entries

In this function there is no entry.

4.4.6.3 Outputs

Returns the value of the mathematical constant PI.

4.4.6.4 Function Call

PI()

4.4.7 Integer Value - TRUNC

4.4.7.1 Operation

Returns the integer value of a number. Example: TRUNC(35.33) will return 35.

4.4.7.2 Entries

***REAL*:**

Actual value from which you want to remove the decimal places.

4.4.7.3 Outputs

Returns the integer value of the number that was entered in the "REAL" input.

4.4.7.4 Function Call

TRUNC(24.4456)

4.4.8 Fractional Value - FRAC

4.4.8.1 Operation

Returns the value of the fractional part of a real number. Example: FRAC(35.33) will return 0.33.

4.4.8.2 Entries

***REAL*:**

Actual value from which you want to get the fractional part.

4.4.8.3 Outputs

Returns a REAL number with only the fractional part of the number that was entered in the "REAL" input.

4.4.8.4 Function Call

FRAC(45,354)

4.4.9 Square Root - SQRT

4.4.9.1 Operation

Returns the square root of the entered number.

4.4.9.2 Entries

***REAL*:**

REAL value from which you want to get the square root.

4.4.9.3 Outputs

Returns the value of the square root of the number that was entered in the "REAL" input.

4.4.9.4 Function Call

SQRT(9)

4.4.10 Absolute Value - ABS

4.4.10.1 Operation

Returns the modulus of a real number.

4.4.10.2 Entries

REAL:

REAL value from which you want to get the modulus.

4.4.10.3 Outputs

Returns the modulus value of the number that was entered in the "REAL" input.

4.4.10.4 Function Call

ABS(-3)

4.5 Data type conversions

4.5.1 Boolean to Real - BOOL_TO_REAL

4.5.1.1 Operation

Converts a variable from type BOOL to type REAL.

4.5.1.2 BOOL

entries:

Value of type BOOL that will be converted to type REAL.

4.5.1.3 Outputs

Returns the boolean value converted to REAL.

4.5.1.4 Function Call

BOOL_TO_REAL(BOOL)

4.5.6 Boolean to Integer - BOOL_TO_INT

4.5.6.1 Operation

Converts a variable from type BOOL to type INT.

4.5.6.2 BOOL

entries:

Value of type BOOL that will be converted to type INT.

4.5.6.3 Outputs

Returns the boolean value converted to INT.

4.5.6.4 Function Call

BOOL_TO_INT(BOOL);

4.5.2 Real to Boolean - REAL_TO_BOOL

4.5.2.1 Operation

Converts a variable from type REAL to type BOOL.

4.5.2.2 REAL

entries:

Value of type REAL that will be converted to type BOOL.

4.5.2.3 Outputs

Returns the actual value converted to BOOL.

4.5.2.4 Function Call

REAL_TO_BOOL(REAL);

4.5.3 Real to Integer - REAL_TO_INT

4.5.3.1 Operation

Converts a variable from type REAL to type INT.

4.5.3.2 REAL

entries:

Value of type REAL that will be converted to type INT.

4.5.3.3 Outputs

Returns the actual value converted to INT.

4.5.3.4 Function Calls

REAL_TO_INT(REAL);

4.5.4 Real to Unsigned integer - REAL_TO_UINT

4.5.4.1 Operation

Converts a variable from type REAL to type UINT.

4.5.4.2 REAL

entries:

Value of type REAL which will be converted to type UINT.

4.5.4.3 Outputs

Returns the actual value converted to UINT.

4.5.4.4 Function Calls

REAL_TO_UINT (REAL);

4.5.5 Unsigned Integer to Integer - UINT_TO_INT

4.5.5.1 Operation

Converts a variable from type UINT to type INT.

4.5.5.2 UINT

entries:

Value of type UINT that will be converted to type INT.

4.5.5.3 Outputs

Returns the UINT value converted to INT.

4.5.5.4 Function Call

UINT_TO_INT(UINT);

4.5.6 Integer to Unsigned Integer - INT_TO_UINT

4.5.6.1 Operation

Converts a variable from type INT to type UINT.

4.5.6.2 INT:

Value of type INT that will be converted to type UINT.

4.5.6.3 Outputs

Returns the INT value converted to UINT.

4.5.6.4 Function Call

INT_TO_UINT(INT);

4.5.7 Integer to Boolean - INT_TO_BOOL

4.5.7.1 Operation

Converts a variable from type INT to type BOOL.

4.5.7.2 INT:

Value of type INT that will be converted to type BOOL.

4.5.7.3 Outputs

Returns the INT value converted to BOOL.

4.5.7.4 Function Call

INT_TO_BOOL(INT);

4.5.8 Integer to Real - INT_TO_REAL

4.5.8.1 Operation

Converts a variable from type INT to type REAL.

4.5.8.2 INT:

Value of type INT that will be converted to type REAL.

4.5.8.3 Outputs

Returns the INT value converted to REAL.

4.5.8.4 Function Call

INT_TO_REAL (INT);

5.0 System Functions

To facilitate PLC programming, the CNC has several functions defined.
The "System Functions" functions have this name because the user is not allowed to edit them. All "System Functions" start with the prefix "sf".
As soon as you type "sf" intelisense will show you all the existing "System Functions".
System functions can have several input data, but are characterized by only one output data.

5.1 Messages

The functions described here will display messages at the top of the AKC-PPC CNC.
These messages are displayed according to their type.
The messages shown here must be properly registered in the AKC-PPC IDE's Alarm Editor. For this please refer to the AKC-PPC IDE user manual.

5.1.1 Alarm - sfShowAlarmMessage

5.1.1.1 Operation

This function will continuously display an "Alarm" message on the screen when the PLC memory that is linked to the "SHOW" input receives an up transition, i.e., changes its value from the "FALSE" state to the "TRUE" state.
When the PLC memory that is linked to the "SHOW" input changes to the "FALSE" state, the message will continue to be displayed.
This type of message is only removed from the screen when the code for clearing errors is sent to the CNC (code 333).
When this function is called the background color of the message displayed is red.
Up to 64 alarms can be used in the PLC application - that is, the indices range from 0 to 63.

5.1.1.2 Entries

***INDEX* (INT):**
Alarm index - defined in the AKC-PPC IDE's message editor.
***SHOW* (BOOL):**
Enables the message display when in the TRUE state.

5.1.1.2 Outputs

Returns a boolean, indicating whether the message is being displayed by the function or not (TRUE or FALSE).

5.1.1.3 Function Call

sfShowAlarmMessage(INDEX, SHOW);

5.1.2 Blinking - sfShowBlinkingMessage

5.1.2.1 Operation

This function will display a "Blinking" message on the screen at 2s intervals when the PLC memory that is linked to the "*SHOW"* input is in the "TRUE" state.
When the PLC memory that is linked to the "*SHOW"* input changes to the "FALSE" state, the message will not will be displayed anymore.
When this function is called the background color of the displayed message will be yellow.
Up to 32 messages can be used in the PLC application - that is, the indexes range from 0 to 31.

5.1.2.2 Entries

***INDEX* (INT):**
Index of the blinking message - defined in the AKC-PPC IDE's message editor.
***SHOW* (BOOL):**
Enables the message display when in the TRUE state.

5.1.2.3 Function Call

sfShowBlinkingMessage(INDEX, SHOW);

5.1.3 Normal - sfShowNormalMessage

5.1.3.1 Operation

This function will continuously display a "Blinking" message on the screen when the PLC memory that is linked to the "BOOL_STATE" entry is in the "TRUE" state.
When the PLC memory that is linked to the "BOOL_STATE" input changes to the "FALSE" state the message will no longer be displayed.
When this function is called the background color of the displayed message will be yellow.
Up to 128 messages can be used in the PLC application - that is, the indexes range from 0 to 127.

5.1.3.2 Entries

INDEX* (INT)*:
Index of the normal message - defined in the AKC-PPC IDE's message editor.
SHOW* (BOOL)*:
Enables the message display when in the TRUE state.

5.1.3.3 Function Call

sfShowNormalMessage(INDEX, SHOW);

5.1.4 Timed - sfShowTimedMessage

5.1.4.1 Operation

Displays a yellow message at the top of the CNC screen. This message is visible for 4 seconds after the function receives a rising edge (from FALSE to TRUE) on the "BOOL_STATE" input. Up to 32 different messages can be used in the PLC application - that is, the indices range from 0 to 31.

5.1.3.1 Operation

This function will display on the screen for 4s a "Timed" message when the PLC memory that is linked to the "BOOL_STATE" input receives an up transition, i.e. changes its value from the "FALSE" state to the "TRUE" state. After the display time has elapsed (4s) the message will no longer be displayed until a climb transition occurs again in the PLC memory that is linked to the "BOOL_STATE" entry.
When this function is called the background color of the displayed message will be yellow.
Up to 32 messages can be used in the PLC application - that is, the indexes range from 0 to
31.

5.1.3.2 Entries

INDEX* (INT)*:
Index of the timed message - defined in the AKC-PPC IDE's message editor.
SHOW* (BOOL)*:
Enables the message display when in the TRUE state.

5.1.3.3 Function Call

sfShowTimedMessage(INDEX, SHOW);

5.2 Codes and Navigation

5.2.1 Identify CNC Received Code - sfKeyCode

5.2.1.1 Operation

When the CNC receives a code this function compares whether the code received by the CNC is the same as the code entered in the "Code" input.
Codes can be sent by HMI screen components, by a physical keyboard, by softkeys, or by scripts.

5.2.1.2 Entries

***Code* (INT):**
Code that will be compared with the code received by the CNC.

5.2.1.3 Outputs

This function returns a boolean.
If the code received by the CNC is equal to the code that was entered in the "Code" input, this function returns TRUE, otherwise it returns FALSE.
If the CNC receives no code this function returns FALSE.

5.2.1.4 Function Call

sfKeyCode(Code);

5.2.2 Grab Selected Horizontal Tree - sfCurrentHorizontalSFKState

5.2.2.1 Operation

Searches the index of the horizontal softkey tree that is selected.

5.2.2.2 Entries

No entries.

5.2.2.3 Outputs

Returns an integer with the index of the horizontal softkey tree that is selected.

5.2.2.4 Function Call

sfCurrentHorizontalSoftkeyState()

5.2.3 Take Selected Vertical Tree - sfCurrentVerticalSoftkeyState

5.2.3.1 Operation

Searches the index of the vertical softkey tree that is selected.

5.2.3.2 Entries

No entrance.

5.2.3.3 Outputs

Returns an integer with the index of the vertical softkey tree that is selected.

5.2.3.4 Function Call

sfCurrentVerticalSoftkeyState()

5.2.4 Check Selected Screen - sfCurrentScreen

5.2.4.1 Operation

Checks the index of the screen that is currently selected.
Returns "TRUE" when the screen number associated with the input variable "Screen_Number" is equal to the screen in focus.

5.2.4.2 Entries

***ScreenNumber* (INT):**
Number of the screen that will be checked.
The screen number can be checked in the "HMI Studio" editor in the AKC-PPC IDE's project tree.

5.2.4.3 Outputs

This function returns a boolean.
If the index of the selected screen is equal to the one entered in the "ScreenNumber" input, this function returns TRUE, otherwise it returns FALSE.

5.2.4.4 Function Call

sfCurrentScreen(ScreenNumber);

5.2.5 Softkey Lock - sfBlockSoftkey

This function is no longer used in versions above 3.0. It is
retained for compatibility reasons.
The softkeys have a property called LockBit and in this property we use a PLC variable to do any kind of locking needed.

5.2.6 Change Softkey State - sfChangeSoftkey

This
function is no longer used in versions above 3.0. It is
retained for compatibility reasons.
Softkeys have a property called MemoryBit and in this property we use a PLC variable to control what state the softkey is in.

5.3 Execution

One of the ways in which ISO programs interact with the PLC are the M, S, and T functions.

5.3.1 sfFunctionM

5.3.1.1 Operation

The purpose of this function is to monitor the M functions sent to the CNC by the ISO programs.
When an ISO program sends an M code to the CNC, this function compares whether the code received by the CNC is the same as the code entered in the "FunctionCode" input. If this happens the return of this function will be TRUE, which means that the M function we are monitoring was sent to the PLC, otherwise the return of the function is FALSE.
The "HoldFunctionM" input holds the execution of the ISO program. So if the "HoldFunctionM" input is set to TRUE when the M function we are monitoring is detected, the CNC execution will be blocked, not allowing the ISO program to proceed to execute the next instruction until the "HoldFunctionM" input goes to FALSE.

5.3.1.2 Entries

***FunctionCode* (INT):**
Code of the M-function you want to monitor.
***HoldFunctionM* (BOOL):**
If this entry is TRUE, CNC execution is stopped on the line of this function until HoldFunctionM falls to
FALSE.

5.3.1.3 Output

This function returns a boolean that indicates if the M function called in a CNC program is equal to the function whose code was passed in the "FunctionCode" entry.

5.3.1.4 Function Call

sfFunctionM(HoldFunctionM, FunctionCode);

5.3.2 sfFunctionS

5.3.2.1 Operation

The purpose of this function is to monitor the S-functions sent to the CNC by the ISO programs.
When an ISO program sends an S code to the CNC, this function compares whether the code received by the CNC is the same as the code entered at the "FunctionCode" input. If this happens the return of this function will be TRUE, which means that the S function we are monitoring was sent to the PLC, otherwise the return of the function will be FALSE.
The "HoldFunctionS" input holds the execution of the ISO program. So if the "HoldFunctionS" input is set to TRUE when the S function we are monitoring is detected, the CNC execution will be blocked, not allowing the ISO program to proceed to execute the next instruction until the "HoldFunctionS" input goes to FALSE.

5.3.2.2 Entries

***FunctionCode* (INT):**
Code of the S-function you want to monitor.
***HoldFunctionS* (BOOL):**
If this entry is TRUE, CNC execution is stopped on the line of this function until HoldFunctionS falls to FALSE.

5.3.2.3 Output

This function returns a boolean that indicates if the function S called in a CNC program is equal to the function whose code was passed in the "FunctionCode" entry.

5.3.2.4 Function Call

sfFunctionS(HoldFunctionS, FunctionCode);

5.3.3 sfFunctionT

5.3.3.1 Operation

The purpose of this function is to monitor the T functions sent to the CNC by the ISO programs.
When an ISO program sends a T-code to the CNC this function compares whether the code received by the CNC is the same as the code entered in the "FunctionCode" input. If this occurs the return of this function is TRUE, which means that the T function we are monitoring was sent to the PLC, otherwise the return of the function is FALSE.
The "HoldFunctionT" input holds the execution of the ISO program. So if the "HoldFunctionT" input is set to TRUE when the T function we are monitoring is detected, the CNC execution will be blocked, not allowing the ISO program to proceed to execute the next instruction until the "HoldFunctionT" input goes to FALSE.

5.3.3.2 Entries

***FunctionCode* (INT):**
Code of the T function you want to monitor. If you want to monitor whether a T function exists independently of the code, you must call the function with the constant svAnyFunctionMST.
***HoldFunctionT* (BOOL):**
If this input is TRUE, CNC execution is stopped on the line of this function until HoldFunctionT falls to FALSE.

5.3.3.3 Output

This function returns a boolean that indicates if the T function called in a CNC program is equal to the function whose code was passed in the "FunctionCode" entry.

5.3.3.4 Function Call

sfFunctionT(HoldFunctionT, FunctionCode);

5.4 Transition

5.4.1 Climb Transition - sfRisingEdge

5.4.1.1 Operation
This function detects an ascent transition, that is, the transition from the FALSE state to the TRUE state of a Boolean variable or expression.
Returns TRUE for one PLC cycle.

5.4.1.2 Entries
Variable **(BOOL):**
Boolean variable or expression that you want to detect the up transition.

5.4.1.3 Outputs
This function returns a boolean that is TRUE for one PLC cycle, indicating that an up transition has occurred in the input boolean.

5.4.1.4 Function Call
sfRisingEdge (Variable)

5.4.2 Downhill Transition - sfFallingEdge

5.4.2.1 Operation
This function detects a drop transition, that is, the transition from the TRUE state to the FALSE state of a Boolean variable or expression.
Returns TRUE for one PLC cycle.

5.4.2.2 Entries
Variable **(BOOL):**
Boolean variable or expression that you want to detect the drop transition.

5.4.2.3 Outputs
This function returns a boolean that is TRUE for one PLC cycle, indicating that a downward transition has occurred in the input boolean.

5.4.2.4 Function Call
sfFallingngEdge (Variable)

5.4.3 Any Transition - sfCheckingEdge

5.4.3.1 Operation
This function detects any state transition (up and down) of a variable or Boolean expression. It returns TRUE for one PLC cycle.

5.4.3.2 Entries
Variable **(BOOL):**
Boolean variable or expression that you want to detect the transition.

5.4.3.3 Outputs
This function returns a boolean that is TRUE for one PLC cycle, indicating that a transition has occurred in the input boolean.

5.4.3.4 Function Call
sfCheckingEdge (Variable)

5.5 Parameters, H Variables, and Reserved Variables

5.5.1 Read Parameter - sfReadParameter

5.5.1.1 Operation

Returns the value of the CNC parameter specified in the function input.

5.5.1.2 Entries

***Axis* (INT):**

Number of the axis from which you want to read the parameter.

***Parameter* (INT):**

Number of the parameter that you want
to read. If the type is general 0: 0-499
If the type is axis or PLC: 0-99

***Type* (INT):**

Type of the parameter that you want to read. It can have the following
values: Type = 0 - General
Type = 1 - Axis
Type = 2 - PLC

5.5.1.3 Outputs

Returns the value of the parameter specified in the input arguments. The returned value is of type REAL.

5.5.1.4 Function Call

sfReadParameter(Axis, Parameter, Type);

5.5.2 Write to Parameter - sfWriteParameter

5.5.2.1 Operation

Changes the value of the parameter indicated in the function input.

5.5.2.2 Entries

***Axis* (INT):**

Number of the axis from which you want to read the parameter.

***Parameter* (INT):**

Number of the parameter that you want
to read. If the type is general 0: 0-499
If the type is axis or PLC: 0-99

***Type* (INT):**

Type of the parameter that you want to read. It can have the following
values: Type = 0 - General
Type = 1 - Axis
Type = 2 - PLC

***Value*(REAL)**:

New value to be assigned to the indicated parameter.

5.5.2.3 Outputs

This function returns a boolean indicating that the parameter has been written.

5.5.2.4 Function Call

sfWriteParameter(Axis, Parameter, Type, Value);

5.5.3 Read H Register - sfReadHRegister

5.5.3.1 Operation

Returns the value of the H register indicated in the function input.

5.5.3.2 Entries

Register*(INT*):

Number of the H register you want to read. Accepts values from 0 to 999.

5.5.3.3 Outputs

Returns the value of the H register indicated in the "Register" argument. The returned value is of type REAL.

5.5.3.4 Function Call

sfReadHRegister (Register)

5.5.4 Write to H Register - sfWriteHRegister

5.5.4.1 Operation

Changes the value of the H register indicated in the function input.

5.5.4.2 Entries

Register*(INT*):

Number of the H register you want to change the value of. Accepts values from 0 to 999.

***Value*(REAL)**:

New value to be assigned to the H register indicated in the "Register" argument.

5.5.4.3 Outputs

This function returns a boolean indicating that the register has been written.

5.5.4.4 Function Call

sfWriteHRegister (Register, Value)

5.5.5 Read Reserved Variable - sfReadReservedVar

5.5.5.1 Operation

Returns the value of the reserved variable specified in the function input.
The list of existing reserved variables can be seen in the "Reserved Variables" manual.

5.5.5.2 Entries

***Var*(INT):**

Number of the reserved variable that you want to read. It accepts values from 10000 to 19999, which is the number of existing reserved variables. Not all reserved variable numbers perform some action.

5.5.5.3 Outputs

Returns the value of the reserved variable indicated in the "INT" argument. The returned value is of type REAL.

5.5.5.4 Function Call

sfReadReservedVar(Var)

5.5.6 Write to Reserved Variable - sfWriteReservedVar

5.5.6.1 Operation

Changes the value of the reserved variable indicated in the function input.
The list of existing reserved variables can be seen in the "Reserved Variables" manual.

5.5.6.2 Entries

***Var*(INT):**
Number of the reserved variable that you want to change the value of, values from 10000 to 19999 can be programmed, which are the numbers of existing reserved variables.

***Value*(REAL):**
New value to be assigned to the reserved variable indicated in the "INT" argument.

5.5.6.3 Outputs

This function returns a boolean indicating that the reserved variable has been written.

5.5.6.4 Function Call

sfWriteReservedVar(Var, Value)

5.6 Trigonometric/Mathematical

5.6.1 Sine - sfMathSin

5.6.1.1 Operation

Returns the sine value of the entered angle.

5.6.1.2 Entries

***Ang*(REAL):**

The actual value (in degrees) of the angle you want to calculate the sine.

5.6.1.3 Outputs

Returns the sine value of the angle that was entered into the "Ang" input.

5.6.1.4 Function Call

sfMathSin(Ang)

5.6.2 Cosine - sfMathCos

5.6.2.1 Operation

Returns the value of the cosine of the entered angle.

5.6.2.2 Entries

***Ang*(REAL):**

The actual value (in degrees) of the angle you want to calculate the cosine of.

5.6.2.3 Outputs

Returns the value of the cosine of the angle that was entered in the "Ang" input.

5.6.2.4 Function Call

sfMathCos(Ang)

5.6.4 Tangent - sfMathTan

5.6.4.1 Operation

Returns the value of the tangent of the entered angle.

5.6.4.2 Entries

***Ang*(REAL):**

The actual value (in degrees) of the angle you want to calculate the tangent.

5.6.4.3 Outputs

Returns the value of the tangent of the angle that was entered in the "Ang" input.

5.6.4.4 Function Call

sfMathTan(Ang)

5.6.5 Tangent Arc - sfMathAtan

5.6.5.1 Operation

Returns the arc tangent value of the entered value.

5.6.5.2 Entries

***Value*(REAL):**

Actual value that you want to calculate the arc tangent.

5.6.5.3 Outputs

Returns the arc tangent value of the number that was entered in the "REAL" input.

5.6.5.4 Function Call

sfMathAtan(Value)

5.6.6 Value of the Constant Pi - sfMathPi

5.6.6.1 Operation

Returns the value of the mathematical constant PI.

5.6.6.2 Entries

In this function there is no entry.

5.6.6.3 Outputs

Returns the value of the mathematical constant PI.

5.6.6.4 Function Call

sfMathPi()

5.6.7 Integer Value - sfMathTrunc

5.6.7.1 Operation

Returns the integer value of a number. Example: TRUNC(35.33) will return 35.

5.6.7.2 Entries

***Value*(REAL):**

Actual value from which you want to remove the decimal places.

5.6.7.3 Outputs

Returns the integer value of the number that was entered in the "REAL" input.

5.6.7.4 Function Call

sfMathTrunc(Value)

5.6.8 Fractional Value - sfMathFrac

5.6.8.1 Operation

Returns the value of the fractional part of a real number. Example: FRAC(35.33) will return 0.33.

5.6.8.2 Entries

***Value*(REAL):**

Actual value from which you want to get the fractional part.

5.6.8.3 Outputs

Returns a REAL number with only the fractional part of the number that was entered in the "REAL" input.

5.6.8.4 Function Call

FRAC(45,354)

5.6.9 Square Root - sfMathSqrt

5.6.9.1 Operation

Calculates the square root of the entered number.

5.6.9.2 Entries

***Value*(REAL):**

REAL value from which you want to get the square root.

5.6.9.3 Outputs

Returns the value of the square root of the number that was entered into the "REAL" input.

5.6.9.4 Function Call

SQRT(9)

5.6.10 Absolute Value - sfMathAbs

5.6.10.1 Operation

Returns the modulus of a real number.

5.6.10.2 Entries

***Value*(REAL):**

REAL value from which you want to get the modulus.

5.6.10.3 Outputs

Returns the modulus value of the number that was entered in the "REAL" input.

5.6.10.4 Function Call

ABS(-3)

5.7 Data type conversions

5.7.1 Boolean to Real - sfConvertBoolToReal

5.7.1.1 Operation

Converts a variable from type BOOL to type REAL.

5.7.1.2 Entries

***Value*(BOOL):**

Value of type BOOL that will be converted to type REAL.

5.7.1.3 Outputs

Returns the boolean value converted to REAL.

5.7.1.4 Function Call

sfConvertBoolToReal(Value)

5.7.6 Boolean to Integer - sfConvertBoolToInt

5.7.6.1 Operation

Converts a variable from type BOOL to type INT.

5.7.6.2 Entries

***Value*(BOOL):**

Value of type BOOL that will be converted to type INT.

5.7.6.3 Outputs

Returns the boolean value converted to INT.

5.7.6.4 Function Call

sfConvertBoolToInt(Value)

5.7.2 Real to Boolean - sfConvertRealToBool

5.7.2.1 Operation

Converts a variable from type REAL to type BOOL.

5.7.2.2 Entries

***Value*(REAL):**

Value of type REAL that will be converted to type BOOL.

5.7.2.3 Outputs

Returns the actual value converted to BOOL.

5.7.2.4 Function Call

sfConvertRealToBool(Value)

5.7.3 Real to Integer - sfConvertRealToInt

5.7.3.1 Operation

Converts a variable from type REAL to type INT.

5.7.3.2 Entries

***Value*(REAL):**

Value of type REAL that will be converted to type INT.

5.7.3.3 Outputs

Returns the actual value converted to INT.

5.7.3.4 Function Calls

sfConvertRealToInt(Value)

5.7.4 Real to Unsigned integer - sfConvertRealToUint

5.7.4.1 Operation

Converts a variable from type REAL to type UINT.

5.7.4.2 Entries

***Value*(REAL):**

Value of type REAL which will be converted to type UINT.

5.7.4.3 Outputs

Returns the actual value converted to UINT.

5.7.4.4 Function Calls

sfConvertRealToUint(Value)

5.7.5 Unsigned integer to Integer - sfConvertUintToInt

5.7.5.1 Operation

Converts a variable from type UINT to type INT.

5.7.5.2 Entries

***Value*(UINT):**

Value of type UINT that will be converted to type INT.

5.7.5.3 Outputs

Returns the UINT value converted to INT.

5.7.5.4 Function Call

sfConvertUintToInt(Value)

5.7.6 Integer to Unsigned Integer - sfConvertIntToUint

5.7.6.1 Operation

Converts a variable from type INT to type UINT.

5.7.6.2 Entries

***Value*(INT):**

Value of type INT that will be converted to type UINT.

5.7.6.3 Outputs

Returns the INT value converted to UINT.

5.7.6.4 Function Call

sfConvertIntToUint(Value)

5.7.7 Integer to Boolean - sfConvertIntToBool

5.7.7.1 Operation

Converts a variable from type INT to type BOOL.

5.7.7.2 Entries

***Value*(INT):**

Value of type INT that will be converted to type BOOL.

5.7.7.3 Outputs

Returns the INT value converted to BOOL.

5.7.7.4 Function Call

sfConvertIntToBool(Value)

5.7.8 Integer to Real - sfConvertIntToReal

5.7.8.1 Operation

Converts a variable from type INT to type REAL.

5.7.8.2 Entries

***Value*(INT):**

Value of type INT that will be converted to type REAL.

5.7.8.3 Outputs

Returns the INT value converted to REAL.

5.7.8.4 Function Call

sfConvertIntToReal(Value)

5.8 Axes

5.8.1 Axis Enabled - sfAxisEnabled

5.8.1.1 Operation

The function returns a boolean value indicating whether the indicated axis is enabled or not.

5.8.1.2 Entries

***Axis*(INT):**

Number of the axis you want to check. Accepts values from 1 to 32.

5.8.1.3 Outputs

Returns a boolean indicating whether the axis is enabled.

5.8.1.4 Function Call

sfAxisEnabled (Axis)

5.8.2 Axis Enabled for Manual Motion - sfAxisEnabledForManualMove

5.8.2.1 Operation

The function returns a boolean value indicating whether the indicated axis is enabled for manual movement or not.

5.8.2.2 Entries

***Axis*(INT):**

Number of the axis you want to check. Accepts values from 1 to 32.

5.8.2.3 Outputs

Returns a boolean indicating whether the axis is enabled for manual movement.

5.8.2.4 Function Call

sfAxisEnabledForManualMove(Axis)

5.8.3 Axis in Motion - sfAxisInMove

5.8.3.1 Operation

The function returns a boolean value indicating whether the indicated axis is moving or not.

5.8.3.2 Entries

***Axis*(INT):**

Number of the axis you want to check. Accepts values from 1 to 32.

5.8.3.3 Outputs

Returns a boolean indicating whether the axis is moving.

5.8.3.4 Function Call

sfAxisInMove(Axis)

5.8.4 Axis Defined - sfAxisIsDefined

5.8.4.1 Operation

The function returns a boolean value indicating whether the indicated axis is defined in the parameters or not.

5.8.4.2 Entries

***Axis*(INT):**

Number of the axis you want to check. Accepts values from 1 to 32.

5.8.4.3 Outputs

Returns a boolean indicating whether the axis is defined in the parameters.

5.8.4.4 Function Call

sfAxisIsDefined(Axis)

5.8.5 Spindle is Spindle - sfAxisIsSpindle

5.8.5.1 Operation

The function returns a boolean value indicating whether the indicated axis is defined as spindle or not.

5.8.5.2 Entries

***Axis*(INT):**

Number of the axis you want to check. Accepts values from 1 to 32.

5.8.5.3 Outputs

Returns a boolean indicating whether the axis is defined as spindle.

5.8.5.4 Function Call

sfAxisIsSpindle(Axis)

5.8.6 Axis is Referenced - sfAxisReferenced

5.8.6.1 Operation

This function allows you to tell the CNC whether the axis is referenced or to read its referencing status.

5.8.6.2 Entries

***Axis*(INT):**

Number of the axis you want to check. Accepts values from 1 to 32.

***Status*(BOOL):**

The referencing state you want to assign to the axis (if the "Type" entry is FALSE).

***Type*(BOOL):**

TRUE: Reads the referencing status of the axis.

FALSE: Sets new axis referencing state.

5.8.6.3 Outputs

Returns a boolean indicating whether the axis is referenced (if the "Type" input is TRUE). If the "Type" input is FALSE, the function sets a new referencing state for the axis and will always return FALSE.

5.8.6.4 Function Call

sfAxisReferenced(Axis, Status, Type)

5.8.7 Execute Preset - sfPresetAxis

5.8.7.1 Operation

This function performs a coordinate preset on the axis indicated at the "Axis" input.

5.8.7 .2 Inputs

***Axis*(INT):**

Number of the axis on which you want to make the preset.

***Coordinate* (REAL):**

Coordinate value for which you want to preset the axis.

***Origin*(INT):**

We can make this preset in 6 different sources (values 54 to 59). You can view these sources by entering the "Sources" screen of the AKC-PPC CNC (CNC screens object 361).

***Preset*(BOOL):**

The coordinate preset is executed on the selected axis and origin when a climb transition occurs on this input (transition from FALSE to TRUE).

5.8.7 .3 Outputs

This function returns a boolean that is TRUE for one PLC cycle, indicating that the preset has been executed.

5.8.7.4 Function call

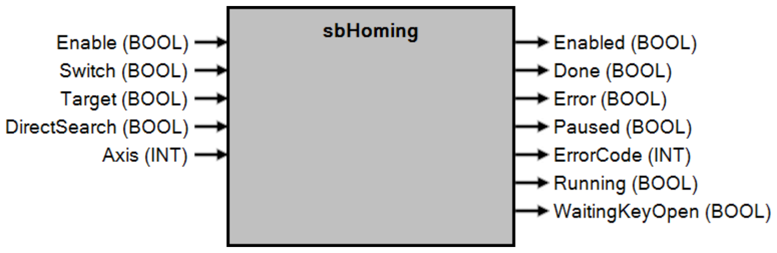
sfPresetAxis(Coordinate, Preset, Origin, Axis)

6.0 System Blocks

Blocks work like variables: you have to instantiate them.
The "instance" is an object, that is, a copy of the main structure. Therefore, you must create a variable with a user-defined name and select the type of this variable according to the functional block that you want to instantiate.

6.1 Axes

6.1.1 Referencing - sbHoming



6.1.1.1 Operation

The function of this block is to tell the CNC a defined position that will be used by the CNC as a reference for the indicated axis.
In the ISO programming system (G-Codes) this data is related to the G53 origin. The parameters P09 and P10 of the indicated axis define the reference position.
Parameter P36 defines the reference mark type, to use this block "PLC" must be selected.
Parameter P37 defines the polarity of the reference mark.
Parameter P39 defines the direction of the movement that the axis will search for reference. Parameter P40 defines the speed of the motion in the reference search.
Parameter P41 defines the reverse speed of the motion in the reference fetch.
The referencing procedure must precede any kind of automatic or manual operation that involves movements to absolute positions.
To use the CNC software stroke limits, set in parameters P48 to P51 of the indicated axis, this procedure must also have been completed.

6.1.1.2 Entries

***Enable* (BOOL):**
This input enables/starts the reference homing process for the indicated axis when it receives the value TRUE. The system variable *svHomingMode* is set to TRUE during this process.
To cancel the reference search process set the value FALSE to this entry.

***Switch* (BOOL):**
This input receives the reference signal that the block will wait for when it starts the process and the axis moves. This input will normally be tied to a digital input on the drive or card.

***Target* (BOOL):**
This entry defines whether the reference search will be done through targets.
If the value of this input is FALSE the CNC will do the normal homing process, moving the axis until the Switch input detects the datum.
If the value of this input is TRUE the CNC assumes that the axes are already positioned at the reference position, so there will be no movement.

***DirectSearch* (BOOL):**

This input defines the direction in which the homing will search for the falling edge of the homing signal. If the value of this input is FALSE, when the homing signal is encountered the CNC reverses the direction of motion, so the axis moves in the opposite direction until the homing signal disappears.
If the value of this input is TRUE when it encounters the reference signal, the CNC continues moving in the same direction until the reference signal disappears.

***Axis* (REAL):**

This entry defines the number of the axis that will be referenced (1 to 32).

6.1.1.3 Outputs

***Enabled* (BOOL):**

This output indicates whether the block is enabled if it has been initialized without errors.

***Running* (BOOL):**

This output indicates whether the CNC is running the homing process on the defined axis.

***Paused* (BOOL):**

This output indicates whether the reference search process paused.

***Error* (BOOL):**

This output indicates whether an error occurred during the reference search process.

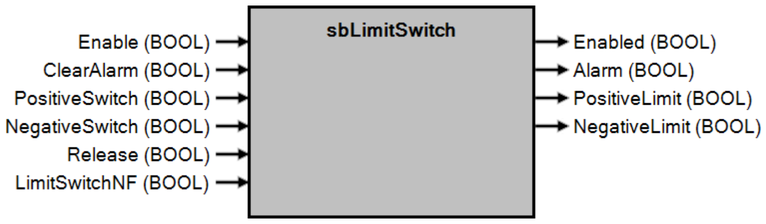
***ErrorCode* (INT):**

The "ErrorCode" output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.1.1.4 Block Call

```
sbHoming(Enable, Switch, Target, DirectSearch, Axis)
```

6.1.2 Limit Switch - sbLimitSwitch



6.1.2.1 Operation

The function of this block is to indicate whether a physical limit switch has been actuated.
If one of the limits is triggered after CNC initialization, the "Alarm" output is also triggered.

6.1.2.2 Entries

***Enable* (BOOL):**
This entry enables the block.
If the value is FALSE the block is disabled and all outputs get null values.

***LimitSwitchNF* (BOOL):**
This input sets the polarity when detecting the end of stroke.
If the value of this input is TRUE the identification of end of stroke faults will occur when the inputs "NegativeSwitch" or "PositiveSwitch" are in the FALSE condition.
If the value of this input is FALSE the identification of end of stroke faults will occur when the inputs "NegativeSwitch" or "PositiveSwitch" are in the TRUE condition.

***NegativeSwitch* (BOOL):**
This input must be assigned to the digital input that is connected to the negative limit sensor.

***PositiveSwitch* (BOOL):**
This input must be assigned to the digital input that is connected to the positive limit sensor.

***Release* (BOOL):**
This entry defines whether all errors generated by this block will be ignored.

***ClearAlarm* (BOOL):**
This input removes the error indication from the output.
Transitioning from the FALSE to TRUE state of the "ClearAlarm" input removes the error indication from the "Alarm" output. If the condition that caused the alarm is still active, ClearAlarm will not remove the error indication.

6.1.2.3 Outputs

***Enabled* (BOOL):**
This output indicates whether the block is enabled.

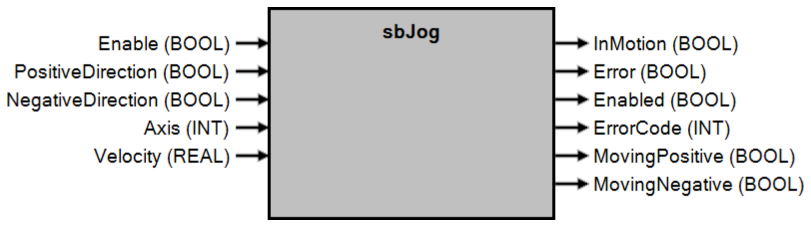
***Alarm* (BOOL):**
This output indicates whether a limit switch is active; this output is only used when the CNC is initialized. If the CNC is not initialized, even with a triggered limit switch, this output remains in the FALSE state. ***NegativeSwitch* (BOOL):**
This output indicates whether the negative limit switch is actuated.

***PositiveSwitch* (BOOL):**
This output indicates whether the positive limit switch is actuated.

6.1.2.4 Block Call

sbLimitSwitch(Enable, ClearAlarm, PositiveSwitch, NegativeSwitch, Release, LimitSwitchNF)

6.1.3 JOG - sbJog



6.1.3.1 Operation

The function of this block is to perform speed-controlled movements on the defined axis in both directions, while either the "NegativeDirection" entry or the "PositiveDirection" entry is set to TRUE.
If the CNC is referenced and the software travel limits are programmed, when using this block, the CNC will respect these limits and will not let the axis go beyond the travel limits.
If the CNC is not referenced and it is necessary
Usually we use this block with its motion inputs connected to digital inputs, some button or softkey.

6.1.3.2 Entries

***Enable* (BOOL):**
This input enables the block to execute manual axis movement when the value of this input is TRUE. If the value of this input is FALSE, the block is disabled and all outputs receive null values.

***Axis* (INT):**
This entry defines the number of the axis (1 to 32) that you want to move.

***NegativeDirection* (BOOL):**
This input prompts the block to execute the movement in the negative direction of the axis defined in the "Axis" input. Motion remains executed as long as this argument is TRUE. When it assumes the status FALSE, the movement is stopped.

***PositiveDirection* (BOOL):**
This input asks the block to execute the movement in the positive direction of the axis defined in the "Axis" input. Motion remains executed as long as this argument is TRUE. When it assumes the status FALSE, the movement is stopped.

***Velocity* (INT):**
This input defines the speed at which the movement will be executed.
If this input is set to 0, the speed used for the movements will be the speed defined in parameter P27 of the defined axis.

6.1.3.3 Outputs

***Enabled* (BOOL):**
This output indicates whether the block is enabled.

***InMotion* (BOOL):**
This output indicates whether the axis defined at the "Axis" input is moving.

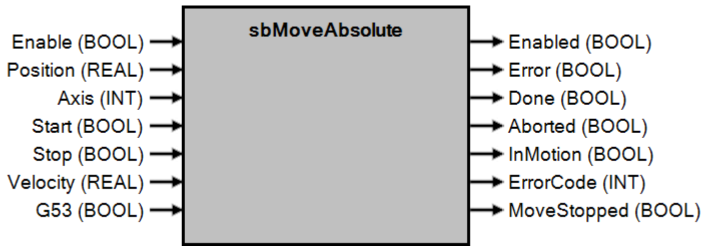
***Error* (BOOL):**
This output indicates whether an error occurred during block processing.

***ErrorCode* (INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.1.3.4 Block Call

sbJog(Enable, PositiveDirection, NegativeDirection, Axis, Velocity)

6.1.4 Absolute Motion - sbMoveAbsolute



6.1.4.1 Operation

The function of this block is to perform a speed-controlled movement in the axis defined in the "Axis" input of the current position of the axis to the position set at the "Position" input.

6.1.4.2 Entries

***Enable* (BOOL):**

This entry enables the block.

If this input is FALSE, the block is disabled and all outputs are given null values.

***Axis* (INT):**

This entry defines the number of the axis that is to be moved.

***Position* (REAL):**

This entry" defines the final absolute position where the defined axis will be positioned.

***Start* (BOOL):**

This input prompts the block to start moving to the position indicated in "Position".

***Stop* (BOOL):**

This input prompts the block to stop the movement that is being executed.

***Velocity* (INT):**

This input defines the speed at which the movement will be executed.

If this input is set to 0, the speed used for the movements will be the speed defined in parameter P27 of the defined axis.

***G53* (BOOL):**

This entry defines whether the movement will be made relative to the G53 origin.

If this entry is FALSE the movement will be made relative to the selected origin (G54, G55, G56 or G57).

6.1.4.3 Outputs

***Enabled* (BOOL):**

This output indicates whether the block is enabled and ready for use.

MoveStopped (BOOL):

This output indicates whether the movement that was being executed has been canceled.

Aborted (BOOL):

This output indicates whether a motion that was being executed has been stopped. This output is activated after the motion cancellation is output.

***Done* (BOOL):**

This output indicates whether the requested movement has been completed.

***InMotion* (BOOL):**

This output indicates whether the set axis" is moving.

***Error* (BOOL):**

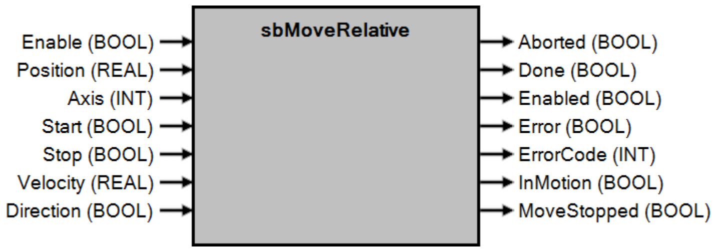
This output indicates whether an error occurred during the execution of the block.

ErrorCode **(INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.1.4.4 Block Call

sbMoveAbsolute(Enable, Position, Axis, Start, Stop, Velocity, G53)

6.1.5 Incremental Movement - sbMoveRelative



6.1.5.1 Operation

The function of this block is to perform a speed-controlled movement on the axis defined at the "Axis" input from the current axis position to the position defined at the "Position" input.

6.1.5.2 Entries

***Enable* (BOOL):**

This entry enables the block.

If this input is FALSE, the block is disabled and all outputs are given null values.

***Axis* (INT):**

This entry defines the number of the axis that is to be moved.

***Position* (REAL):**

This input defines the value of the increment that will be made to the current position of the axis. This value can be positive to increment the position, or negative to decrement the current position.

***Direction* (BOOL):**

This entry defines how the increment will be made.

If this input is FALSE then the increment used will be the one defined in the "Position" input.

If this input is TRUE and the increment defined at the "Position" input is positive, the movement will be made to the negative sense.

***Start* (BOOL):**

This entry requests to start the movement to increment the position.

***Stop* (BOOL):**

This input prompts you to stop the motion that is being executed.

***Velocity* (INT):**

This input defines the speed at which the movement will be executed.

If this input is set to 0, the speed used for the movements will be the speed defined in parameter P27 of the defined axis.

6.1.5.3 Enabled

outputs (BOOL):

This output indicates whether the block is enabled.

MoveStopped (BOOL):

This output indicates whether the movement that was being executed has been canceled.

Aborted (BOOL):

This output indicates whether a motion that was being executed has been stopped. This output is activated after the motion cancellation is output.

Done (BOOL):

This output indicates whether the requested movement has been completed.

InMotion (BOOL):

This output indicates whether the associated "Axis" is moving.

***Error* (BOOL):**

This output indicates whether an error occurred during block processing.

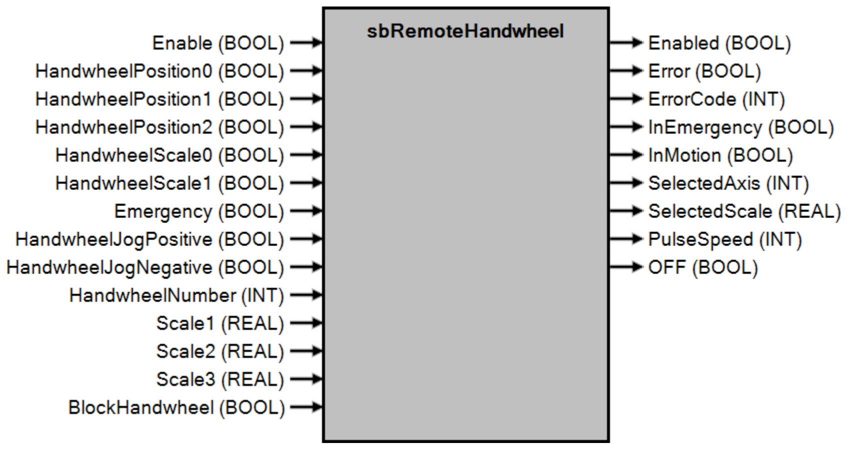
- **ErrorCode (INT):**

This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.1.5.4 Block Call

sbMoveRelative(Enable, Position, Axis, Start, Stop, Velocity, Direction)

6.1.6 Remote Crank - sbRemoteHandwheel



6.1.6.1 Operation

The function of this block is to control the remote handle available to
the CNC. The remote handle is hardware that is sold separately from
the CNC.
With the remote crank the CNC makes incremental or JOG movements according to the selected scale. The remote crank has an emergency button.
The axis selector has 8 positions, allowing you to select up to 7 axes.
The scale selector has 4 positions, 3 increment and JOG scale values. The CNC
can use up to 3 cranks.
The remote cranks must be configured in the general parameters.
The setting for Remote Handle 1 must be made in the general parameters P36 to P39. The setting for Remote Handle 2 must be made in the general parameters P40 to P43. The setting for Remote Handle 3 must be made in the general parameters P44 to P47.

6.1.6.2 Entries

***Enable* (BOOL):**
This input enables the block, if this signal is "FALSE" the crank is disabled.
***HandwheelPosition0* (BOOL):**
This input sets bit 0 of the remote crank axis selection. This input will use a digital input from the CNC that will be associated with it in the PLC.
***HandwheelPosition1* (BOOL):**
This input sets bit1 of the remote crank axis selection. This input will use a digital input from the CNC that will be associated with it in the PLC.
***HandwheelPosition2* (BOOL):**
This input sets bit 2 of the remote crank axis selection. This input will use a digital input from the CNC that will be associated with it in the PLC.
***HandwheelScale0* (BOOL):**
This input sets bit 0 of the remote crank scaling selection. This input will use a digital input from the CNC that will be associated with it in the PLC.
***HandwheelScale1* (BOOL):**
This input sets bit 1 of the remote crank scaling selection. This input will use a digital input from the CNC that will be associated with it in the PLC.
***Emergency* (BOOL):**
This input defines the emergency button of the remote crank must be associated with this block input. This input will use a digital input from the CNC that will be associated with it in the PLC.

***HandwheelJogPositive* (BOOL):**
This input defines the positive motion knob to be associated with this block input. This input will use a digital input from the CNC that will be associated with it in the PLC. This input performs either JOG or incremental movements, depending on the type of scaling that is selected.
***HandwheelJogNegative* (BOOL):**
This input defines the negative motion knob to be associated with this block input. This input will use a digital input from the CNC that will be associated with it in the PLC. This input performs either JOG or incremental moves, depending on the type of scaling that is selected.
***HandwheelNumber* (INT):**
This entry sets the remote crank number (1 to 3, as mentioned earlier).
***Scale1* (REAL):**
This input defines the value of the increment that will be used in Scaling 1.
***Scale2* (REAL):**
This input defines the value of the increment that will be used in Scaling 2.
***Scale3* (REAL):**
This input defines the value of the increment that will be used in Scaling 3.
***BlockManWheel* (BOOL):**
The entry "BlockManWheel " is a boolean that blocks the operation of the crank.

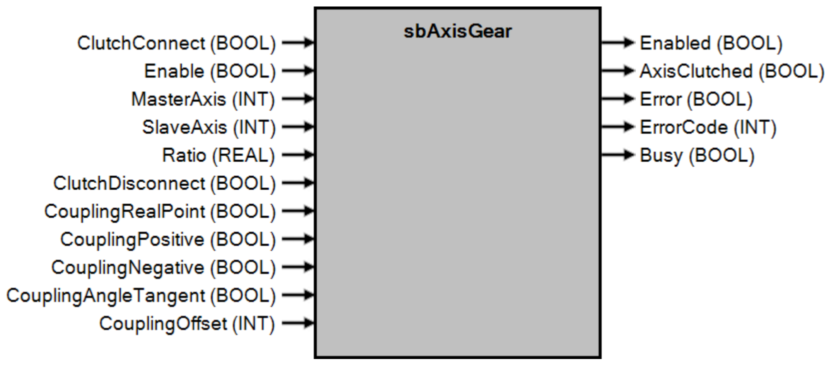
6.1.6.3 Outputs

***Enabled* (BOOL):**
This output indicates whether the block is enabled.
***InEmergency* (BOOL):**
This output indicates whether the emergency crank button is pressed.
***InMotion* (BOOL):**
This output indicates whether the selected axis is moving.
***OFF* (BOOL):**
This output indicates if no axis is selected at the remote crank.
***PulseSpeed* (INT):**
This output indicates the remote crank speed in pulses per sample every 16ms.
***SelectedAxis* (INT):**
This output indicates which axis is selected.
***SelectedScale* (REAL):**
This output indicates the value of the scale that is selected.
***Error* (BOOL):**
This output indicates whether an error occurred during the execution of the block.
***ErrorCode* (INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.1.6.4 Block Call

sbRemoteHandwheel(Enable, HandwheelPosition0, HandwheelPosition1, HandwheelPosition2, HandwheelScale0, HandwheelScale1, Emergency, HandwheelJogPositive, HandwheelJogNegative, HandwheelNumber, Scale1, Scale2, Scale3, BlockHandwheel)

6.1.7 Shaft Coupling - sbAxisGear



6.1.7.1 Operation

This block has the function of coupling a slave axis to a master axis.
With this you can make a slave axis follow the movements of the master with a defined coupling relationship. The coupling relationship between the axes can be positive or negative. If it is positive, the slave axis follows the master axis in the same direction as the master's movement. Negative values cause the slave axis to move in the opposite direction as the master axis moves.
The coupling ratio module determines the relationship between the movement of the master and slave axes. If the coupling ratio is greater than 0 and less than 1 it causes the slave axis to move less than the master axis moves. For example, for a ratio of 0.5, for every millimeter of movement of the master shaft, the slave shaft will move 0.5 mm.
If the coupling ratio is equal to 1 the slave axis will move at the same rate as the master axis. If the coupling ratio is greater than 1 the slave shaft will move more than the master shaft. For example, for a ratio equal to 2, for every millimeter of movement of the master shaft, the slave shaft will move 2 mm.

6.1.7.2 Entries

***Enable* (BOOL):**
This entry enables the block.
If this input is FALSE, the block is disabled and all outputs are given null values.
***ClutchConnect* (BOOL):**
This input requests the CNC to couple the slave axis to the master taking into account all the values assigned to the rest of the inputs.
***ClutchDisconnect* (BOOL):**
This input requests the CNC to decouple the slave axis from the master, leaving it free to move independently of the master.
***CouplingAngleTangent* (BOOL):**
This input defines whether the coupling of the slave axis to the master will be tangential.
When requesting a coupling, if the "CouplingAngleTangent" input is set to TRUE, the slave axis will follow tangent to the path in the plane defined by the CNC via ISO functions G17 (XY plane), G18 (YZ plane) or G19 (ZX plane), i.e., in the case of the XY plane with the C (rotary) axis being the slave axis, if XY performs a movement making a circle, the C axis would make a tangent movement to the circle described by XY.
***CouplingNegative* (BOOL):**
This input defines whether the slave axis will only be coupled to the master when the master moves in the negative direction.
***CouplingPositive* (BOOL):**
This input defines whether the slave axis will only be coupled to the master when the master moves in the positive direction.

CouplingOffset **(INT):**
This input defines whether the CNC will make a movement that is executed on the slave axis relative to the master axis. This movement is executed only once with each coupling request.
CouplingRealPoint **(BOOL):**
This input defines whether the CNC will do the actual point coupling, i.e. is it coupled to an encoder, resolver or other position transducer.
When requested to couple with this argument set to FALSE the coupling is done by the CNC's theoretical point, that is, it is coupled to the point calculated by the CNC.
MasterAxis **(INT):**
This entry defines the axis number corresponding to the master axis.
SlaveAxis **(INT):**
This entry defines the axis number corresponding to the master axis.
Ratio **(REAL):**
This input defines the coupling factor between slave and master.
If a linear axis is attached to a rotary axis, each degree corresponds to 0.0001 mm.

6.1.7.3 Outputs

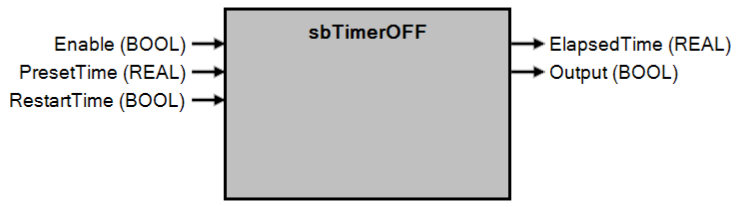
Enabled **(BOOL):**
This output indicates whether the block is enabled.
AxisClutched **(BOOL):**
This output indicates whether the slave axis is coupled to its master axis.
Error **(BOOL):**
This output indicates whether an error occurred during block processing.
ErrorCode **(INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.1.7.4 Block Call

sbAxisGear(ClutchConnect, Enable, MasterAxis, SlaveAxis, Ratio, ClutchDisconnect, CouplingRealPoint, CouplingPositive, CouplingNegative, CouplingAngleTangent, CouplingOffset)

6.2 Timers

6.2.1 Timer (Normally Off) - sbTimerOFF



6.2.1.1 Operation

This block has the function of counting time and activating an output at the end of the count.
When the block is enabled, the "Output" output changes its state to TRUE and stays in this condition for the time duration defined in "PresetTime". After this time, the output goes to the FALSE state.
To make a new time count keep the block enabled and send a pulse of at least one cycle on the "RestartTime" entry. The count will restart when the "RestartTime" input returns to the FALSE state.

6.2.1.2 Entries

***Enable* (BOOL):**
This entry enables the block.
If this input is FALSE, the block is disabled and all outputs are given null values.
***PresetTime* (REAL):**
This input sets the time to be counted in the timer, in milliseconds.
***RestartTime* (BOOL):**
This entry resets the time count.

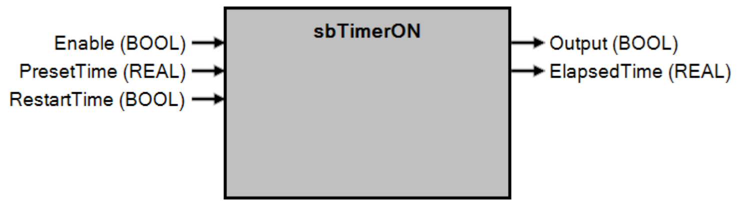
6.2.1.3 Outputs

***ElapsedTime* (REAL):**
This output indicates the elapsed time in the current count in milliseconds.
***Output* (BOOL):**
This output indicates the status of the time counting.

6.2.1.4 Block Call

sbTimerOFF(Enable, PresetTime, RestartTime)

6.2.2 Timer (Normally On) - sbTimerON



6.2.2.1 Operation

This block has the function of counting time and activating an output at the end of the count.
When the block is enabled, the "Output" output changes its state to FALSE and stays in this condition for the time duration defined in "PresetTime". After this time, the output goes to the TRUE state.
To make a new time count keep the block enabled and send a pulse of at least one cycle
on the "RestartTime" entry. The count will restart when the "RestartTime" input returns to the FALSE state.

6.2.2.2 Entries

***Enable* (BOOL):**
This entry enables the block.
If this input is FALSE, the block is disabled and all outputs are given null values.
***PresetTime* (REAL):**
This input sets the time to be counted in the timer, in milliseconds.
***RestartTime* (BOOL):**
This entry resets the time count.

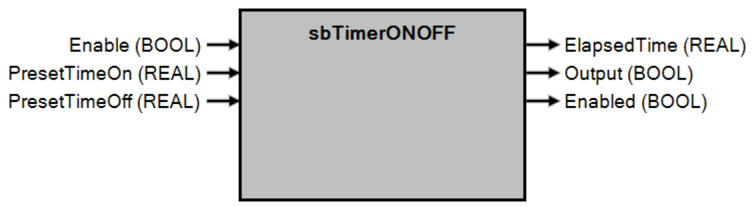
6.2.2.3 Outputs

***ElapsedTime* (REAL):**
This output indicates the elapsed time in the current count in milliseconds.
***Output* (BOOL):**
This output indicates the status of the time counting.

6.2.2.4 Block Call

sbTimerON(Enable, PresetTime, RestartTime)

6.2.3 Timer (Blinking) - sbTimerONOFF



6.2.3.1 Operation

This block has the function of counting time and activating an output cyclically.
When the block is enabled, the "Output" output remains in TRUE for the time set at the "PresetTimeON" input. At the end of this time the output goes to FALSE and remains so for the time set at the "PresetTimeOFF" input. The count is cycled until the "Enable" input is disabled (FALSE).

6.2.3.2 Entries

***Enable* (BOOL):**
This entry enables the block.
If the value is FALSE the block is disabled and all outputs get null values.
***PresetTimeON* (REAL):**
This input sets the time in milliseconds to be counted in the timer with the "Output" output set to TRUE.
***PresetTimeOFF* (REAL):**
This input sets the time in milliseconds to be counted in the timer with the "Output" output set to FALSE.

6.2.3.3 Outputs

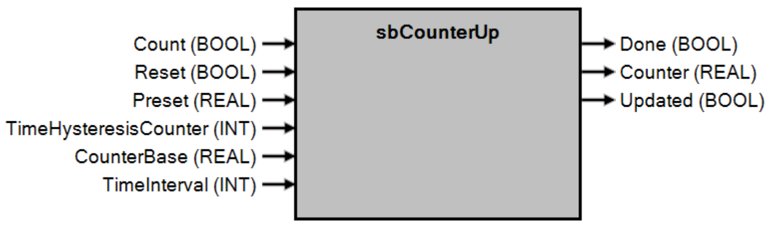
***Enabled* (BOOL):**
This output indicates whether the block is enabled.
***ElapsedTime* (REAL):**
This output indicates the elapsed time in milliseconds. It shows the time that has elapsed from the moment the timer started counting. The value goes from 0 to PresetTimeOFF + PresetTimeON, and then resets to zero.
***Output* (BOOL):**
This output indicates the status of the time counting.

6.2.3.4 Block Call

sbTimerONOFF(Enable, PresetTimeOn, PresetTimeOff)

6.3 Counters

6.3.1 Countdown Counter - sbCounterUp



6.3.1.1 Operation

This block has the function of counting a numerical value in an ascending form.
On receiving a rising transition on the "Count" input the "Counter" output will be incremented with the value set on the "CounterBase" input.
If the "TimeInterval" input is set to zero, by holding the "Count" input to TRUE, the counter is not incremented again, i.e. the count only stays on the up transition of the "Count" input.
If "TimeInverval" is set to a value other than 0, the value is automatically incremented by the time given in "TimeInterval" if the "Count" input is set to TRUE.
When the value of the "Counter" output is greater than or equal to the "Preset" input the "Done" output will go to TRUE. Even after the Done output equals TRUE, if the Count input is triggered, the Counter value continues to increment until the block receives TRUE at the Reset input.

6.3.1.2 Entries

***Count* (BOOL):**
This input performs the count. When this input receives a rising transition the "Counter" output will be incremented by the value set on the "CounterBase" input.
***Reset* (BOOL):**
This input resets the counter, i.e. the "Counter" output goes to the value 0.
***Preset* (REAL):**
This input sets the target value of the count. Upon reaching this value the "Done" output goes to the TRUE state.
***TimeHysteresisCounter* (INT):**
This input sets the hysteresis time in milliseconds for the Count input. This time is the delay to start the automatic increment if the Count input remains TRUE.
***CounterBase* (REAL):**
This input defines the base value that will be used to increment the counter.
***TimeInterval* (INT):**
This input sets the time interval in milliseconds to perform automatic increments if the "Count" input is set to TRUE. If this argument is zero, automatic incrementing does not take place.

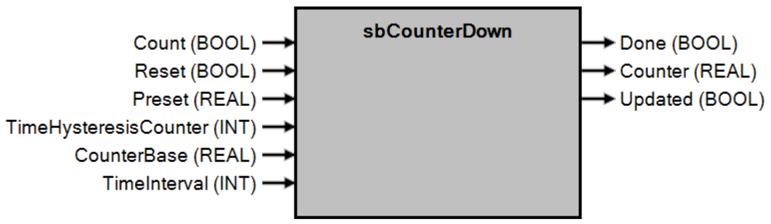
6.3.1.3 Outputs

***Done* (BOOL):**
This output indicates whether the "Counter" output has reached the value set on the "Preset" input.
***Counter* (REAL):**
This output indicates the current count value. Even after reaching the value of the "Preset" input, the count continues, if
the block is not reset via the "Reset" input.
***Updated* (BOOL):**
Indicates that the "Counter" output value has been updated. Remains active for one PLC cycle.

6.3.1.4 Block Call

sbCounterUp(Count, Reset, Preset, TimeHysteresisCounter, CounterBase, TimeInterval)

6.3.2 Countdown Counter - sbCounterDown



6.3.2.1 Operation

This block has the function of counting a numerical value in a descending order.
On receiving a rising transition on the "Count" input, the "Counter" output will be decremented by the set value in the "CounterBase" entry.
If the "TimeInterval" input is set to zero, by keeping the "Count" input at TRUE, the counter is not decremented again, i.e. the count only stays on the up transition of the "Count" input.
If "TimeInverval" is set to a value other than 0, the value is automatically decremented by the time given in "TimeInterval" if the "Count" input is set to TRUE.
When the value of the "Counter" output is less than or equal to zero, the "Done" output will go to TRUE.
Even after the "Done" output equals TRUE, if the "Count" input is triggered, the value of "Counter" continues to be decremented until the block receives TRUE on the "Reset" input.

6.3.2.2 Entries

***Count* (BOOL):**
This input performs the count. When this input receives an up transition the output "Counter" will be decremented by the value set on the "CounterBase" input.

***Reset* (BOOL):**
This input resets the counter, i.e. the "Counter" output receives the value set on the "Preset" input.

***Preset* (REAL):**
This input sets the target value of the count. Upon reaching this value the "Done" output goes to the TRUE state.

***TimeHysteresisCounter* (INT):**
This input sets the hysteresis time in milliseconds for the Count input. This time is the delay to start the automatic decrement if the Count input remains TRUE.

***CounterBase* (REAL):**
This input defines the base value that will be used to decrement the counter.

***TimeInterval* (INT):**
This input sets the time interval in milliseconds to execute automatic decrements if the "Count" input is set to TRUE. If this argument is zero, the automatic decrement does not happen.

6.3.2.3 Outputs

***Done* (BOOL):**
This output indicates whether the "Counter" output has reached the value zero.

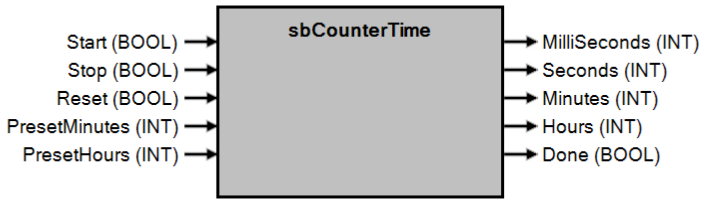
***Counter* (REAL):**
This output indicates the current count value. Even after reaching the value of the "Preset" input, the count continues, if
the block is not reset via the "Reset" input.

***Updated* (BOOL):**
Indicates that the "Counter" output value has been updated. Remains active for one PLC cycle.

6.3.2.4 Block Call

sbCounterDown(Count, Reset, Preset, TimeHysteresisCounter, CounterBase, TimeInterval)

6.3.3 Time Counter - sbCounterTime



6.3.3.1 Operation

This block has the function of counting time, as in a stopwatch.
A rising transition on the "Start" input starts the time counting, showing on the outputs of the block the times in milliseconds, seconds, minutes, and hours.
An up transition at the "Stop" input causes the count to be paused. To restart it, the block must receive a new up transition at the "Start" input.
A rising transition on the "Reset" input, restarts the time counting of all outputs.
When the value of the preset inputs (minutes and hours) are reached, the output "Done" will change to the TRUE state. Even after these inputs are reached, the time will continue to be counted (The counter will only stop when it receives an up transition at the "Stop" input.
The total counting time is determined by the combination of the 4 time outputs (milliseconds, seconds, minutes, and hours).

6.3.3.2 Entries

***Start* (BOOL):**
This entry starts time counting.
***Stop* (BOOL):**
This entry interrupts the time counting.
***Reset* (BOOL):**
This entry resets the time count.
***PresetMinutes* (INT):**
This entry sets the target value to minutes.
***PresetHours* (INT):**
This entry sets the target value for hours.

6.3.3.3 Outputs

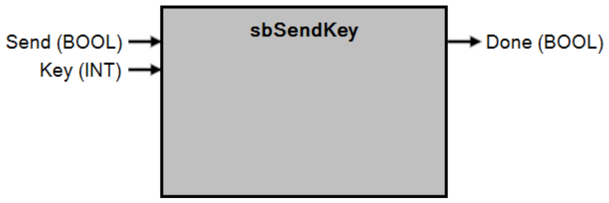
***MilliSeconds* (INT):**
This output indicates the elapsed time in milliseconds (0-999).
***Seconds* (INT):**
This output indicates the elapsed value in seconds (0-59).
***Minutes* (INT):**
This output indicates the elapsed value in minutes (0-59).
***Hours* (INT):**
This output indicates the elapsed value in hours (0-1092).
***Done* (BOOL):**
This output indicates whether the values programmed in the "PresetMinutes" and "PresetHours" inputs have been reached.

6.3.4.4 Block Call

sbCounterTime(Start, Stop, Reset, PresetMinutes, PresetHours)

6.4 Codes and Softkeys

6.4.1 Send Code - sbSendKey



6.4.1.1 Operation

The function of this block is to send the code defined at the "Key" input to the CNC. The code can be a screen (focus) code, character (ASCII or non-ASCII) code, operation mode code, or any other CNC key code.
If the code is for a screen, the CNC will update the current screen. If it is a mode, it updates the current mode of operation. If it is a key (ASCII or non-ASCII), it is as if that specific key has been pressed at the CNC.

6.4.1.2 Entries

***Key* (INT):**
This input defines the code to be sent to the CNC. It can be a code for screens (focus), characters (ASCII or non-ASCII), operation mode, or any other CNC code.

***Send* (BOOL):**
This entry executes the sending of the code to the CNC.
On the up transition of this input, the code defined on the "Key" input will be sent to the CNC.

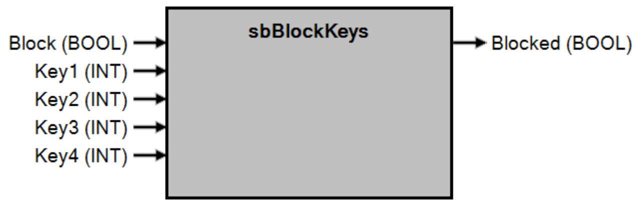
6.4.1.3 Outputs

***Done* (BOOL):**
This output indicates whether the signal has been sent to the CNC.

6.4.1.4 Block Call

sbSendKey(Send, Key)

6.4.2 Lock Code - sbBlockKeys



6.4.2.1 Operation

This block has the function of blocking the receipt of the codes defined in inputs "Key1", "Key2", "Key3" and "Key4" to the CNC. The code can be a screen code (focus), character code (ASCII or non-ASCII), operation mode code, or any other
CNC keys.
When you want to lock 3 keys or less, simply set the other "Key" inputs to 0.

6.4.2.2 Entries

***Key1* (INT):**
This entry defines the code of the key you want to lock (1)

***Key2* (INT):**
This entry defines the code of the key you want to lock (2)

***Key3* (INT):**
This entry defines the code of the key you want to lock (3)

***Key4* (INT):**
This entry defines the code of the key you want to lock (4)

***Block* (BOOL):**
This input locks the keys of the codes that have been set in inputs "Key1", "Key2", "Key3" and "Key4". The lock is active when the value of this input is TRUE,

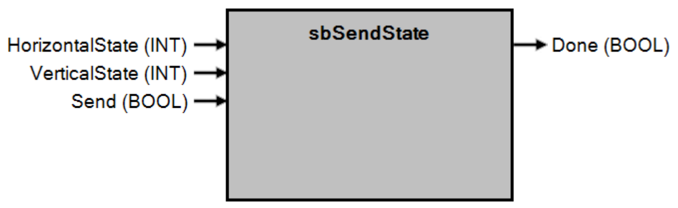
6.4.2.3 Outputs

***Blocked* (BOOL):**
This output indicates whether the block is blocking the codes that have been set in inputs "Key1", "Key2", "Key3" and "Key4".

6.4.2.4 Block Call

sbBlockKeys(Block, Key1, Key2, Key3, Key4)

6.4.3 Send Softkey State - sbSendState



6.4.3.1 Operation

This block changes the current state of the CNC softkeys - both horizontal and vertical.

6.4.3.2 Entries

***HorizontalState* (INT):**

This entry defines the number of the horizontal softkey state that will be selected.

***VerticalState* (INT):**

This entry defines the number of the vertical softkey state that will be selected.

***Send* (BOOL):**

This entry prompts you to change the state of the softkeys.
An up transition on this input changes the current state of the CNC softkeys to the states defined in the "HorizontalState" and "VerticalState" inputs.

6.4.3.3 Outputs

***Done* (BOOL):**

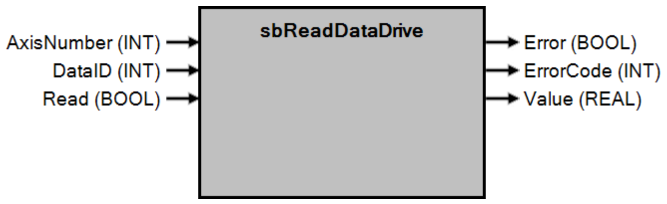
This output indicates whether the block was executed, so the softkey states have changed to the defined states in the "HorizontalState" and "VerticalState" entries.

6.4.3.4 Block Call

sbSendState(HorizontalState, VerticalState, Send)

6.5 Drives

6.5.1 Read Data from Drive - sbReadDataDrive



6.5.1.1 Operation

This block has the function of reading information from the AKD drives. It
can read position, speed and torque.
To read one of this information, simply enter the drive number in the "AxisNumber" input, the type of information in "DataID" and set the "Read" input to TRUE. Then the value is returned in the output "Value".

6.5.1.2 Entries

***AxisNumber* (INT):**
This input defines the number of the drive that you want to read information from. To find out the drive number, check parameter 12 (EtherCAT Drive Number / Analog Channel) of the corresponding axis.

***DataID* (INT):**
This input defines the type of data you want to read. Possible values are:
0 = Position
1 = Speed
2 = Torque

***Read* (BOOL):**
This entry prompts you to read the data.

6.5.1.3 Outputs

***Value* (REAL):**
This output returns the value of the requested data.

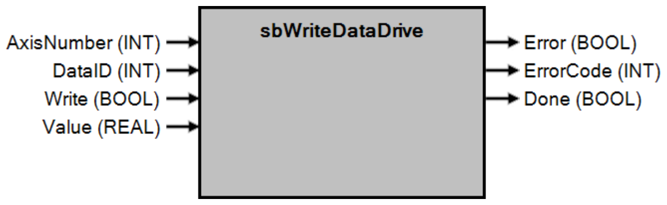
***Error* (BOOL):**
This output indicates whether an error occurred during block processing.

***ErrorCode* (INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.5.1.4 Block Call

sbReadDataDrive(AxisNumber, DataID, Read)

6.5.2 Write Data from Drive - sbWriteDataDrive



6.5.2.1 Operation

This block has the function of writing information to the AKD drives. Currently, you
can only write the value to the drive's analog output.
To write this information, simply set the drive number in the "AxisNumber" entry, the information ID in "DataID", assign the new value to the "Value" entry, and set the "Write" entry to TRUE.

6.5.2.2 Entries

***AxisNumber* (INT):**
This input defines the number of the drive that you want to write the information to. To find out the drive number, check parameter 12 (EtherCAT Drive Number / Analog Channel) of the corresponding axis.

***DataID* (INT):**
This input defines the type of data you want to write. Possible values are: 0 =
Analog output of the drive

***Write* (BOOL):**
This entry prompts you to write the data.

***Value* (REAL):**
This input defines the value that will be written to the drive.

6.5.2.3 Outputs

***Done* (BOOL):**
This output indicates whether the request to write to the drive was sent.

***Error* (BOOL):**
This output indicates whether an error occurred during block processing.

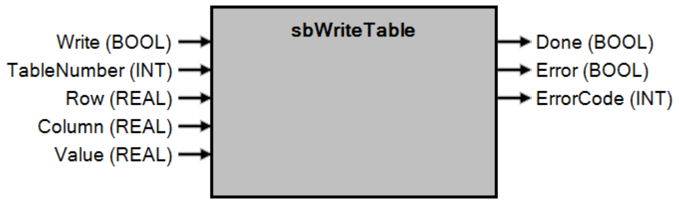
***ErrorCode* (INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.5.2.4 Block Call

sbWriteDataDrive(AxisNumber, DataID, Write, Value)

6.6 Table Access

6.6.1 Write to Table Field - sbWriteTable



6.6.1.1 Operation

The function of this block is to write data to CNC tables.
To write a value to a table cell, you must enter in the block entries the table number, column and row where you want to write the new value. You must also enter the value that will be written to that particular cell.

6.6.1.2 Entries

***Write* (BOOL):**
This input performs the write to the table when it receives a rise transition.

***TableNumber* (BOOL):**
This entry defines the number of the table where you want to write the data.

***Row* (INT):**
This entry defines the row number of the table.

***Column* (INT):**
This entry defines the column number of the table.

***Value* (REAL):**
This input defines the value to be written into the chosen cell.

6.6.1.3 Outputs

***Done* (BOOL):**
This output indicates whether writing has been done to the table.

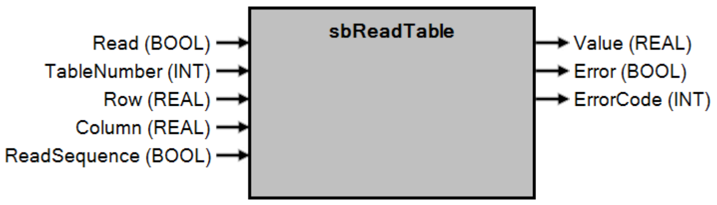
***Error* (BOOL):**
This output indicates whether an error occurred during block processing.

***ErrorCode* (INT):**
This output indicates the type of the error if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.6.1.4 Block Call

sbWriteTable(Write, TableNumber, Row, Column, Value)

6.6.2 Read Table Field - ReadTable



6.6.2.1 Operation

The function of this block is to read data from CNC tables.
To read a value in a table cell, you must enter in the block entries the table number, the column and the row where you want to read the value.

6.6.2.2 Entries

***Read* (BOOL):**
This input performs the read on the table when it receives a rising transition.
If you want to read the value from another position in the table or from another table, you must supply another rising edge in this argument.
If you want to read sequential values from the same table, you have the option to keep the "Read" and "ReadSequence" values at TRUE (see the explanation of the "ReadSequence" entry).

***TableNumber* (INT):**
This entry defines the table number from which you want to read the data(s).

***Row* (REAL):**
This entry defines the row number of the table.

***Column* (REAL):**
This entry defines the column number of the table.

***ReadSequence* (BOOL):**
This input defines whether the reading will be done sequentially.
By assigning TRUE to this input and TRUE to the "Read" input, the return of read values becomes incremental.
In the first PLC cycle, the block will return the value of the cell specified in the inputs. In the
Next, the block increments the number of the line to be read from. When the last row is reached, the block increments the column number and repeats the process. When the end of the table is reached, the block returns to the first
value from the table and the process continues until sequential reading is stopped when "ReadSequence" is set to FALSE.

6.6.2.3 Outputs

***Error* (BOOL):**
This output indicates whether an error occurred during block processing.

***ErrorCode* (INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

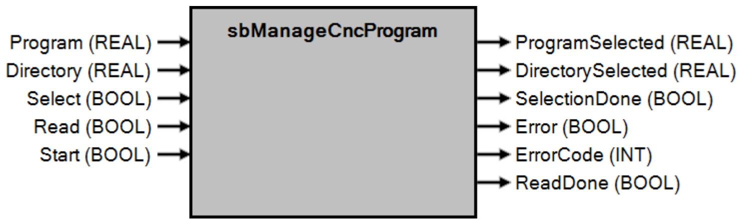
***Value* (REAL):**
This output returns the value read from the indicated cell. The value is updated always after the "Read" undergoes a climb transition or once every PLC cycle if "ReadSequence" is set to "TRUE".

6.6.2.4 Block Call

sbReadTable(Read, TableNumber, Row, Column, ReadSequence)

6.7 CNC Programs

6.7.1 CNC Program Control - sbManageCncProgram



6.7.1.1 **Operation**

This block has the function of selecting a program at the CNC and executing it, or reading which program is currently selected.
To select a new program, you must enter the directory number where the program is located in the "Directory" entry, enter the program number in the "Program" entry, and set the "Select" entry to TRUE.
To return the current program selected at the CNC, you must set the "Read" input to TRUE. The current program number is returned in the output "ProgramSelected" and its directory in the output "DirectorySelected". In this case it is not necessary to enter the values of "Directory" and "Program".
To run the selected program, simply provide a climb transition at the "Start" input.

6.7.1.2 **Entries**

***Select* (BOOL):**
This entry selects a program at the CNC.
***Read* (BOOL):**
This entry prompts you to read the program that is selected at the CNC.
***Program* (REAL):**
This entry defines the program number that you want to select at the CNC.
***Directory* (REAL):**
This entry defines the number of the directory from which the program is to be selected. If you want to select a program that is in the root of the CNC, place 0 in this parameter.
***Start* (BOOL):**
This input starts the execution of the selected program at the CNC, upon receiving a climb transition.

6.7.1.3 **Outputs**

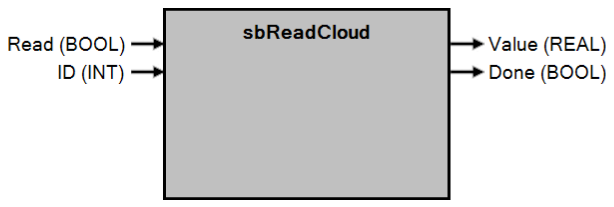
***ProgramSelected* (REAL):**
This output returns the program number selected by the CNC in the case of a read request.
***DirectorySelected* (REAL):**
This output returns the directory number selected by the CNC in the case of a read request.
***SelectionDone* (BOOL):**
This output indicates whether the program has been selected.
***Error* (BOOL):**
This output indicates whether an error occurred during block processing.
***ErrorCode* (INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.
***ReadDone* (BOOL):**
This output indicates whether the reading has been completed.

6.7.1.4 **Block Call**

sbManageCNCProgram(Program, Directory, Select, Read, Start);

6.8 Cloud

6.8.1 Read Data from the Cloud - sbReadCloud



6.8.1.1 Operation

This block has the function of reading data that is available in the CNC Cloud service. The information can be read by passing the corresponding ID into the "ID" input and setting the "Read" input to TRUE. When the "Read" input is set to TRUE, the block reads the corresponding variable and provides the value
on the "Value" output.

6.8.1.2 Entries

***Read* (BOOL):**

This input performs the reading of the variable.

***ID* (INT):**

This input defines the type of data you want to read. Possible values are:

0 - Machine state (0 - Idle, 1 - Preparing, 2 - Producing, 3 - Emergency, 4 - Maintenance)
1 - Total pieces/sockets scheduled
2 - Total parts/sockets produced
3 - Order Number
4 - Time spent per unit
5 - Selected program
6 - Directory of the selected program (0 - Programs, 1 - SUB, 2 - CYC)
7 - Performance - % exec
8 - Performance - % plc
9 - Performance - % motion
10 - Performance - % ethercat
11 - Power (Consumption)

6.8.1.3 Outputs

***Done* (BOOL):**

This output indicates whether the block has been executed and the value is available at the "Value" output.

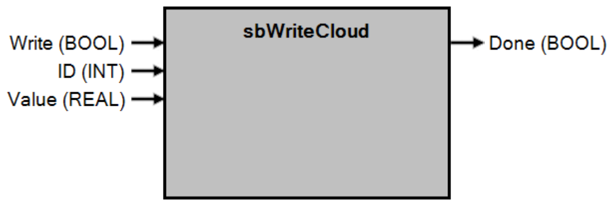
***Value* (REAL):**

This output returns the read value, corresponding to the data type defined in the "ID" input.

6.8.1.4 Block Call

sbReadCloud(Read, ID)

6.8.2 Writing Data in the Cloud - sbWriteCloud



6.8.2.1 Operation

This block is used to write data to the variables available in the CNC Cloud service. Information can be written by passing the corresponding ID into the "ID" input, the value into the "Value" input and setting the "Write" input to TRUE.

6.8.2.2 Entries

***Write* (BOOL):**

This input performs the writing of the value to the variable.

***ID* (INT):**

This input defines the type of data you want to write. Possible values are:

0 - Machine state (0 - Idle, 1 - Preparing, 2 - Producing, 3 - Emergency, 4 - Maintenance)
1 - Total pieces/sockets scheduled
2 - Total parts/sockets produced
3 - Order Number
4 - Time spent per unit
5 - Selected program
6 - Directory of the selected program (0 - Programs, 1 - SUB, 2 - CYC)
7 - Performance - % exec
8 - Performance - % plc
9 - Performance - % motion
10 - Performance - % ethercat
11 - Power (Consumption)

***Value* (Real):**

This entry sets the value that you want to write to the variable specified in the "ID" entry.

6.8.2.3 Outputs

***Done* (BOOL):**

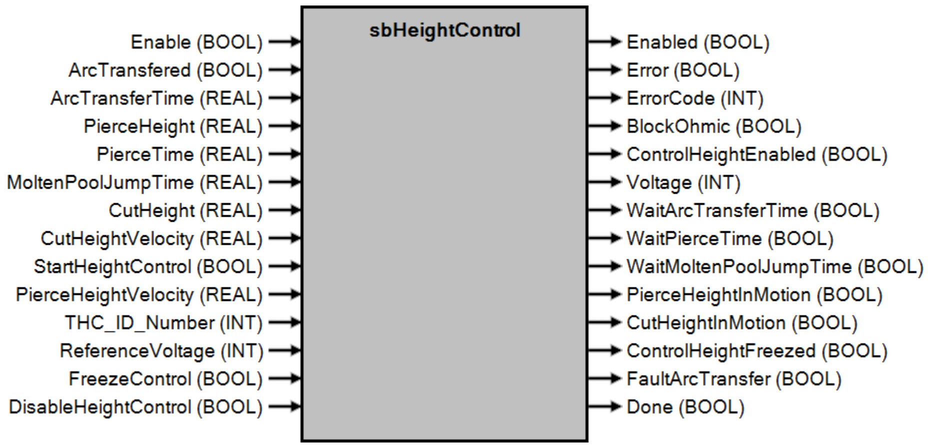
This output indicates whether the block has been executed.

6.8.2.4 Block Call

sbWriteCloud(Write, ID, Value)

6.9 THC

6.9.1 Height Control - sbHeightControl



6.9.1.1 Operation

This block has the function of performing the THC control used in plasma cutting machines, where an axis perpendicular to the cutting plane adjusts its height depending on the ripples of the plate. This correction is done by an analog feedback, that is, an analog reference corresponding to the arc voltage is passed to the CNC so that the height correction is executed.
Once the block is enabled, you are prompted to start the height control. The process consists of:
• Waiting for the time for arc transfer, where the ArcTransfered information is expected to return after ignition of the plasma source;
• Position the shaft at the speed-controlled pierce height (PierceHeight, PierceHeightVelocity);
• Count Pierce Time (PierceTime);
• Count dribble jump time (MoltenPoolJumpTime);
• Position at the speed-controlled cutting height (CutHeight, CutHeightVelocity);
• Once positioned at the cutting height, start the height control.

Machine Parameters

There are some machine parameters that need to be programmed for this block to function correctly. These parameters must be programmed in the "PLC" tab of the CNC parameters screen. In this case, the user must add the parameters in the application being developed.
P960 = Axis number associated with THC1
P961 = Axis number associated with THC2
P962 = Axis number associated with THC3
P963 = Axis number associated with THC4
P964 = Analog input associated with THC1 height control analog feedback P965 = Analog input associated with THC2 height control analog feedback P966 = Analog input associated with THC3 height control analog feedback P967 = Analog input associated with THC4 height control analog feedback
P975 = reaction window: Voltage range around the control voltage within which the height control is automatically frozen. The center of the control range is the control voltage.
P976 = control window: Voltage range in which the THC is effectively controlled. Outside this window, the THC height control is frozen. The control window must be larger than the reaction window.
P977 = Proportional gain for height control loop (analog feedback). P978 = Integral gain for height control loop (analog feedback).
P979 = Differential gain for height control loop (analog feedback).
P981 = Full scale voltage, i.e. voltage at the torch corresponding to 10 volts on the analog input of the THC.
P984 = Voltage percentage to detect sudden changes in the arc voltage. If there is a sudden increase in voltage equivalent to the value of this parameter, the THC height control is frozen.

6.9.1.2 Entries

Enable **(BOOL):**
This input enables the block. If this input is FALSE, the block is disabled and all outputs get null values.
StartHeightControl **(BOOL):**
This input initializes the height control enable process upon receiving a climb transition.
ArcTransferTime **(REAL):**
This entry defines the time in seconds to wait for arc transfer (ArcTransfered). If the arc is not transferred until this time has elapsed, the process is aborted and an error is indicated (FaultArcTransfer).
ArcTransfered **(BOOL):**
This input receives the expected signal during the arc transfer time (ArcTransferTime). During this time, this signal is expected to go to logic state TRUE.
PierceHeight **(REAL):**
This entry defines the absolute position where the THC will be positioned to start drilling the plate.
PierceHeightVelocity **(REAL):**
This input sets the speed at which the CNC positions the THC at the pierce position.
PierceTime **(REAL):**
This input defines the time the shaft is stopped at the pierce position before going to the next step of the process.
MoltenPoolJumpTime **(REAL):**
This entry sets the time for lees skipping. The height control stays at the pierce height for the time set in this argument.
CutHeight **(REAL):**
This input defines the absolute position where the THC will be positioned to start cutting.
CutHeightVelocity **(REAL):**
This input sets the speed at which the CNC positions the THC to the cutting position.
DisableHeightControl **(BOOL):**
This input disables height control.
At the climb transition the block is asked to disable the height control, i.e. the CNC stops looking
at analog feedback to correct the cut height.
FreezeControl **(BOOL):**
This input prompts the block to freeze the current THC height, i.e., on the up transition the block maintains
o THC at the height it was at before this function was requested. This function is only activated if height control is enabled.
ReferenceVoltage **(INT):**
This input sets the voltage for the programmed cut height (CutHeight). When the CNC positions
o THC at cut height, the arc voltage is expected to match the value of this argument. When positioning at cut height the height control expects the arc voltage to assume this value.
THC_ID_Number **(INT):**
This input defines the number of the THC that you want to enable. The CNC is set up to control up to 4 height control units, i.e. we can program values from 1 to 4.

6.9.1.3 Outputs

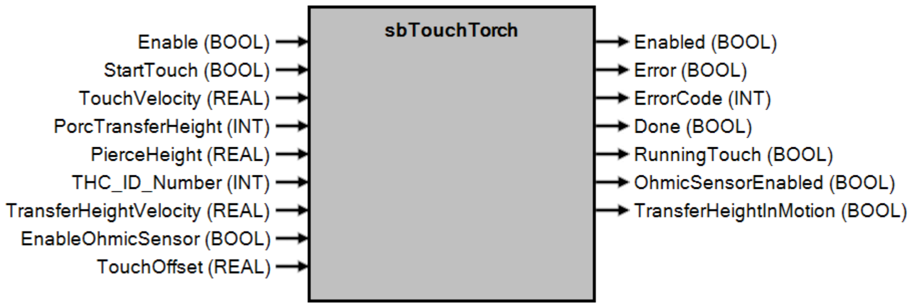
Enabled **(BOOL):**
This output indicates whether the block is enabled.
BlockOhmic**:**
This output indicates whether the ohmic touch is blocked.
ControlHeightEnabled **(BOOL):**
This output indicates whether the process to enable control has been completed and is operating.
ControlHeightFreezed **(BOOL):**
This output indicates whether the height control is frozen.
CutHeightInMotion **(BOOL):**
This output indicates whether the CNC is positioning the THC at the cut height.

***PierceHeightInMotion* (BOOL):**
This output indicates whether the CNC is positioning the THC at the pierce height.
***Voltage* (INT):**
This output indicates the voltage received by the analog feedback.
***WaitArcTransferTime* (BOOL):**
This output indicates whether the arc transfer time has elapsed.
***WaitMoltenPoolJumpTime* (BOOL):**
This output indicates whether the time of the lees skipping process has passed.
***WaitPierceTime* (BOOL):**
This output indicates whether the time of the drilling process has elapsed.
***FaultArcTransfer* (BOOL):**
This output indicates whether there has been a failure while waiting for the time to transfer arc.
***Error* (BOOL):**
This output indicates whether an error occurred during block processing.
***ErrorCode* (INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.9.1.4 Block Call

sbHeightControl(Enable, ArcTransfered, ArcTransferTime, PierceHeight, PierceTime, MoltenPoolJumpTime, CutHeight, CutHeightVelocity, StartHeightControl, PierceHeightVelocity, THC_ID_Number, ReferenceVoltage, FreezeControl, DisableHeightControl)

6.9.2 Ohmic Touch - sbTouchTorch



6.9.2.1 Operation

This block has the function of performing plate detection and presetting to zero at the detected coordinate.
The process consists of moving the axis in the negative direction until the plate is found. After it is found, the coordinate is preset to zero and then the axis is positioned at a transfer height.
There are two ways to detect the plate: by ohmic sensor or by torque.
When ohmic sensor detection is used, it is preset to zero as soon as the sensor is triggered.
The ohmic sensor is set in parameters P968 (for THC 1), P969 (for THC 2), P970 (for THC 3) and P971 (for THC 4).
When torque detection is used, when the plate is detected, the position is added to an offset (set at the "TouchOffset" input) and then preset to zero.
The torque detection process is done by detecting a delay (lag - P974). When the delay in the plate capture process is greater than the one defined in the parameter, it is understood that the plate has been found.
Torque detection is always enabled, while ohmic sensor detection can be disabled by setting the "EnableOhmicSensor" input to FALSE. This block is responsible for performing plate detection and presetting to zero at the detected coordinate.
The process consists of moving the axis in the negative direction until the plate is found. After it is found, the coordinate is preset to zero and then the axis is positioned at a transfer height.
There are two ways to detect the plate: by ohmic sensor or by torque.
When ohmic sensor detection is used, it is preset to zero as soon as the sensor is triggered.
The ohmic sensor is set in parameters P968 (for THC 1), P969 (for THC 2), P970 (for THC 3) and P971 (for THC 4).
When torque detection is used, when the plate is detected, the position is added to an offset (set at the "TouchOffset" input and then preset to zero.
The torque detection process is done by detecting a delay (lag - P974). When the delay in the plate capture process is greater than the one defined in the parameter, it is understood that the plate has been found.
Torque detection is always enabled, while ohmic sensor detection can be disabled by setting the "EnableOhmicSensor" input to FALSE.

Machine Parameters

P968 = Digital input associated with the ohmic sensor of THC1 P969 = Digital input associated with the ohmic sensor of THC2 P970 = Digital input associated with the ohmic sensor of THC3 P971 = Digital input associated with the ohmic sensor of THC4 P974 = Lag for torque touch detection.

6.9.2.2 Entries

***Enable* (BOOL):**
This input enables the block. If this input is FALSE, the block is disabled and all outputs get null values.
***EnableOhmicSensor* (BOOL):**
This input enables plate detection by ohmic sensor.

PierceHeight **(REAL):**
This entry defines the absolute position where the THC will be positioned to start drilling the plate.
PorcTransferHeight **(INT):**
This entry defines the percentage of transfer height. This argument indicates the percentage of the drilling height at which to position the shaft.
Example: if PorcTransferHeight is 50 and PierceHeight is 10, the transfer height will be 50% of 10, i.e. 5.
StartTouch **(BOOL):**
This input initializes the touch process that does the plate capture, upon receiving a rise transition.
TouchOffset **(REAL):**
This input defines the value to be added to the shaft position at the time torque detection occurs.
TouchVelocity **(INT):**
This input sets the axis speed during plate capture.
TransferHeightVelocity **(INT):**
This input defines the speed at which the CNC positions the axis to the transfer position.
THC_ID_Number **(INT):**
This input defines the number of the THC you want to enable, there are 4 possible THCs (from 1 to 4).

6.9.2.3 Outputs

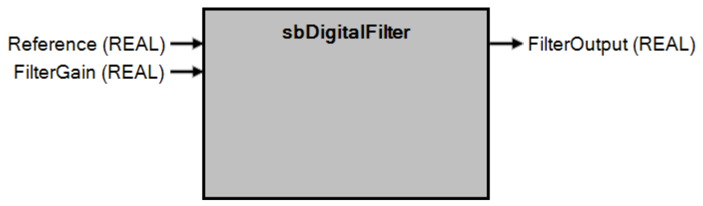
Enabled **(BOOL):**
This output indicates whether the block is enabled and ready for use.
Done **(BOOL):**
This output indicates whether the processing of the block has been completed.
OhmicSensorEnabled **(BOOL):**
This output indicates whether ohmic sensor detection is enabled.
Capture should be disabled whenever a situation arises where the sensor may be triggered incorrectly,
such as when cutting under water.
RunningTouch **(BOOL):**
This output indicates if the plate capture process is in progress (movement in the negative direction), but the plate has not yet been detected.
TransferHeightInMotion **(BOOL):**
This output indicates whether the CNC is positioning the axis at the transfer position.
Error **(BOOL):**
This output indicates whether an error occurred during block processing.
ErrorCode **(INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.9.2.4 Block Call

sbTouchTorch(Enable, StartTouch, TouchVelocity, PorcTransferHeight, PierceHeight, THC_ID_Number, TransferHeightVelocity, EnableOhmicSensor, TouchOffset)

6.10 Control

6.10.1 Digital Filter - sbDigital Filter



6.10.1.1 Operation
This block has the function of filtering an analog signal based on the formula:
In any control system with an analog feedback signal present, there is a risk of noise that can compromise the integrity of the signal, producing a less robust system.
This block is a first order digital filter. It filters the signal provided at the "Reference" input, and the filter strength is selected at the "FilterGain" input.

6.10.1.2 Entries

***Reference* (REAL):**
This input receives the signal that will be filtered.

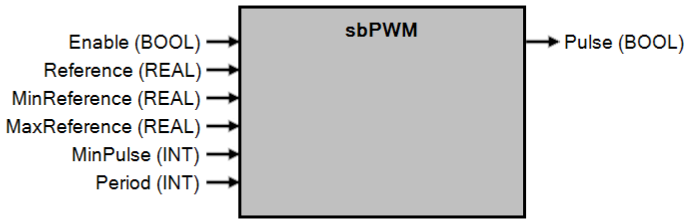
***FilterGain* (REAL):**
This input sets the gain used to attenuate the oscillations of the reference signal. It can be a value from 0.05 to 1.00. The smaller the value, the greater the filter performance. If this value is 1, the filter is disabled and the input value is passed to the output.

6.10.1.3 Outputs

***FilterOutput* (REAL):**
This output is the filtered signal.

6.10.1.4 Block Call
sbDigitalFilter(Reference, FilterGain)

6.10.2 PWM - sbPWM



6.10.2.1 Operation

This block has the function of generating a PWM signal.
Pulse-Width Modulation (PWM) is a modulation technique used to encode a message into a pulsed signal. Although this technique
of modulation can be used to encode information for transmission, its main use is to allow control of the power supplied to electrical devices, especially inertial loads such as motors and heating elements.
The pulse width is calculated by the formula:

$$\text{TimeON} = \textbf{Period x Duty Cycle}$$

The term Duty Cycle describes the proportion of time "on" in relation to the regular interval or "period" of time: a low duty cycle corresponds to low power, because the power supply is off most of the time. The duty cycle is expressed as a percentage, with 100% being fully on. The duty cycle can be calculated by:

$$\textbf{Duty cycle} = (\textbf{Reference} - \text{ReferenceMIN}) / (\text{ReferenceMAX} - \text{ReferenceMIN})$$

If the calculated time for the (on) pulse is less than the value set in the "MinPulse" input, the calculated value will be replaced by the time set in "MinPulse" and the complementary (off) time will be calculated by:

$$\text{TimeOFF} = (\text{Calculated MinPulse/TimeON}) \text{ x } (\textbf{Period} - \text{Calculated TimeON})$$

6.10.2.2 Entries

***Enable* (BOOL):**
This input enables the modulation calculation. Keep active as long as the calculation is used.
***Reference* (REAL):**
This input receives the signal that together with the maximum and minimum inputs define the pulse width.
***MaxReference* (REAL):**
This input defines the maximum value that the reference input can reach.
***MinReference* (REAL):**
This input defines the minimum value that the reference input can reach.
***MinPulse* (INT):**
This entry sets the minimum allowed time, in milliseconds, for on time.
***Period* (INT):**
This input sets the signal period, in milliseconds.

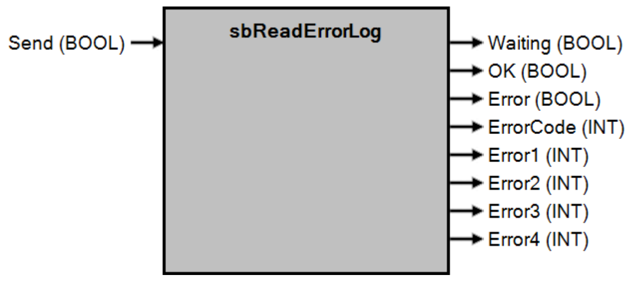
6.10.2.3 Outputs

***Pulse* (BOOL):**
This output is PWM with a cycle based on the input arguments.

6.10.2.4 Block Call

sbPWM(Enable, Reference, MinReference, MaxReference, MinPulse, Period)

6.10.1 Read Logged Errors - sbReadErrorLog



6.10.1.1 Entries

The function of this block is to return the first 4 errors in the CNC error list.

6.10.1.1 Entries

Send(bool)

***Send* (BOOL):**

This entry fetches the first 4 errors in the error list and sends them to the respective outputs of the block.

6.10.1.2 Outputs

***Waiting* (BOOL):**

This output indicates whether the CNC is waiting for errors to be read.

***OK* (BOOL):**

This output indicates whether the error reading has been done.

***Error* (BOOL):**

This output indicates whether an error occurred during block processing.

***ErrorCode* (INT):**

This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

***Error1* (INT):**

This output indicates the first error in the CNC error list.

***Error2* (INT):**

This output indicates the second error in the CNC error list.

***Error3* (INT):**

This output indicates the third error in the CNC error list.

***Error4* (INT):**

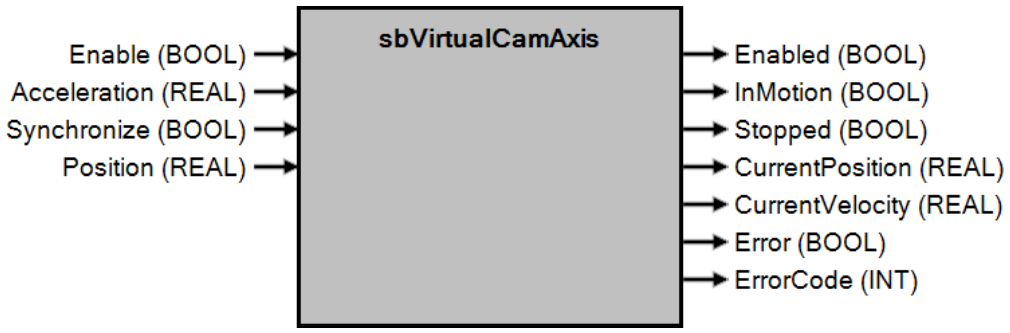
This output indicates the fourth error in the CNC error list.

6.10.1.3 Block Call:

sbReadErrorLog(Send)

6.11 Came

6.11.1 Virtual Cam - sbVirtualCamAxis



6.11.1.1 Operation

This block has the function of creating a virtual cam, that is, a virtual axis where we can associate functions, such as drives of digital outputs by angle (digital cam), links to other axes so that in an angular range these other axes make a movement defined in a table (axis cam), etc.
It is only possible to have one instance of this block, that is, it is not possible to have more than one virtual cam.

6.11.1.2 Entries

***Enable* (BOOL):**
This input enables the block. If the value is FALSE, the block is disabled and all outputs of the block receive null values.
***Acceleration* (INT):**
This input defines the acceleration and deceleration time of the movements requested from the cam in milliseconds.
***Sync* (BOOL):**
This input requests to synchronize the virtual cam with the actual cam, the synchronization is started when the actual cam passes the position set in "Position". After this position the virtual and actual cams will be synchronized.
***Position* (REAL):**
This entry defines the absolute position where the actual cam will be synchronized.

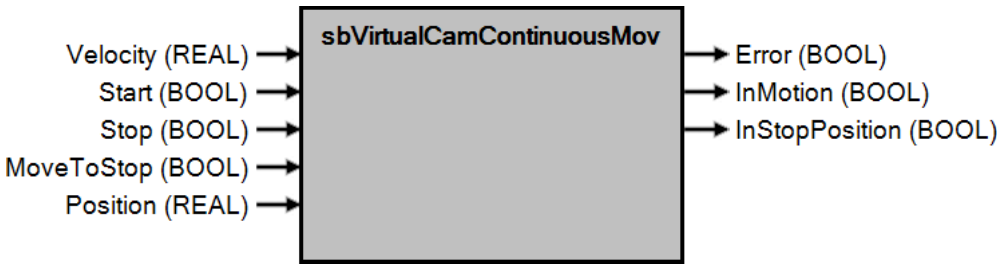
6.11.1.3 Outputs:

***Enabled* (BOOL):**
This output indicates whether the block is enabled.
***InMotion* (BOOL):**
This output indicates whether the virtual cam is moving.
***Stopped* (BOOL):**
This output indicates whether the movement being executed by the virtual cam has been stopped.
***CurrentPosition* (BOOL):**
This output indicates the current position of the cam.
***CurrentVelocity* (BOOL):**
This output indicates the current speed of the cam.
***Error* (BOOL):**
This output indicates whether one occurred during the processing of the block.
***ErrorCode* (INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.11.1.4 Block Call:

sbVirtualCamAxis(Enable, Acceleration, Synchronize, Position)

6.11.2 Continuous Virtual Cam - sbVirtualCamContinuousMove



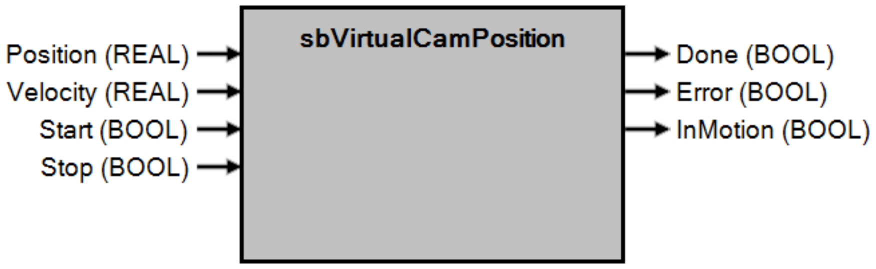
6.11.2.1 Operation
This block has the function of executing continuous movements on the virtual cam with speed defined in "Velocity".

6.11.2.2 Entries
***Velocity* (INT):**
This input defines the speed at which the requested cam movements (RPM) will be executed.
***Start* (BOOL):**
This entry requests to start the continuous movement of the virtual cam.
***Stop* (BOOL):**
This input prompts the cam to stop immediately, the cam will stop at the position it is in at the time of the prompt.
***MoveToStop* (BOOL):**
This entry requests a stop at the position set in "Position".
***Position* (REAL):**
This input defines the absolute position where the virtual cam will be positioned when a "MoveToStop" is requested.

6.11.2.3 Outputs:
***Error* (BOOL):**
This output indicates whether an error occurred during the block processing. Verify that there is an enabled instance of the "VirtualCamAxis" block and that the inputs have valid values: the "Position" input between 0 and 360 and the "Velocity" input different from 0.
If none of the above are invalid, disable the block and enable it again.
***InMotion* (BOOL):**
This output indicates whether the virtual cam is executing continuous movement or movement to the stop angle.
***InStopPosition* (BOOL):**
This output indicates whether the virtual cam is stopped.

6.11.2.4 Block Call
sbVirtualCamContinuousMove(Velocity, Start, Stop, MoveToStop, Position)

6.11.3 Came By Position - sbVirtualCamPosition



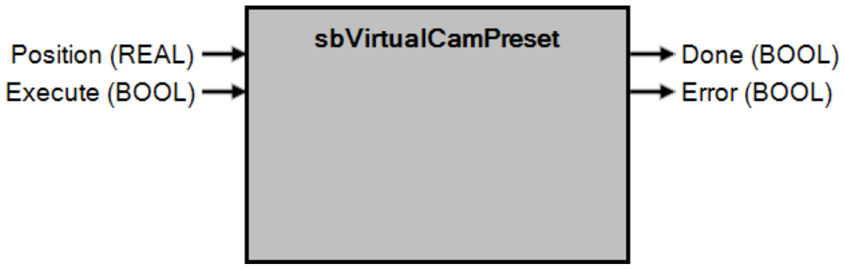
6.11.3.1 Operation
This block has the function of controlling absolute positioning in the virtual cam, it positions the cam to the position entered in the "Position" input and at the speed programmed in the "Velocity" input.

6.11.3.2 Entries
***Position* (REAL):**
This entry defines the absolute position where the cam will be positioned.
***Velocity* (INT):**
This input defines the speed at which the motion will be executed (RPM).
***Start* (BOOL):**
This entry prompts you to start the movement to the position indicated in "Position".
***Stop* (BOOL):**
This input prompts the cam to stop immediately. The cam will stop at the position it is in at the time of the request.

6.11.3.3 Outputs:
***Done* (BOOL):**
This output indicates whether movement has been completed.
***Error* (BOOL):**
This output indicates whether an error occurred during block processing.
Check that there is an enabled instance of the "VirtualCamAxis" block and that the inputs have valid values: the "Position" input between 0 and 360 and the "Velocity" input different from 0.
If none of the above are invalid, disable the block and enable it again.
***InMotion* (BOOL):**
This output indicates whether the virtual cam is performing positioning.

6.11.3.4 Block Call
sbVirtualCamPosition(Position, Velocity, Start, Stop)

6.11.4 Virtual Cam Preset - sbVirtualCamPreset



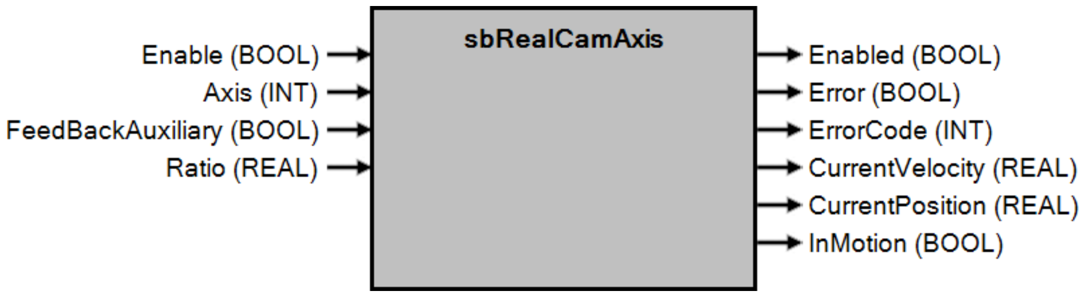
6.11.4.1 Operation
This block has the function of executing a preset on the virtual cam with the value that is at the "Position" input.

6.11.4.2 Position
(REAL) entries:
This input defines the position the cam will assume when the block is executed.
Execute (BOOL):
This input requests that the cam assume the position set on the "Position" input.

6.11.4.3 Outputs:
Done (BOOL):
This output indicates whether the movement has been completed.
Error (BOOL):
This output indicates whether one has occurred during block processing. Verify that there is an enabled instance of the "VirtualCamAxis" block and that the axis is stopped when requesting the preset. If the axis is moving, the block generates a failure.
If none of the above are invalid, disable the block and enable it again.

6.11.4.4 Block Call
sbVirtualCamPreset(Position, Execute)

6.11.5 Real Axis CamAxis - sbRealCamAxis



6.11.5.1 Operation

This block has the function of creating a real cam. It is similar to a virtual cam, except that in this case the shaft follows the movements of the encoder or shaft to which it is attached.

6.11.5.2 Entries

***Enable* (BOOL):**

This entry enables the block.

If the value is FALSE, the block is disabled and all outputs from the block are given null values.

***Axis* (INT):**

This input defines which axis will be the master. If you are using an encoder you must inform which axis (AKD) it is connected to.

***Ratio* (REAL):**

This input sets the coupling factor between slave and master. It can be programmed 1 for a 1:1 ratio, 2 so that the slave axis makes 2 turns to 1 turn of the master, or 0.5 so that the master axis makes 2 turns to 1 turn of the slave. This factor is variable and you can set the appropriate factor for your application.

***FeedBackAuxiliary* (BOOL):**

If set to FALSE, the feedback used will be an auxiliary encoder. If set to TRUE, it will use the axis feedback defined at the "Axis" input.

6.11.5.3 Outputs:

***Enabled* (BOOL):**

This output indicates that the block is enabled and ready for use.

***CurrentPosition* (BOOL):**

This output indicates the current position of the cam.

***CurrentVelocity* (BOOL):**

This output indicates the current speed of the cam.

***Error* (BOOL):**

This output indicates whether an error occurred during block processing.

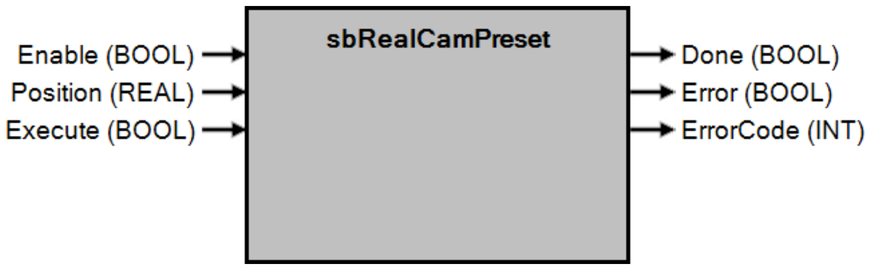
***ErrorCode* (INT):**

This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.11.5.4 Block Call

```
sbRealCam(Enable, Axis, FeedBackAuxiliary, Ratio);
```

6.11.6 Real Axis Cam Preset - sbRealCamPreset



6.11.6.1 Operation
This block has the function of presetting the actual cam with the value that is at the Position input.

6.11.6.2 Entries
***Enable* (BOOL):**
This entry enables the block.
If the value is FALSE, the block is disabled and all outputs from the block are given null values.

***Position* (REAL):**
This input defines the position the cam will assume when the block is executed.

***Execute* (BOOL):**
This input requests that the cam assume the position set on the "Position" input.

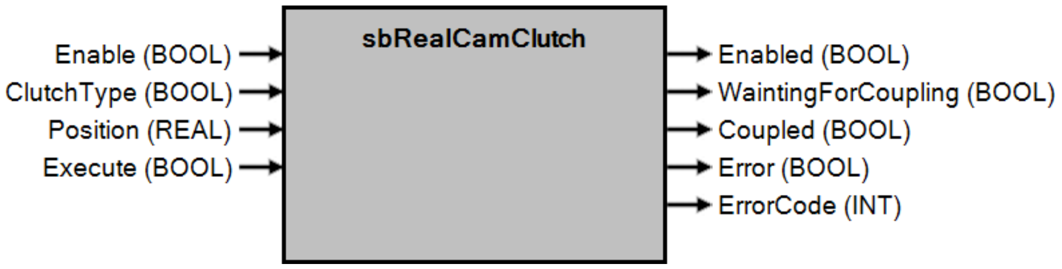
6.11.6.3 Outputs:
***Done* (BOOL):**
This output indicates whether movement has been completed.

***Error* (BOOL):**
This output indicates whether an error occurred during block processing.

***ErrorCode* (INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.11.6.4 Block Call
sbRealCamPreset(Enabled, Position, Execute);

6.11.7 Real Cam Coupling - sbRealCamClutch



6.11.7.1 Operation

This block has the function of making the coupling with the real cam. In it we define whether a coupling occurs immediately or after a certain position. If we use the form of coupling after a certain position, we must define it in the "Position" input. After setting the shaft number, the type of coupling and if there is enabled
the block, you must set the "Execute" input to TRUE.

6.11.7.2 Entries

***Enable* (BOOL):**
This entry enables the block.
If the value is FALSE, the block is disabled and all outputs from the block are given null values.
***ClutchType* (BOOL):**
This input defines the type of coupling, where:
0: couples immediately;
1: Engages after passing through the position determined at the "Position" input.
***Position* (REAL):**
This input sets the position used to engage the cam when the "ClutchType" input is set to TRUE.
***Execute* (BOOL):**
This input requests the coupling of the cam.

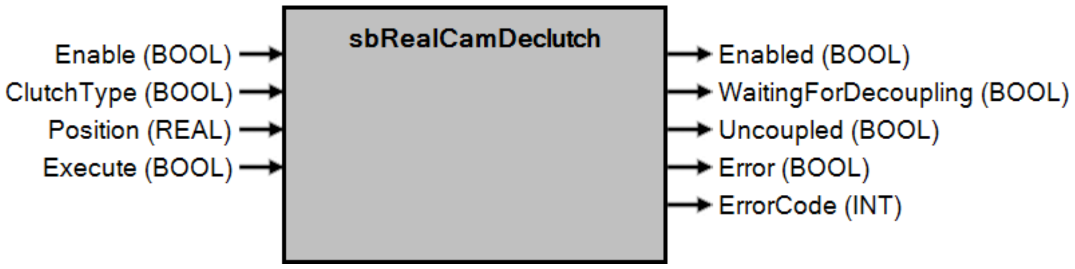
6.11.7.3 Outputs:

***Enabled* (BOOL):**
This output indicates that the block is enabled.
***Coupled* (BOOL):**
This output indicates that the cam is engaged.
***Error* (BOOL):**
This output indicates that an error has occurred while processing the block.
***ErrorCode* (INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.
***WaitingForCoupling* (BOOL):**
This output indicates whether the cam is waiting for the programmed position on the input
"Position" to be able to couple. When coupling is performed, the "Coupled" output is set to TRUE.

6.11.7.4 Block Call

sbRealCamClutch (Enable, ClutchType, Position, Execute)

6.11.8 Real Cam Decoupling - sbRealCamDeclutch



6.11.8.1 Operation

This block has the function of decoupling the real cam. In it we define whether an immediate coupling occurs or after a certain position. If we use the form of uncoupling after a certain position, we must set it on the "Position" input. After setting the axis number, the decoupling type and if you have enabled the block, you must set the "Execute" input to TRUE.

6.11.8.2 Entries

***Enable* (BOOL):**
This entry enables the block.
If the value is FALSE, the block is disabled and all outputs from the block are given null values.

***ClutchType* (BOOL):**
This input defines the type of decoupling.
If this input is set to FALSE, immediate decoupling will take place. If it is set to TRUE, the decoupling will be done after passing through the position programmed on the Position input.

***Position* (REAL):**
This input sets the position used to uncouple the cam when the "ClutchType" input is set to TRUE.

***Execute* (BOOL):**
This input prompts the cam to decouple.

6.11.8.3 Outputs:

***Enabled* (BOOL):**
This output indicates whether the block is enabled.

***Uncoupled* (BOOL):**
This output indicates whether cam is uncoupled.

***Error* (BOOL):**
This output indicates whether an error occurred during block processing.

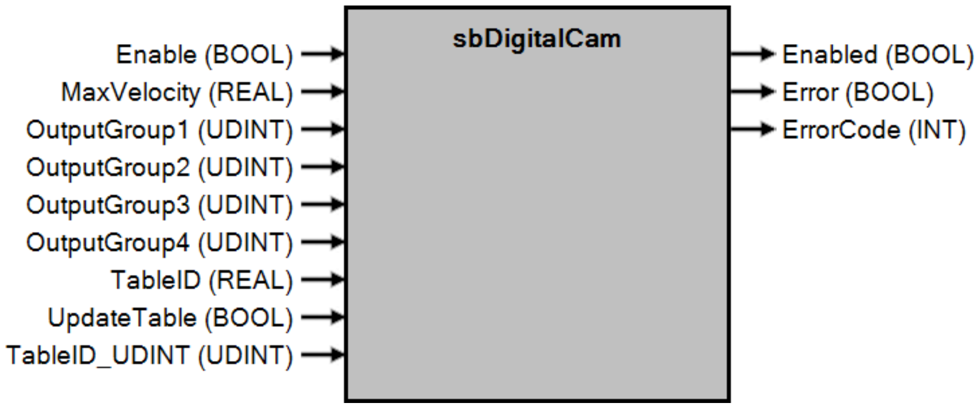
***ErrorCode* (INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

***WaitingForDecoupling* (BOOL):**
This output indicates whether the cam is waiting for the programmed position on the input "Position" to be able to decouple. When decoupling is performed, the "Uncoupled" output becomes TRUE.

6.11.8.4 Block Call

sbRealCamDeclutch(Enable, ClutchType, Position, Execute);

6.11.9 Came Digital - sbDigitalCam



6.11.9.1 Operation

This block has the function of creating a digital cam with a structure of up to 32 digital outputs.
Using a table, configured by the user, it is possible to drive these outputs by angles or by time. In the table editor of the AKC-PPC IDE, a table of 32 rows and 5 columns should be created, where:
-Column 1 refers to the starting angle in degrees
-column 2 is the final angle in degrees or time in milliseconds
-Column 3 is the Shift (cam displacement as a function of speed) in degrees
-column 4 is the master cam type (0=Real, 1=Virtual)
-Column 5 is for selecting the cam by time (0=Degrees, 1=Time)

6.11.9.2 Entries

***Enable* (BOOL):**
This entry enables the block.
If the value is FALSE, the block is disabled and all outputs from the block are given null values.

***MaxVelocity* (REAL):**
This input sets the speed of the real (encoder, resolver, etc.) or virtual cam - it depends on which cam is linked to the digital cam.

***OutputGroup1* (UDINT):**
This input defines the digital output group number that will be associated with cam outputs 0 to 7.

***OutputGroup2* (UDINT):**
This input defines the digital output group number that will be associated with cam outputs 8 to 15.

***OutputGroup3* (UDINT):**
This input defines the digital output group number that will be associated with cam outputs 16 to 23.

***OutputGroup4* (UDINT):**
This input defines the digital output group number that will be associated with cam outputs 24 to 31.

***TableID* (REAL):**
This entry sets the digital cam configuration table number.

***UpdateTable* (BOOL):**
This input sets the bit that will be used to update the digital cam table.

***TableID_UDINT* (UDINT):**
This entry defines table number (must be typed again).

6.11.9.3 Outputs:

***Enabled* (BOOL):**
This output indicates whether the block is enabled.

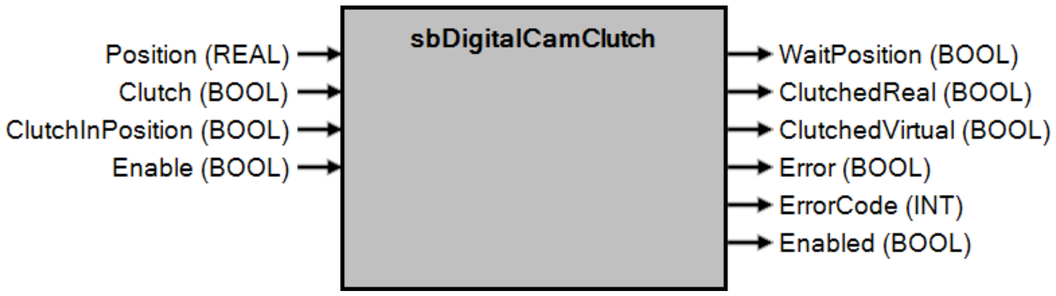
***Error* (BOOL):**
This output indicates whether an error occurred during block processing.

***ErrorCode* (INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.11.9.4 Block Call

sbCameDigital(Enable, MaxVelocity, OutputGroup1, OutputGroup2, OutputGroup3, OutputGroup4, TableID, UpdateTable, TableID_UDINT)

6.11.10 Digital Cam Coupling - sbDigitalCamClutch



6.11.10.1 Operation

This block has the function of performing the coupling of the digital cam created in "sbDigitalCam" to the master cam, either virtual or real.

6.11.10.2 Entries

***Position* (REAL):**

This input sets the position used to couple the cam if the drive on the "ClutchInPosition" input is used. If there is an offset applied to the cam, the position will take this offset into account. ***Clutch* (BOOL):**

This input requests to immediately couple the digital cam to the associated cam (real or virtual), i.e., at the current position, the associated cam will be coupled to the digital cam.

***ClutchInPosition* (BOOL):**

This input requests to engage the digital cam of the associated cam (real or virtual) at the moment the position of the associated cam passes the position set on the "Position" input.

***Enable* (BOOL):**

This entry enables the block.

If the value is FALSE, the block is disabled and all outputs from the block are given null values.

6.11.10.3 Outputs:

***WaitPosition* (BOOL):**

This output indicates whether the CNC is waiting for the actual cam to reach the position defined at the "Position" input to couple.

***ClutchedReal* (BOOL):**

This output indicates whether the digital cam is coupled to the actual cam.

***ClutchedVirtual* (BOOL):**

This output indicates whether the digital cam is coupled to the virtual cam.

***Error* (BOOL):**

This output indicates whether an error occurred during block processing.

***ErrorCode* (INT):**

Indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

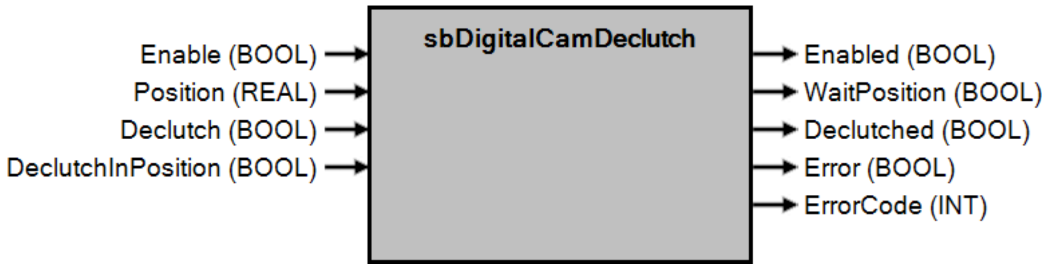
***Enabled* (BOOL):**

This output indicates whether the block is enabled.

6.11.10.4 Block Call

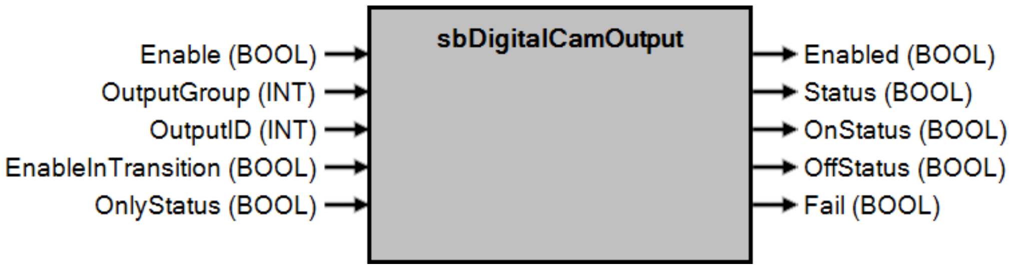
sbDigitalCamClutch(Position, Clutch, ClutchInPosition, Enable)

6.11.11 Digital Cam Decoupling - sbDigitalCamDeclutch



6.11.5.1 Operation
This block has the function of decoupling the digital cam created in "sbDigitalCam" from the master cam, either virtual or real.
6.11.11.2 Entries
Enable **(BOOL):**
This entry enables the block.
If the value is FALSE, the block is disabled and all outputs from the block are given null values.
Position **(REAL):**
This input sets the position used to uncouple the cam if the drive on the "DeclutchInPosition" input is used. If there is an offset applied to the cam, the position will take this offset into account. ***Declutch*** **(BOOL):**
This input requests to immediately decouple the digital cam from the associated cam (real or virtual), i.e., at the current position of the associated cam it will be decoupled from the digital cam.
DeclutchInPosition **(BOOL):**
This input requests to decouple the digital cam from the associated cam (real or virtual) the moment the position of the associated cam passes the position set on the "Position" input.
6.11.11.3 Outputs:
Enabled **(BOOL):**
This output indicates whether the block is enabled.
Declutched **(BOOL):**
This output indicates whether the digital cam is decoupled from the associated cam (real or virtual).
WaitPosition **(BOOL):**
This output indicates whether the CNC is waiting for the actual cam to reach a position defined in "Position" before uncoupling.
Error **(BOOL):**
This output indicates whether an error occurred during block processing.
ErrorCode **(INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.
6.11.11.4 Block Call
sbDigitalCamDeclutch (Enable, Position, Declutch, DeclutchInPosition)

6.11.12 Digital Cam Output - sbDigitalCamOutput



6.11.12.1 Operation

This block has the function of creating a digital cam, that is, through it you can use the states of a digital cam.

6.11.12.2 Entries

***Enable* (BOOL):**
This entry enables the block.
If the value is FALSE, the block is disabled and all outputs from the block are given null values.

***OutputGroup* (INT):**
This input defines the number of the digital output group that will be associated with the outputs that you want to check the states of.

***OutputID* (INT):**
This entry defines the number of the group output defined in "OutputGroup" that you want to check the states of.

***EnableInTransition* (BOOL):**
This entry enables the output on transition.

***OnlyStatus* (BOOL):**
This input configures the block not to change the state of the digital output (if it is set to TRUE).

6.11.12.3 Outputs:

***Enabled* (BOOL):**
This output indicates whether the block is enabled and ready for use.

***OffStatus* (BOOL):**
This output indicates the falling edge state of the digital cam defined at the "OutputGroup" and "OutputID" inputs. It will stay in TRUE for one PLC cycle if the digital cam transitions from TRUE to FALSE.

***OnStatus* (BOOL):**
This output indicates the rising edge state of the digital cam defined at the "OutputGroup" and "OutputID" inputs. It will stay in TRUE for one PLC cycle if the digital cam transitions from FALSE to TRUE.

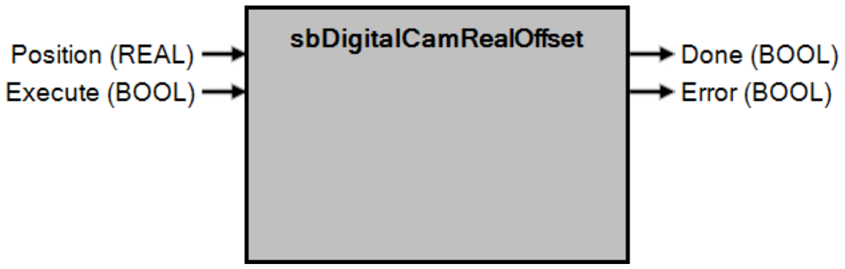
***Status* (BOOL):**
This output indicates the current state of the cam defined in "OutputGroup" and "OutputID".

***Error* (BOOL):**
This output indicates whether an error occurred during block processing. Check whether there is an enabled instance of the "sbDigitalCam" block. If so, the error was generated internally in the CNC.

6.11.12.4 Block Call

sbDigitalCamOutput(Enable, OutputGroup, OutputID, EnableInTransition, OnlyStatus)

6.11.13 Offset Came Digital Real - sbDigitalCamRealOffset



6.11.13.1 Operation

This block has the function of performing an offset between the digital cam and the actual cam, i.e., if the digital cam is coupled to the actual cam and you want to apply a position difference between these cams, you can use this function.

6.11.13.2 Entries

***Position* (REAL):**
This input defines the offset value, in degrees, that you want to apply between the digital cam and the actual cam.

***Execute* (BOOL):**
This input causes the block to apply the offset defined at the "Position" input to the digital cam. To do this perform a climb transition on this input.

6.11.13.3 Outputs:

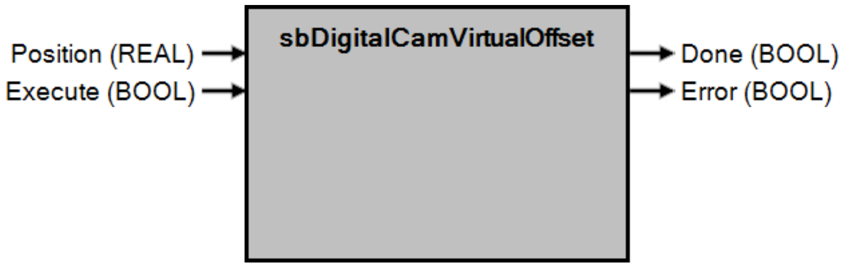
***Done* (BOOL):**
This output indicates that the offset has completed.

***Error* (BOOL):**
This output indicates whether an error occurred during block processing.
Check if there is an enabled instance of the "sbDigitalCam" block or if the digital cam is coupled to the actual cam. If the instance exists and the cams are coupled, the error was generated internally in the CNC.
Disable the block and enable it again.

6.11.13.4 Block Call

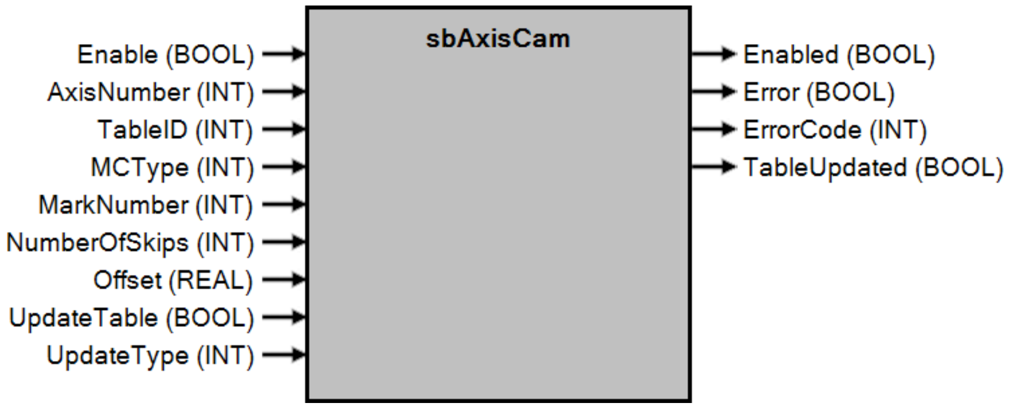
sbDigitalCamRealOffset(Position, Execute)

6.11.14 Offset Came Digital Virtual - sbDigitalCamVirtualOffset



6.11.14.1 Operation
This block has the function of performing an offset between the digital cam and the virtual cam.
6.11.14.2 Entries
Position **(REAL):**
This input defines the offset value, in degrees, that you want to apply between the digital and virtual cam.
Execute **(BOOL):**
This input causes the block to apply the offset defined at the "Position" input to the digital cam.
6.11.14.3 Outputs:
Done **(BOOL):**
This output indicates whether the offset has completed.
Error **(BOOL):**
This output indicates whether an error occurred during block processing. Check if there is an enabled instance of the "sbDigitalCam" block or if the digital cam is coupled to the virtual cam.
If the instance exists and the cams are engaged, the error was generated internally in the CNC. Disable the block and enable it again.
6.11.14.4 Block Call
sbDigitalCamVirtualOffset(Position, Execute)

6.11.15 Shaft Cam - sbAxisCam



6.11.15.1 Operation

This block has the function of creating an axis cam. The main characteristic of this type of cam is to move an axis according to what has been programmed in the corresponding table. This
cam can be coupled with a master cam of either real or virtual type.
In the table editor of the AKC-PPC IDE, an 8 row and 8 column table should be created, where:
-Column 1 refers to the starting angle in degrees
-column 2 is the final angle in degrees or time in milliseconds
-column 3 is the bond length in millimeters
-column 4 refers to the table number with complex profile
-Column 5 is the initial transition zone in percent of the angle
-Column 6 is the final transition zone in percent of the angle
-Column 7 is the initial coupling factor in millimeters/degree
-column 8 is the final coupling factor in millimeters/degree

6.11.15.2 Enable

(BOOL) inputs:
This entry enables the block.
AxisNumber (INT):
This input defines the number of the axis that will be controlled by the cam.
TableID (INT):
This entry defines the table number of the axis cam settings.
MCType (INT):
Sets the type of cam that will be attached (0=real master cam,1=virtual master cam).
MarkNumber (INT):
This input sets the number of the photo input that will be used for cam correction. If set to 0 no photocell will be used.
NumberOfSkips (INT):
This entry sets the number of skips.
Offset (REAL):
This input sets the value for offsetting the axis cam.
UpdateTable (BOOL):
This entry programs the link table.
UpdateType (INT):
This entry programs the auxiliary table with data from the link table

6.11.15.3 Enabl

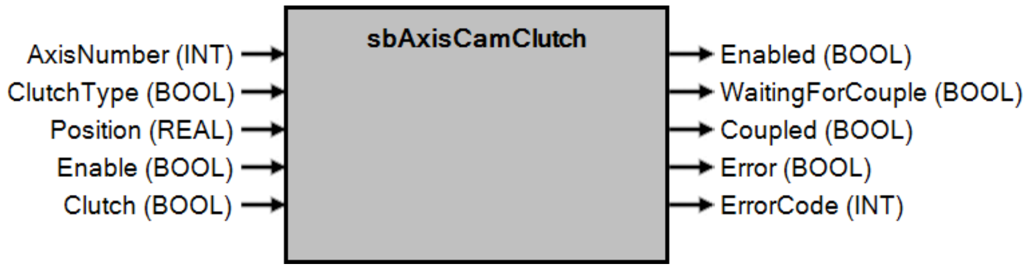
ed outputs
(BOOL):
This output indicates whether block is enabled.
Error (BOOL):
This output indicates whether an error occurred during block processing.

ErrorCode (INT):
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.11.15.4 Block Call

sbAxisCam(Enable, AxisNumber, TableID, MCType, MarkNumber, NumberOfSkips, Offset, UpdateTable, UpdateType)

6.11.16 Shaft Cam Coupling - sbAxisCamClutch



6.11.16.1 Operation

This block has the function of coupling the shaft cam. Here we define whether a coupling occurs immediately or after a certain position. If you use the form of coupling after position, you must define one at the "Position" input. After setting the axis number, the type of coupling and enabling the block, you must set the "Clutch" input to TRUE.

6.11.16.2 Entries

***AxisNumber* (INT):**
This entry defines the number of the axis to be controlled.

***ClutchType* (BOOL):**
This input defines the type of coupling, where: 0:
couples immediately;
1: Engages after passing through the position determined at the "Position" input.

***Position* (REAL):**
This input sets the position used to engage the cam when the "ClutchType" input is set to TRUE.

***Enable* (BOOL):**
This entry enables the block.
If the value is FALSE, the block is disabled and all outputs from the block are given null values.

***Clutch* (BOOL):**
This input requests the coupling of the cam.

6.11.16.3 Outputs

***Enabled* (BOOL):**
This output indicates whether the block is enabled.

***WaitingForCouple* (BOOL):**
This output indicates whether the cam is waiting to pass the programmed position before it can engage.

***Coupled* (BOOL):**
This output indicates whether the axle cam is coupled to the master cam.

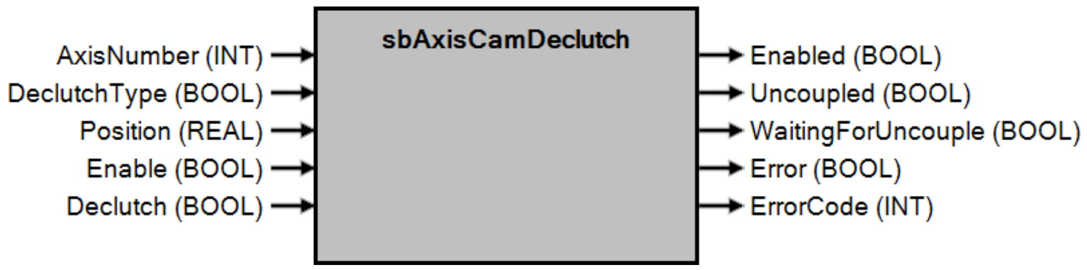
***Error* (BOOL):**
This output indicates whether an error occurred during block processing.

***ErrorCode* (INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.

6.11.16.4 Block Call

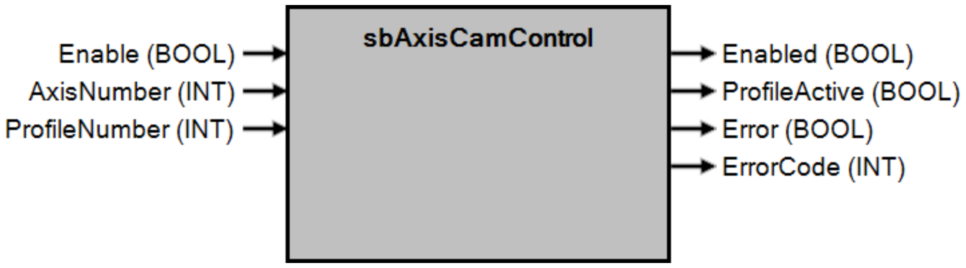
sbAxisCamClutch(AxisNumber, ClutchType, Position, Enable, Clutch)

6.11.17 Shaft Cam Decoupling - sbAxisCamDeclutch



6.11.17.1 Operation
This block has the function of decoupling the axle cam.
6.11.17.2 Entries
***AxisNumber* (INT):**
This entry defines the number of the axis to be controlled.
***DeclutchType* (BOOL):**
This input defines the decoupling type, where: 0:
decouples immediately;
1: Decouples after passing through the position determined at the "Position" input.
***Position* (REAL):**
This input sets the position used to decouple the cam when the "DeclutchType" input is set to TRUE.
***Enable* (BOOL):**
This entry enables the block.
If the value is FALSE, the block is disabled and all outputs from the block are given null values.
***Declutch* (BOOL):**
This input does the decoupling if all the previous inputs are set correctly.
6.11.17.3 Outputs
***Enabled* (BOOL):**
This output indicates whether the block is enabled.
***Uncoupled* (BOOL):**
This output indicates whether the axle cam has been uncoupled from the master cam.
***WaitingForUncouple* (BOOL):**
This output indicates whether the cam is waiting to pass the programmed position before it can uncouple.
***Error* (BOOL):**
This output indicates whether an error occurred during block processing.
***ErrorCode* (INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.
6.11.17.4 Block Call
sbAxisCamDeclutch(AxisNumber, DeclutchType, Position, Enable, Declutch)

6.11.18 Axis Cam Control - sbAxisCamControl



6.11.18.1 Operation
This block has the function of selecting the profile to be executed.

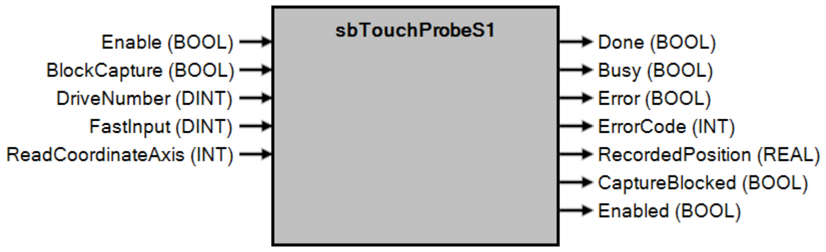
6.11.18.2 Entries
***Enable* (BOOL):**
This entry enables the block, for execution of the programmed links.
If the value is FALSE, the block is disabled and all outputs from the block are given null values.
***AxisNumber* (INT):**
This entry defines the number of the axis on which the link will be executed.
***ProfileNumber* (BOOL):**
This entry defines the profile number (table row).

6.11.18.3 Outputs
***Enabled* (BOOL):**
This output indicates whether the block is enabled.
***Error* (BOOL):**
This output indicates whether an error occurred during block processing.
***ErrorCode* (INT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.
***ProfileActive* (BOOL):**
This output indicates whether the profile is activated.

6.11.18.4 Block Call
sbAxisCamControl(Enable, AxisNumber, ProfileNumber)

6.12 Packaging

6.12.1 Quick Input Reading - TouchProbeS1



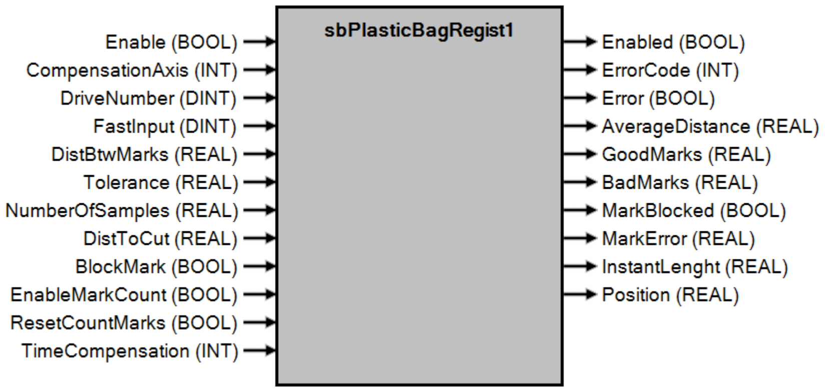
6.12.1.1 Operation
This block has the function of setting a fast digital input of the drive and returning the captured position of the axis when a climb transition occurs on that digital input.
This function block does not cause movement in the axis.

6.12.1.2 Entries
***Enable* (BOOL):**
This entry enables from the block.
***DriveNumber* (SINT):**
This input selects the axis on which the position will be captured.
***FastInput* (DINT):**
This input selects which fast digital input will be used to trigger the capture event.
***BlockCapture* (BOOL):**
This entry blocks the capture event.
***ReadCoordinateAxis* (INT):**
This input defines the number of the axis that the block will capture position.

6.12.1.3 Outputs
***Done* (BOOL):**
This output indicates that the capture was successful.
***CaptudeBlocked* (BOOL):**
This output indicates that the capture is blocked.
***Enabled* (BOOL):**
This output indicates that the block is enabled.
***Busy* (BOOL):**
This output indicates that the block has been commanded to capture the position of the axis and is waiting for the capture event.
***Error* (BOOL):**
This output indicates that an error has occurred while processing the block.
***ErrorCode* (BOOL):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.
***RecordedPosition* (REAL):**
This output indicates the captured position on the axis defined in the "DriveNumber" input at the time of the rising edge of the input defined in "FastInput".

6.12.1.4 Block Call
sbTouchProbeS1(Enable, BlockCapture, DriveNumber, FastInput, ReadCoordinateAxis)

6.12.2 Trademark Registration - PlasticBagRegist1



6.12.2.1 Operation
This block has the function of controlling the register of marks for the specified axis according to the reading of marks made by the configured photocell.

6.12.2.2 Entries
***Enable* (BOOL):**
This entry enables the execution of the block.
***DriverNumber* (INT):**
This input sets the number of the drive to which the photocell is connected.
***FastInput* (INT):**
This input selects which fast digital input will be used to trigger the capture event.
***CompensationAxis* (INT):**
This input defines the Axis number to which the photocell correction will be applied.
***DistBtwMarks* (REAL):**
This entry defines the distance between marks.
***Tolerance* (REAL):**
This entry defines the window around the expected mark detection position for considering a mark good.
***DistToCut* (REAL):**
This input defines the distance between the mark detection and the material cut.
***NumberOfSamples* (REAL):**
This entry defines the number of distance samplings between marks for averaging.
***BlockMark* (BOOL):**
This input blocks the photocell reading.
***ResetCountMarks* (BOOL):**
This entry resets the output variable that counts the marks.
***TimeCompensation* (INT):**
This input compensates for the photo position reading (in microseconds).
***EnableMarkCount* (BOOL):**
This input enables the mark queue, in case the photocell is offset from the correction/cutting actuation point.

6.12.2.3 Enable
d outputs
(BOOL):
This output indicates that the block is enabled.
***Error* (BOOL):**
This output indicates an error occurred while processing the block.
***ErrorCode* (DINT):**
This output indicates the error code if the "Error" output is TRUE. The meanings of the errors are listed in the error table of the blocks.
***AverageDistance* (REAL):**
This output indicates the average distance between marks.

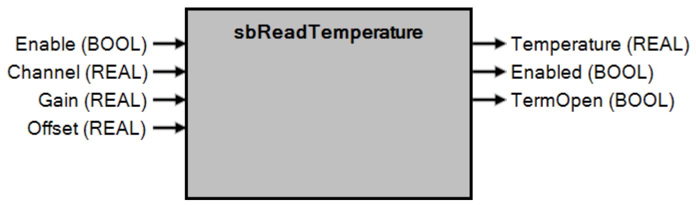
***MarkError* (REAL):**
This output indicates the difference between expected position and captured position.
***GoodMarks* (REAL):**
This output indicates the number of good marks.
***BadMarks* (REAL):**
This output indicates the number of bad marks.
***MarkBlocked* (BOOL):**
This output indicates that the mark capture by the photocell has been blocked.
***InstantLenght* (REAL):**
This output indicates the instantaneous measured position.
Position (REAL):
This output indicates the position of the axis where the photocell is.

6.12.2.4 Block Call

sbPlasticBagRegist1(Enable, CompensationAxis, DriveNumber, FastInput, DistBtwMarks, Tolerance, NumberOfSamples,
DistToCut, BlockMark, EnableMarkCount, ResetCountMarks, TimeCompensation)

6.13 Temperature

6.13.1 Read Temperature - ReadTemperature



6.13.1.1 Operation

This block has the function of reading the temperature. It returns the temperature read from the thermocouple multiplied by the value set at the "Gain" input and added (or subtracted) by the value set at the "Offset" input.
Equation:

Temperature = (ThermocoupleTemp * Gain) + Offset

Wher
e: **ThermocoupleTemp** is the temperature read from the thermocouple
Gain is the "Gain" input
Offset is the "Offset" entry
Temperature is the "Temperature" output

6.13.1.2 Entries:

***Enable* (BOOL):**
This entry enables the block.
***Channel* (REAL):**
This input defines the channel number from which the thermocouple is to be read.
***Gain* (REAL):**
This input sets the gain that will be multiplied by the value read from the thermocouple.
***Offset* (REAL):**
This input sets the offset that will be added (or subtracted, if negative) to the result of multiplying the value read from the thermocouple
and the gain set on the "Gain" input.

6.13.1.3 Outputs:

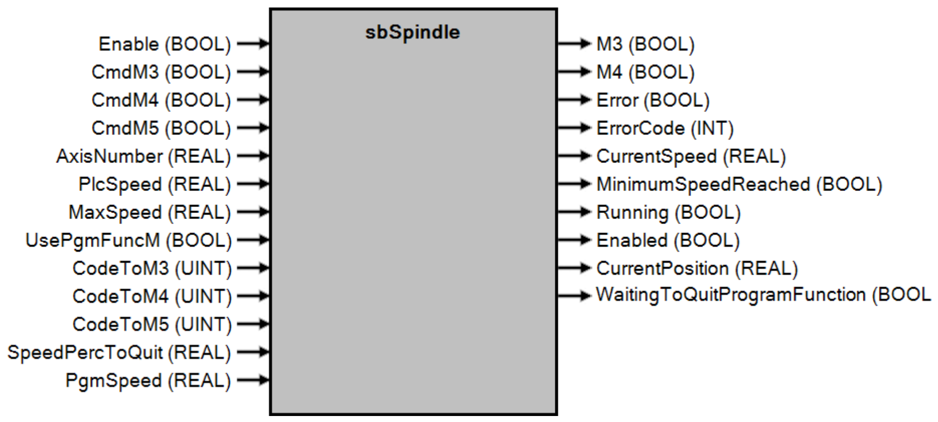
***Enabled* (BOOL):**
This output indicates that the block has been enabled.
***Temperature* (REAL):**
This output indicates the temperature value read, with gain and offset already applied.
***TermOpen* (REAL):**
This output indicates whether the thermocouple is open.

6.13.1.4 Block Call

sbReadTemperature(Enable, Channel, Gain, Offset)

6.14 Spindle

6.14.1 General shaft tree - sbSpindle



6.14.1.1 Operation
This block has the function of continuously moving a spindle.
The spindle can be driven via the PLC using inputs "CmdM3", "CmdM4", and "CmdM5". The spindle can also be driven by executing an ISO program, where the M function codes are set on the "CodeToM3", "CodeToM4", and "CodeToM5" inputs.
This block can be used in more than one spindle.

6.14.1.2 Entries
***Enable* (BOOL):**
This input enables the block to use the spindle when the value of this input is TRUE. If the value of this input is FALSE the block is disabled and all outputs receive null values.
***CmdM3* (BOOL):**
This input requests the block to execute the clockwise movement of the axis defined in the "AxisNumber" input. To interrupt this movement it must be requested through the "CmdM5" input. ***CmdM4* (BOOL):**
This input prompts the block to execute the counterclockwise movement of the axis defined in input "AxisNumber". To stop this movement it is necessary to request it via the "CmdM5" input.
***CmdM5* (BOOL):**
This entry prompts the block to cancel the movement in the axis defined in the "AxisNumber" entry.
***AxisNumber* (REAL):**
This entry defines the number of the axis (1 to 32) that you want to move.
***PlcSpeed* (REAL):**
This input sets the speed at which the axis moves when motion is requested by the inputs "CmdM3" or "CmdM4".
***MaxSpeed* (REAL):**
This input defines the maximum speed at which the axis can move. This input will be used internally to set the value of the analog output.
***UsePgmFuncM* (BOOL):**
This input defines whether the block will be used to receive M-function codes via the ISO program when executing programs by the CNC.
***CodeToM3* (UINT):**
This entry defines the M function code number (sent through the ISO program) that will execute the clockwise movement of the axis defined in the "AxisNumber" entry.

CodeToM4 **(UINT):**
This entry defines the M-function code number (sent through the ISO program) that will execute the counterclockwise movement of the axis defined in the "AxisNumber" entry.
CodeToM5 **(UINT):**
This entry defines the M function code number (sent through the ISO program) that will stop the movement of the axis defined in the "AxisNumber" entry.
SpeedPercToQuit **(REAL):**
This input defines the value in percent that will be used to discharge the M function and continue ISO program execution.
PgmSpeed **(REAL):**
This input defines the speed at which the axis moves when a movement is requested by the ISO program using the codes defined in the "CodeToM3" or "CodeToM4" entries.

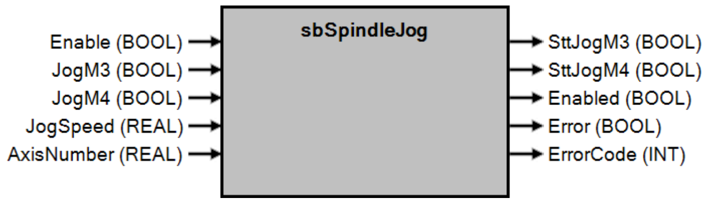
6.14.1.3 Outputs

M3 **(BOOL):**
This output indicates that the axis is moving clockwise.
M4 **(BOOL):**
This output indicates that the shaft is moving counterclockwise.
Error **(BOOL):**
This output indicates that an error has occurred on the axis.
ErrorCode **(INT):**
This output indicates the code of the error that happened on the axis.
CurrentSpeed **(REAL):**
This output indicates the current speed that the shaft is moving.
MinimumSpeedReached **(BOOL):**
This output indicates that the axis is moving and has reached the speed percentage defined in the input "SpeedPercToQuit.
Running **(BOOL):**
This output indicates that the axis is moving.
Enabled **(BOOL):**
This output indicates that the block is enabled and ready for use.
CurrentPosition **(REAL):**
This output indicates the current position that the axis is in.
WaitingToQuitProgramFunction **(BOOL):**
This output indicates that the block has received an ISO program code to move the axis. The block has already started the axis and is waiting to reach the speed percentage defined in the "SpeedPercToQuit" input in order to discharge the M function and continue program execution by the CNC.

6.14.1.4 Block Call

sbSpindle(Enable, CmdM3, CmdM4, CmdM5, AxisNumber, PlcSpeed, MaxSpeed, UsePgmFuncM, CodeToM3, CodeToM4, CodeToM5, SpeedPercToQuit, PgmSpeed)

6.14.2 Tree Axis Jog - sbSpindleJog



6.14.2.1 Operation

This block has the function of momentarily moving a spindle.
The spindle drive can be done through the PLC using the inputs "CmdM3", "CmdM4", and "CmdM5".
This block can be used in more than one spindle.

6.14.2.2 Entries

***Enable* (BOOL):**

This input enables the block to use the spindle when the value of this input is TRUE. If the value of this input is FALSE the block is disabled and all outputs receive null values.

***JogM3* (BOOL):**

This input requests the block to execute the momentary clockwise movement of the axis defined in the "AxisNumber" input. The motion will be executed as long as this input is active.

***JogM4* (BOOL):**

This input requests the block to execute the momentary counterclockwise motion of the axis defined in the "AxisNumber" input. The motion will be executed while this input is active.

***JogSpeed* (REAL):**

This input sets the speed at which the axis moves when motion is requested by the inputs
"JogM3" or "JogM4".

***AxisNumber* (REAL):**

This entry defines the number of the axis (1 to 32) that you want to move.

6.14.2.3 Outputs

***SttJogM3* (BOOL):**

This output indicates that the axis is moving clockwise.

***SttJogM4* (BOOL):**

This output indicates that the shaft is moving counterclockwise.

***Enabled* (BOOL):**

This output indicates that the block is enabled and ready for use.

***Error* (BOOL):**

This output indicates that an error has occurred on the axis.

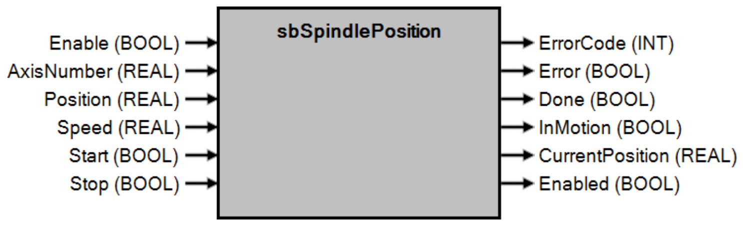
***ErrorCode* (INT):**

This output indicates the code of the error that happened on the axis.

6.14.2.4 Block Call

sbSpindleJog(Enable, JogM3, JogM4, JogSpeed, AxisNumber)

6.14.3 Spindle Position - sbSpindlePosition



6.14.3.1 Operation

This block has the function of moving a spindle to a position defined in the "Position" input.
The spindle must be started via the PLC using the "Start" input.
This block can be used in more than one spindle.

6.14.3.2 Entries

***Enable* (BOOL):**
This input enables the block to use the spindle when the value of this input is TRUE. If the value of this input is FALSE the block is disabled and all outputs receive null values.

***AxisNumber* (REAL):**
This entry defines the number of the axis (1 to 32) that you want to move.

***Position* (REAL):**
This entry defines the final position of the movement that will be executed.

***Speed* (REAL):**
This input sets the speed at which the axis moves when a move is requested by the "Start" input.

***Start* (BOOL):**
This input requests the block to execute the movement to the position defined in the "Position" input with speed set at the "Speed" input.

***Stop* (BOOL):**
This input requests the block to stop the movement that is being executed.

6.14.3.3 Outputs

***ErrorCode* (INT):**
This output indicates the error code that happened in the block.

***Error* (BOOL):**
This output indicates that an error has occurred in the processing of the block.

***Done* (BOOL):**
This output indicates that the requested movement has been completed.

***InMotion* (BOOL):**
This output indicates that the axis is executing the requested motion.

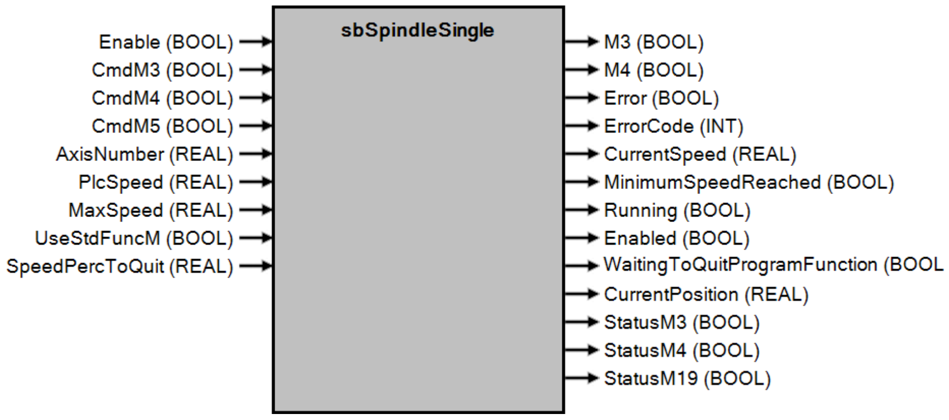
***CurrentPosition* (REAL):**
This output indicates the current position of the axis defined in the "AxixNumber" input.

***Enabled* (BOOL):**
This output indicates that the block is enabled and ready for use.

6.14.3.4 Block Call

sbSpindlePosition(Enable, AxisNumber, Position, Speed, Start, Stop)

6.14.4 Single Tree Axis - sbSpindleSingle



6.14.4.1 Operation

This block has the function of continuously moving a spindle.
The spindle can be driven via the PLC using the inputs "CmdM3", "CmdM4", and "CmdM5". The spindle can also be driven through ISO program execution, where the standard M function codes (M3, M4, and M5) will be used if the "UseStdFuncM" input is set to TRUE.
This block can be used in machines that have only one spindle.

6.14.4.2 Entries

***Enable* (BOOL):**
This input enables the block to use the spindle when the value of this input is TRUE. If the value of this input is FALSE the block is disabled and all outputs receive null values.

***CmdM3* (BOOL):**
This input requests the block to execute the clockwise movement of the axis defined in the "AxisNumber" input. To interrupt this movement it must be requested through the "CmdM5" input. ***CmdM4* (BOOL):**
This input prompts the block to execute the counterclockwise movement of the axis defined in input "AxisNumber". To stop this movement it is necessary to request it via the "CmdM5" input.

***CmdM5* (BOOL):**
This entry prompts the block to cancel the movement in the axis defined in the "AxisNumber" entry.

***AxisNumber* (REAL):**
This entry defines the number of the axis (1 to 32) that you want to move.

***PlcSpeed* (REAL):**
This input sets the speed at which the axis moves when motion is requested by the inputs "CmdM3" or "CmdM4".

***MaxSpeed* (REAL):**
This input defines the maximum speed at which the axis can move. This input will be used internally to set the value of the analog output.

***UseStdFuncM* (BOOL):**
This input defines whether the block will be used to receive M-function codes via the ISO program when executing programs by the CNC.

***SpeedPercToQuit* (REAL):**
This input defines the value in percent that will be used to discharge the M function and continue ISO program execution.

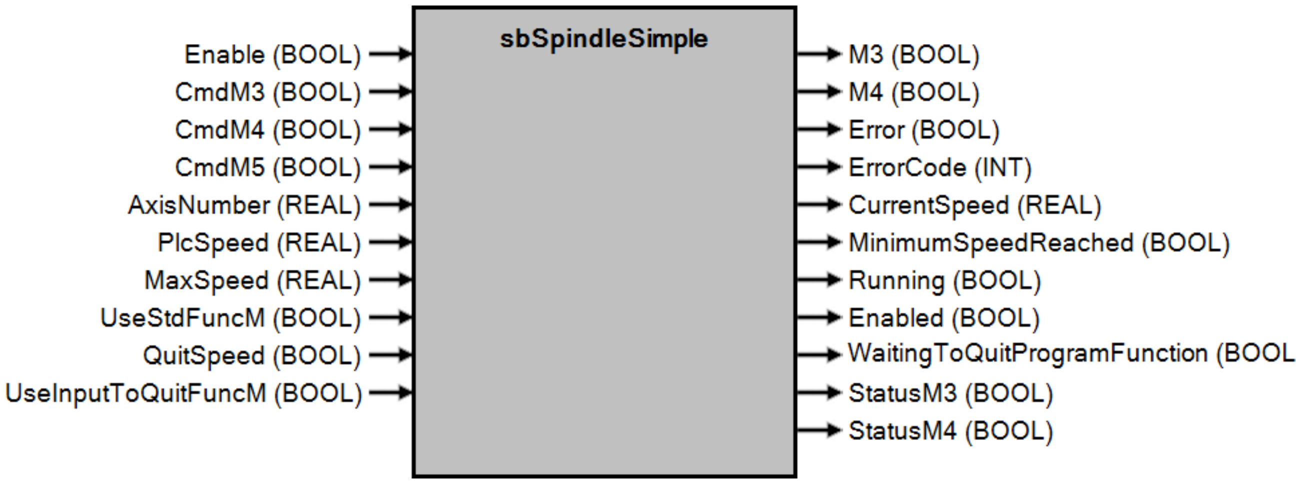
6.14.4.3 Outputs

M3 **(BOOL):**
This output indicates that the axis is moving clockwise.
M4 **(BOOL):**
This output indicates that the shaft is moving counterclockwise.
Error **(BOOL):**
This output indicates that an error has occurred on the axis.
ErrorCode **(INT):**
This output indicates the code of the error that happened on the axis.
CurrentSpeed **(REAL):**
This output indicates the current speed that the shaft is moving.
MinimumSpeedReached **(BOOL):**
This output indicates that the axis is moving and has reached the speed percentage defined in the input "SpeedPercToQuit.
Running **(BOOL):**
This output indicates that the axis is moving.
Enabled **(BOOL):**
This output indicates that the block is enabled and ready for use.
WaitingToQuitProgramFunction **(BOOL):**
This output indicates that the block has received an ISO program code to move the axis. The block has already started the axis and is waiting to reach the speed percentage defined in the "SpeedPercToQuit" input in order to discharge the M function and continue program execution by the CNC.
CurrentPosition **(REAL):**
This indicates the current position that the axis is in.
StatusM3 **(BOOL):**
This output indicates that the CNC has received a command to move the spindle clockwise.
StatusM4**(BOOL):**
This output indicates that the CNC has received a command to move the spindle counterclockwise.
StatusM19**(BOOL):**
This output indicates that the CNC has received a command to move the spindle clockwise.

6.14.4.4 Block Call

sbSpindleSingle(Enable, CmdM3, CmdM4, CmdM5, AxisNumber, PlcSpeed, MaxSpeed, UseStdFuncM, SpeedPercToQuit)

6.14.4 Single Spindle - sbSpindleSimple



6.14.4.1 Operation

This block has the function of continuously moving a spindle.

The spindle can be driven via the PLC using the inputs "CmdM3", "CmdM4", and "CmdM5". The spindle can also be driven through ISO program execution, where the standard M function codes (M3, M4, and M5) will be used if the "UseStdFuncM" input is set to TRUE.

This block can be used in machines that have only one spindle.

6.14.4.2 Entries

***Enable* (BOOL):**

This input enables the block to use the spindle when the value of this input is TRUE. If the value of this input is FALSE the block is disabled and all outputs receive null values.

***CmdM3* (BOOL):**

This input requests the block to execute the clockwise movement of the axis defined in the "AxisNumber" input. To interrupt this movement it must be requested through the "CmdM5" input. ***CmdM4* (BOOL):**

This input prompts the block to execute the counterclockwise movement of the axis defined in input "AxisNumber". To stop this movement it is necessary to request it via the "CmdM5" input.

***CmdM5* (BOOL):**

This entry prompts the block to cancel the movement in the axis defined in the "AxisNumber" entry.

***AxisNumber* (REAL):**

This entry defines the number of the axis (1 to 32) that you want to move.

***PlcSpeed* (REAL):**

This input sets the speed at which the axis moves when motion is requested by the inputs "CmdM3" or "CmdM4".

***MaxSpeed* (REAL):**

This input defines the maximum speed at which the axis can move. This input will be used internally to set the value of the analog output.

***UseStdFuncM* (BOOL):**

This input defines whether the block will be used to receive M-function codes via the ISO program when executing programs by the CNC.

***QuitSpeed* (BOOL):**

This input defines the signal will be used to determine the continuation of ISO program execution, if the "UseInputToQuitFuncM" for TRUE.

UseInputToQuitFuncM **(BOOL):**
This input defines whether the "QuitSpeed" input will be used to determine the continuation of ISO program execution. If this input is FALSE the CNC will continue execution normally, without waiting for the spindle. For safety, a timing function can be added to the ISO program to address this situation.

6.14.4.3 Outputs

M3 **(BOOL):**
This output indicates that the axis is moving clockwise.

M4 **(BOOL):**
This output indicates that the shaft is moving counterclockwise.

Error **(BOOL):**
This output indicates that an error has occurred on the axis.

ErrorCode **(INT):**
This output indicates the code of the error that happened on the axis.

CurrentSpeed **(REAL):**
This output indicates the current speed that the shaft is moving.

MinimumSpeedReached **(BOOL):**
This output indicates that the axis is moving and has reached the speed percentage defined in the input "SpeedPercToQuit.

Running **(BOOL):**
This output indicates that the axis is moving.

Enabled **(BOOL):**
This output indicates that the block is enabled and ready for use.

WaitingToQuitProgramFunction **(BOOL):**
This output indicates that the block has received an ISO program code to move the axis. The block has already started the axis and is waiting to reach the speed percentage defined in the "SpeedPercToQuit" input in order to discharge the M function and continue program execution by the CNC.

StatusM3 **(BOOL):**
This output indicates that the CNC has received a command to move the spindle clockwise.

StatusM4**(BOOL):**
This output indicates that the CNC has received a command to move the spindle counterclockwise.

6.14.4.4 Block Call

sbSpindleSimple(Enable, CmdM3, CmdM4, CmdM5, AxisNumber, PlcSpeed, MaxSpeed, UseStdFuncM, QuitSpeed, UseInputToQuitFuncM)

7.0 PLC Example

The following is an excerpt from a base PLC program made for the CNC.
This program is full of comments for a better understanding by any programmer who may need to understand/change the program.

```
//====================
// PROGRAM START
//====================

//==============================
// Reading Parameters
//==============================

IF svParametersUpdated THEN
    PosicaoTrocaX := sfReadParameter( 0, 0, 2 );
    PosicaoTrocaY := sfReadParameter( 0, 1, 2 );
    PosicaoTrocaZ := sfReadParameter( 0, 2, 2 );
END_IF;

//==============================
// System bits
//==============================
//----------------------------------------------------------------------------------------------------
//Emergency
//----------------------------------------------------------------------------------------------------

//Emergency buttons svEmergencyCnc
   := iEmergency;

//Emergency exit
   oEmergency := svEmergencyOutput;

//Messages
   MESSAGE(0, ((NOT svEmergencyCnc) AND svInitializationMode));
   ALARM(0, (NOT svEmergencyOutput AND svCncInitialized));

//----------------------------------------------------------------------------------------------------
//Starting and Stopping and Blocking
//----------------------------------------------------------------------------------------------------

//Start for CNC
   svStartCnc := iStart;

//Stop for CNC
   svStopCnc := iStop;

//Abort execution
   svAbortCnc := ((NOT svEmergencyCnc) OR (NOT iInverterOK) OR (NOT iPressaoDoArOK));

//Departure lock
   svBlockStart := (NOT iInverterOK) OR (NOT iPressaoDoArOK);

//Messages
   TIMED_MESSAGE(0, (svBlockStart AND svStartCnc));
   MESSAGE(2, (NOT svStopCnc));
   MESSAGE(12, (NOT iInverterOK));
   MESSAGE(13, (NOT iPressaoDoArOK));
```

```
//=====================================================================
//Manual movements - Jog
//=====================================================================

    / /enabling jog moves
    HabJogX := svCncInitialized AND (NOT svExecutionRun) AND (bJogPositiveX OR iJogPosX OR
bJogNegativeX OR iJogNegX);
    HabJogY := svCncInitialized AND (NOT svExecutionRun) AND (bJogPositiveY OR iJogPosY OR
bJogNegativeY OR iJogNegY);
    HabJogZ := svCncInitialized AND (NOT svExecutionRun) AND (bJogPositiveZ OR iJogPosZ OR
bJogNegativeZ OR iJogNegZ);

    //performs the jog moves (instances of the sbJog block)
    JogX(HabJogX, (bJogPositiveX OR iJogPosX) AND NOT End-ofCourseX.PositiveLimit,
(bJogNegativeX OR iJogNegX) AND NOT End-ofCourseX.NegativeLimit, 1, 0);
    JogY(HabJogY, (bJogPositiveY OR iJogPosY) AND NOT End-ofCourseY.PositiveLimit,
(bJogNegativeY OR iJogNegY) AND NOT End-ofCourseY.NegativeLimit, 2, 0);
    JogZ(HabJogZ, (bJogPositiveZ OR iJogPosZ) AND NOT EndFromCourseZ.PositiveLimit,
(bJogNegativeZ OR iJogNegZ) AND NOT EndFromCourseZ.NegativeLimit, 3, 0);

    //Messages of jog errors
    MESSAGE(3, JogX.Error);
    MESSAGE(4, JogY.Error);
    MESSAGE(5, JogZ.Error);



//=====================================================================
//Finals of course
//=====================================================================

    //Travel end for each axis (instances of sbLimitSwitch block)
    EndPathX(svCncInitialized, svOnKeyCE, iOffPathX OR ReferenceRunning,
iEndFromCourseNegX, FALSE, TRUE);
    EndFromCourseY(svCncInitialized, svOnKeyCE, iFendFromCoursePosY OR ReferenceRunning,
iFendFromCourseNegY, FALSE, TRUE);
    EndFromCourseZ(svCncInitialized, svOnKeyCE, iFendFromCoursePosZ OR ReferenceRunning,
iFendFromCourseNegZ, FALSE, TRUE);

    //aborts execution if it has any endpoints
    IF R_EDGE(EndCourseX.Alarm OR EndCourseY.Alarm OR EndCourseZ.Alarm) THEN svAbortCnc
             := TRUE;
    END_IF;

    End-of-CourseError := End-ofCourseX.Alarm OR End-ofCourseY.Alarm OR End-
    ofCourseZ.Alarm;

    //end of stroke messages
    MESSAGE(6, End-ofCourseX.PositiveLimit);
    MESSAGE(7, End-ofCourseX.NegativeLimit);
    MESSAGE(8, End-ofCourseY.PositiveLimit);
    MESSAGE(9, End-ofCourseY.NegativeLimit);
    MESSAGE(10, EndFromCourseZ.PositiveLimit);
    MESSAGE(11, EndFromCourseZ.NegativeLimit);
    MESSAGE(14, EndFromCourseError;)
```

```
//======================================================================
//Axis reference
//======================================================================

    //requests reference lookup IF
    sfKeyCode(2041) THEN
        StartFindReference := TRUE; END_IF;

    //unmark axis reference - on request cancel search or interrupt IF
    sfKeyCode(2042) OR sfKeyCode(2044) THEN
        StartFindReference := FALSE;
        ReferenceAxisX.Done := FALSE;
        ReferenceAxisY.Done := FALSE;
        ReferenceAxisZ.Done := FALSE;
    END_IF;

    //reference to each axis (instances of the sbHoming block)
    ReferenceAxisZ(StartSourceReference, iFendFromPosZ, FALSE, FALSE, 3);
    ReferenceAxisX(ReferenceAxisZ.Done, iFendFromPosX, FALSE, FALSE, 1);
    ReferenceAxisY(ReferenceAxisZ.Done, iFendFromPosY, FALSE, FALSE, 2);

    //indication of referencing ok
    ReferencedAxes := ReferenceAxisX.Done AND ReferenceAxisY.Done AND
ReferenceAxisZ.Done;
    //referral indication in progress
    ReferenceRunning := ReferenceAxisX.Running OR ReferenceAxisY.Running OR
ReferenceAxisZ.Running;

    //cancels the search request, at the end of the search of the first
    axis IF R_EDGE(ReferenceAxisZ.Done) THEN
        StartFindReference := FALSE; END_IF;
```