List of related manuals

<table>
<thead>
<tr>
<th>Hardware manuals and guides</th>
<th>Code (English)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PVS800-57 hardware manual</td>
<td>3AUA0000053689</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Firmware manuals and guides</th>
<th>Code (English)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PVS800 firmware manual</td>
<td>3AUA0000058422</td>
</tr>
<tr>
<td>Application guide: Adaptive program for PVS800</td>
<td>3AUA0000091276</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option manuals and guides</th>
<th>Code (English)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manuals and quick guides for I/O extension modules, fieldbus adapters, etc.</td>
<td>2)</td>
</tr>
</tbody>
</table>

1) Delivered as a printed copy with the PVS800.

2) Delivered as a printed copy with the option.
Application guide

Adaptive program for PVS800 central inverters
# Table of contents

List of related manuals ................................................. 2

1. Introduction to the guide
   What this chapter contains .......................................... 7
   Applicability ............................................................ 7
   Safety instructions ...................................................... 7
   Target audience .......................................................... 7
   Contents of the guide ................................................... 8

2. Adaptive program
   What this chapter contains .......................................... 9
   What an adaptive program is ......................................... 9
   How to build the program ............................................. 10
   How to connect the adaptive program to the PVS800 master control program .................................................. 11
   How to control the execution of the program ....................... 11

3. Application blocks
   What this chapter contains .......................................... 13
   General rules ............................................................. 13
   Block inputs ............................................................ 13
   User parameters ........................................................ 13
   Function blocks ........................................................ 14
   ABS ................................................................. 14
   ADD ................................................................. 14
   AND ................................................................. 15
   BSET ................................................................. 15
   COMPARE .......................................................... 16
   COUNT .............................................................. 17
   DPOT ................................................................. 17
   EVENT ............................................................... 18
   FILTER .............................................................. 18
   MAX ................................................................. 19
   MIN ................................................................. 19
   MULDIV ............................................................ 19
   OR ................................................................. 20
   PI ................................................................. 21
   PI-BAL ............................................................. 21
   RAMP ............................................................ 22
   SR ............................................................... 23
   TOFF .............................................................. 25
   TON ............................................................... 26
   TRIGG ............................................................. 27
   WR-I ............................................................... 28
   WR-PB ............................................................ 29
   XOR ............................................................... 30
   I/O and communication blocks .................................. 31
   A/F WORD ........................................................ 31
### Further information

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI1</td>
<td>31</td>
</tr>
<tr>
<td>AI2</td>
<td>32</td>
</tr>
<tr>
<td>AI3</td>
<td>32</td>
</tr>
<tr>
<td>AO1</td>
<td>33</td>
</tr>
<tr>
<td>AO2</td>
<td>33</td>
</tr>
<tr>
<td>CW</td>
<td>34</td>
</tr>
<tr>
<td>DI1…DI6, DI IL</td>
<td>34</td>
</tr>
<tr>
<td>DO1</td>
<td>35</td>
</tr>
<tr>
<td>DO2</td>
<td>35</td>
</tr>
<tr>
<td>DO3</td>
<td>35</td>
</tr>
<tr>
<td>EXT1 DI…DI3</td>
<td>36</td>
</tr>
<tr>
<td>EXT2 DI…DI3</td>
<td>36</td>
</tr>
<tr>
<td>EXT3 DI…DI3</td>
<td>37</td>
</tr>
<tr>
<td>EXT4 DI…DI3</td>
<td>37</td>
</tr>
<tr>
<td>EXT5 DI…DI3</td>
<td>38</td>
</tr>
<tr>
<td>EXT DO</td>
<td>38</td>
</tr>
<tr>
<td>EXT1…5 AI1…AI2</td>
<td>39</td>
</tr>
<tr>
<td>EXT1…5 AO1…AO2</td>
<td>39</td>
</tr>
<tr>
<td>FUNG IN</td>
<td>40</td>
</tr>
<tr>
<td>FUNG OUT</td>
<td>40</td>
</tr>
<tr>
<td>PZD3 OUT</td>
<td>41</td>
</tr>
<tr>
<td>PZD4 OUT</td>
<td>42</td>
</tr>
<tr>
<td>PZD5 OUT</td>
<td>42</td>
</tr>
<tr>
<td>PZD8 OUT</td>
<td>44</td>
</tr>
<tr>
<td>PZD9 OUT</td>
<td>44</td>
</tr>
<tr>
<td>PZD10 OUT</td>
<td>45</td>
</tr>
<tr>
<td>PZD3 IN</td>
<td>45</td>
</tr>
<tr>
<td>PZD4 IN</td>
<td>46</td>
</tr>
<tr>
<td>PZD5 IN</td>
<td>46</td>
</tr>
<tr>
<td>PZD6 IN</td>
<td>47</td>
</tr>
<tr>
<td>PZD7 IN</td>
<td>47</td>
</tr>
<tr>
<td>PZD8 IN</td>
<td>48</td>
</tr>
<tr>
<td>PZD9 IN</td>
<td>48</td>
</tr>
<tr>
<td>PZD10 IN</td>
<td>49</td>
</tr>
<tr>
<td>SUB</td>
<td>49</td>
</tr>
</tbody>
</table>
Introduction to the guide

What this chapter contains
This chapter gives general information on the guide.

Applicability
The guide is applicable to the master control program of PVS800 central inverters.

Note: All parameter numbers and names refer to the PVS800 master control program (see PVS800 Firmware manual) unless otherwise indicated.

Safety instructions
Follow all safety instructions delivered with the PVS800.

• Read the complete safety instructions before you install, commission, or use the PVS800. The complete safety instructions are given at the beginning of the Hardware Manual.

• Read the software function specific warnings and notes before changing the default settings of the function. These warnings and notes are presented together with the parameter descriptions wherever appropriate.

WARNING! ABB is not responsible for the operation of a custom-made adaptive program, or for any damage or injury caused by the use of it.

Target audience
This manual is intended for people who adjust the parameters of, or operate, monitor or troubleshoot PVS800 central inverters.
The reader is expected to know the standard electrical wiring practices, electronic components, and electrical schematic symbols.

**Contents of the guide**

The guide is to be used together with the *PVS800 firmware manual*. The firmware manual contains the basic information on the master control program parameters including the parameters of the adaptive program. This guide gives more detailed information on the adaptive program:

- what the adaptive program is
- how to build an adaptive program
- how the function blocks operate
- how to document the adaptive program.
Adaptive program

What this chapter contains

This chapter describes the basics of the adaptive program and instructs in building a program.

What an adaptive program is

Conventionally, the user can control the operation of the PVS800 by parameters. Each parameter has a fixed set of choices or a range of settings. The parameters make programming easy, but the choices are limited: you cannot customize the operation any further. The adaptive program makes freer customization possible without the need of a special programming tool or language.

The adaptive program is built of function blocks (see listing starting on page 13) by using the DriveAP 2.x PC tool or the CDP 312R control panel. The maximum size of the adaptive program is 10 blocks at 10 ms time level and 20 blocks at 100 ms time level.

The user can document the program by drawing it on block diagram template sheets.

Notes:

- Only the master control program of the PVS800 is intended to be programmed this way.
- If the adaptive program originally created using the DriveAP 2.x tool is modified with the control panel, the function block view in DriveAP 2.x will be different since the location of the function blocks cannot be determined by the control panel.
How to build the program

The programmer connects a function block to other blocks through a block parameter set. The sets are also used for reading values from the master control program, and transferring data to the master control program. Each block parameter set consists of five parameters.

The figure shows the use of block parameter set 1 in the PVS800 master control program (parameters 55.05…55.09).

- Parameter 55.05 selects the function block type
- Parameter 55.06 selects the source that input I1 of the function block is connected to
- Parameter 55.07 selects the source that input I2 of the function block is connected to
- Parameter 55.08 selects the source that input I3 of the function block is connected to
- Parameter 55.09 stores the value of the function block output. The user cannot edit the parameter value.
How to connect the adaptive program to the PVS800 master control program

The output of the adaptive program needs to be connected to the PVS800 master control program. For that purpose, the user needs

- a **WR-I** block for parameters that need a numerical value
- a **WR-PB** block for packed Boolean types of parameters where bits of a word are set.

How to control the execution of the program

The adaptive program executes the function blocks in numerical order. The execution order of the blocks can be changed by DriveAP 2.x. The user can

- select the time level, either 10 or 100 ms
- select the operating mode of the program (stop, start, editing)
- delete or add blocks
- add comments (only with DriveAP 2.x)
- add symbolic names for block outputs (only with DriveAP 2.x).

Example:

Controlling parameter 31.04 UDC **START LIM** using a **WR-I** block.

Example:

Changing the state of bit 3 in parameter 09.13 **AP FW** using a **WR-PB** block.
Adaptive program
Application blocks

What this chapter contains
The chapter describes the function blocks available in the PVS800 master control program.

General rules
The use of the first input is compulsory (it must not be left unconnected). Use of the second and third input is voluntary for most blocks. As a rule of thumb, an unconnected input does not affect the output of the block.

Block inputs
The blocks use three input formats: integer, boolean and text string. The format to use depends on the block. For example, the ADD block uses integer inputs and the OR block boolean inputs. Text string format is used only by EVENT blocks.

Notes:
• The inputs of the blocks are read when the execution of the block starts, not simultaneously for all blocks.
• If blocks are programmed with CDP 312R control panel (or DriveWindow PC tool), DriveAP 2.x cannot form a graphic block diagram. Only use DriveAP for adaptive programming if documentation of blocks is needed.

User parameters
There are 10 user parameters available for application purposes. See parameters 53.01 NUMERIC 1 ... 53.10 NUMERIC 10. Group 19 parameters can also be used.
## Function blocks

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
<th>Illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ABS</strong></td>
<td>Arithmetic function</td>
<td>This block is used to obtain the absolute value of an integer number.</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Illustration</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Operation</strong></td>
<td>The output is the absolute value of INPUT multiplied by input MUL and divided by input DIV.</td>
<td>OUT =</td>
</tr>
<tr>
<td><strong>Connections</strong></td>
<td>Inputs INPUT, MUL and DIV: 24-bit integer values (23 bits + sign)</td>
<td>Output OUT: 24-bit integer (23 bits + sign)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
<th>Illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADD</strong></td>
<td>Arithmetic function</td>
<td>This block is used to calculate the sum of integers.</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Illustration</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Operation</strong></td>
<td>The output is the sum of the inputs.</td>
<td>OUT = ADD1 + ADD2 + ADD3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This block is also used for subtraction. See <strong>SUB</strong> block.</td>
</tr>
<tr>
<td><strong>Connections</strong></td>
<td>Input ADD1, ADD2 and ADD3: 24-bit integer values (23 bits + sign)</td>
<td>Output OUT: 24-bit integer (23 bits + sign)</td>
</tr>
</tbody>
</table>
### AND

**Type** Logical function

**Summary** This block is used to form a logical AND function of boolean input variables.

**Illustration**

```
  BLOCK x
  AND
    I1
    I2
    I3
    OUT
```

**Operation** The output is true if all connected inputs are true. Otherwise the output is false. Truth table:

<table>
<thead>
<tr>
<th>I1</th>
<th>I2</th>
<th>I3</th>
<th>OUT (binary)</th>
<th>OUT (value on display)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>True (All bits 1)</td>
<td>-1</td>
</tr>
</tbody>
</table>

**Connections**

- Input `I1`, `I2` and `I3`: Boolean values
- Output `OUT`: 24-bit integer value (packed boolean)

### BSET

**Type** A bit setting of an integer word

**Summary** This block is used to change the state of one selected bit of an integer value. The integer usually contains packed boolean data.

**Illustration**

```
  BLOCK x
  BITSET
    ENABLE
    BITNR
    INPUT
    OUT
```

**Operation** If `ENABLE` is active (= 1), the function sets the bit defined by `BITNR` (0 = bit 0, 1 = bit 1, ...), and if not active (= 0), the function resets the bit defined by `BITNR`.

- **ENABLE**: Boolean value, set bit = 1, reset bit = 0
- **BITNR**: Bit number (0 = bit no. 0 ... 15 = bit no. 15)
- **INPUT**: Input word for chaining several blocks or for masking bit pattern

**Connections**

- Input `ENABLE`: Boolean value
- Inputs `BITNR` and `INPUT`: 24-bit integer values (23 bits + sign)
- Output `OUT`: 24-bit integer (23 bits + sign)
**Type**

Comparative function

**Summary**

This block is used to compare two integers.

**Illustration**

```
<table>
<thead>
<tr>
<th>BLOCK x</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPAR</td>
</tr>
<tr>
<td>I1</td>
</tr>
<tr>
<td>I2</td>
</tr>
<tr>
<td>OUT</td>
</tr>
<tr>
<td>HYS</td>
</tr>
</tbody>
</table>
```

**Operation**

Output bits 0, 1 and 2:
- If $I_1 > I_2$, OUT = ... 001 (output bit 0 is set)
- If $I_1 = I_2$, OUT = ... 010 (output bit 1 is set)
- If $I_1 < I_2$, OUT = ... 100 (output bit 2 is set).

Output bit 3:
- If $I_1 > I_2$, OUT = ... 1xxx (Output bit 3 is set and remains set until $I_1 < I_2 - I_3$, after which bit 3 is reset.)

Output bit 4 (O4 in figure):
- If $I_1 - I_2 - I_3 \geq 0$, output bit 4 = 1
- If $I_1 - I_2 + I_3 < 0$, output bit 4 = 0.

Output bit 5 (O5 in figure):
- If $I_3 \geq \mid I_1 - I_2 \mid$, output bit 5 = 1. **Note:** I3 must be $\geq 0$.
- If $I_3 < \mid I_1 - I_2 \mid$, output bit 5 = 0.

When this output is connected to a logical input, it is true if any bit is true.

**Connections**

Input I1, I2 and HYS: 24-bit integer values (23 bits + sign)

Output OUT: 24-bit integer (packed boolean)
## COUNT

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Counter function</td>
</tr>
</tbody>
</table>

This block is a reset counter with maximum value limit.

### Illustration

[Block diagram showing connections: BLOCK x, COUNT, TRIGG, RESET, MAX, OUT]

### Operation

The counter function counts rising edges of input TRIGG. The counter is reset by input RESET and limited to the value set with input MAX.

- **TRIGG:** Trigger input
- **RESET:** Resets output
- **MAX:** Maximum limit of output

**Note:** If pin 3 is left unconnected, the maximum count value is zero.

### Connections

- **Input TRIGG and RESET:** Boolean values
- **Input MAX:** 24-bit integer value (23 bits + sign)
- **Output OUT:** 24-bit integer (23 bits + sign)

## DPOT

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Up/down counter</td>
</tr>
</tbody>
</table>

This block is a digitally-controlled potentiometer.

### Illustration

[Block diagram showing connections: BLOCK x, DPOT, POS DIR, NEG DIR, RAMP STEP, OUT]

### Operation

The digitally controlled ramp function increments or decrements the output according to control inputs POS DIR and NEG DIR. Input POS DIR ramps the output to positive direction and NEG DIR to negative direction. If both inputs are active, nothing happens. The step is defined by input RAMP STEP.

- **POS DIR:** Control to positive direction
- **NEG DIR:** Control to negative direction
- **RAMP STEP:** Ramp step on program cycle

### Connections

- **Input POS DIR and NEG DIR:** Boolean values
- **Input RAMP STEP:** 24-bit integer value (23 bits + sign)
- **Output OUT:** 24-bit integer (23 bits + sign)
<table>
<thead>
<tr>
<th>EVENT</th>
<th>Type</th>
<th>Summary</th>
<th>Event function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>This block is an application-based alarm or fault event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Illustration</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Operation</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The block is used to write an event to the alarm or fault logger. A fault event will trigger a fault and trip the PVS800. Alarm events are reflected by the status word alarm bit. Input INPUT triggers the event. Input TEXT PAR selects the parameter index from which the event message (text string) is read. Use text parameters 53.11 STRING 1 ... 53.24 STRING 14 for user application-specific texts. Type the respective text by clicking the pin with shift pressed. Input TYPE selects the type of the event (warning or fault).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Connections</strong></td>
</tr>
<tr>
<td></td>
<td>INPUT</td>
<td>TEXT PAR</td>
<td>TYPE</td>
</tr>
<tr>
<td></td>
<td>0 -&gt; 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Parameter index</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inputs INPUT, TEXT PAR:</td>
<td>24-bit integer values (23 bits + sign)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Input TYPE:</td>
<td>String (compulsory)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FILTER</th>
<th>Type</th>
<th>Filtering function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>This block is used as a first order low pass filter for integer values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Illustration</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Operation</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The output is the filtered value of INPUT. Input TIME is the filtering time constant.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT = INPUT × (1 - e^{-TIME})</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> The internal calculation uses 48 bits accuracy to avoid offset errors.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Connections</strong></td>
</tr>
<tr>
<td></td>
<td>INPUT:</td>
<td>24-bit integer value (23 bits + sign)</td>
</tr>
<tr>
<td></td>
<td>TIME:</td>
<td>24-bit integer value (23 bits + sign). One corresponds to 1 ms.</td>
</tr>
<tr>
<td></td>
<td>Output OUT:</td>
<td>24-bit integer (23 bits + sign)</td>
</tr>
</tbody>
</table>
### MAX

**Type**  
Comparative function: maximum selector

**Summary**  
This block is used to select the highest value of inputs to the output.

#### Illustration

![MAX block diagram](#)

#### Operation

The values at the inputs I1, I2 and I3 are compared and the highest value is written to output OUT.

\[
OUT = \text{MAX} (I1, I2, I3)
\]

#### Connections

- Input I1, I2 and I3: 24-bit integer values (23 bits + sign)
- Output OUT: 24-bit integer (23 bits + sign)

### MIN

**Type**  
Comparative function: minimum selector

**Summary**  
This block is used to select the lowest value of inputs to the output.

#### Illustration

![MIN block diagram](#)

#### Operation

The values at the inputs I1, I2 and I3 are compared and the lowest value is written to output OUT.

\[
OUT = \text{MIN} (I1, I2, I3)
\]

#### Connections

- Input I1, I2 and I3: 24-bit integer values (23 bits + sign)
- Output OUT: 24-bit integer (23 bits + sign)

### MULDIV

**Type**  
Arithmetic function

**Summary**  
This block is used to scale an integer value by dividing the product of two integers with a third value.

#### Illustration

![MULDIV block diagram](#)

#### Operation

The output is the product of INPUT multiplied by MUL and divided by DIV.

\[
OUT = \frac{(\text{INPUT} \times \text{MUL})}{\text{DIV}}
\]

#### Connections

- Inputs INPUT, MUL and DIV: 24-bit integer values (23 bits + sign)
- Output OUT: 24-bit integer (23 bits + sign)
### OR

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical function</td>
<td>This block is used to form general combinatory expressions with boolean variables.</td>
</tr>
</tbody>
</table>

#### Illustration

![Diagram of OR block with inputs I1, I2, and I3 and output OUT.](attachment:or_diagram.png)

#### Operation

The output is true if any of the inputs is true.

**Truth table:**

<table>
<thead>
<tr>
<th>$I_1$</th>
<th>$I_2$</th>
<th>$I_3$</th>
<th>OUT</th>
<th>OUT (Value on display)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>True</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>True</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>True</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>True</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>True</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>True</td>
<td>-1</td>
</tr>
</tbody>
</table>

#### Connections

- **Input $I_1$, $I_2$, and $I_3$:** Boolean values
- **Output OUT:** 24-bit integer value (packed boolean)
### PI

**Type** | PI controller  
---|---
**Summary** | This block is used as a standard PI regulator in closed loop systems.

#### Illustration

![Diagram of PI block](image)

#### Operation

The output is $\text{INPUT} \times \frac{K}{100} + \frac{I}{100} \cdot \text{INPUT}$

**Note:** The internal calculation uses 48 bits accuracy to avoid offset errors.

#### Connections

- **Input INPUT:** 24-bit integer value (23 bit + sign)
- **Input K:** 24-bit integer value (23 bit + sign). Gain factor. 100 corresponds to 1. 10 000 corresponds to 100.
- **Input I:** Integrator coefficient. 100 corresponds to 1. 10 000 corresponds to 100.
- **Output OUT:** 24-bit integer (23 bits + sign). The range is limited to $-10000...10000$.

### PI-BAL

**Type** | Initialisation block for the PI controller
---|---

#### Illustration

![Diagram of PI-BAL block](image)

#### Operation

The block initialises the PI block first. When input BAL is true, the block writes the value of BAL REF to the output of the PI block. When input BAL becomes false, the block releases the output of the PI controller block which continues normal operation from the set output.

**Note:** The block may be used only with the PI block. The execution of the block must be after the PI block.

#### Connections

- **Input BAL:** Boolean value
- **Input BAL REF:** 24-bit integer value (23 bits + sign)
**RAMP**

**Type**

Ramp function

**Summary**

This block is used to limit the rate of change of a signal.

**Illustration**

![Ramp Diagram](image)

**Operation**

The block uses input INPUT as a reference value. The step values (inputs I2 and I3) increase or decrease the output OUT as long as the output differs from limit INPUT. When OUT = INPUT, the output remains steady.

- **Input INPUT**: Reference value
- **Input I2**: Step to positive direction (step/sec). Increase the output, when OUT < INPUT.
- **Input I3**: Step to negative direction (step/sec). Decrease the output, when OUT > INPUT.

\[
\begin{align*}
\text{OUT}_n &= \text{OUT}_{n-1} + I_2 \quad \text{when INPUT} \geq \text{OUT} \\
\text{OUT}_n &= \text{OUT}_{n-1} - I_3 \quad \text{when INPUT} < \text{OUT} \\
\text{OUT}_n &= \text{INPUT} \quad \text{when INPUT} = \text{OUT}
\end{align*}
\]

**Example:**

- **Input INPUT**: 0 -> 150 -> -100 ->0
- **Input I2**: 100 step/sec
- **Input I3**: 10 step/sec

**Output:**

- Going up: Ramp step from Input I2
- Going down: Ramp step from Input I3.

**Connections**

- Inputs INPUT, STEP+ and STEP-: 24-bit integer value (23 bits + sign)
- Output OUT: 24-bit integer (23 bits + sign)
**Application blocks**

<table>
<thead>
<tr>
<th>Type</th>
<th>Logical function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
<td>This block is used as a memory for boolean variables.</td>
</tr>
</tbody>
</table>

**Illustration**

```
    BLOCK x
     SR
      SET
      RESET
      OUT
```

**Operation**

Input SET sets and RESET inputs reset the output.  
If input SET and both RESET inputs are false, the current value remains at the output.  
If input SET is true and both RESET inputs are false, the output is true.  
If one or both of the RESET inputs is true, the output is false.

<table>
<thead>
<tr>
<th>SET</th>
<th>RESET</th>
<th>RESET</th>
<th>OUT (Binary)</th>
<th>OUT (Value on display)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Output</td>
<td>Output</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>True (All bits 1)</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
</tbody>
</table>

**Connections**

Inputs SET and both RESET: Boolean values  
Output OUT: 24-bit integer value (23 bits + sign)
### SWITCH-B

<table>
<thead>
<tr>
<th>Type</th>
<th>Logical function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>This block is used as a changeover switch for boolean type of data.</td>
</tr>
</tbody>
</table>

#### Illustration

![Diagram of SWITCH-B](image)

#### Operation

The output is equal to input NO if input ACT is true and equal to input NC if input ACT is false.

<table>
<thead>
<tr>
<th>ACT</th>
<th>NO</th>
<th>NC</th>
<th>OUT</th>
<th>OUT (Value on display)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NO</td>
<td>NC</td>
<td>NC</td>
<td>True = -1</td>
</tr>
<tr>
<td>1</td>
<td>NO</td>
<td>NC</td>
<td>NO</td>
<td>False = 0</td>
</tr>
</tbody>
</table>

NO = normally open, NC = normally closed

#### Connections

- Input ACT, NO and NC: Boolean values
- Output OUT: 24-bit integer value (packed boolean)

---

### SWITCH-I

<table>
<thead>
<tr>
<th>Type</th>
<th>Logical function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>This block is used as a changeover switch for integer type of data.</td>
</tr>
</tbody>
</table>

#### Illustration

![Diagram of SWITCH-I](image)

#### Operation

The output is equal to input NO if input ACT is true and equal to input NC if input ACT is false.

<table>
<thead>
<tr>
<th>ACT</th>
<th>NO</th>
<th>NC</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NO</td>
<td>NC</td>
<td>NC</td>
</tr>
<tr>
<td>1</td>
<td>NO</td>
<td>NC</td>
<td>NO</td>
</tr>
</tbody>
</table>

NO = normally open, NC = normally closed

#### Connections

- Input ACT: Boolean value
- Input NO and NC: 24-bit integer values (23 bits + sign)
- Output OUT: 24-bit integer value (23 bits + sign)
**TOFF**

<table>
<thead>
<tr>
<th>Type</th>
<th>Timing function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
<td>This block is used for boolean off state delay.</td>
</tr>
</tbody>
</table>

**Illustration**

![Illustration diagram]

**Operation**
The output is true when INPUT is true. The output is false when INPUT has been false for a time equal or longer than TD.

**Connections**

<table>
<thead>
<tr>
<th>Input INPUT:</th>
<th>Boolean value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input TD:</td>
<td>24-bit integer value (23 bits + sign). One corresponds to 1 ms.</td>
</tr>
<tr>
<td>Output OUT:</td>
<td>24-bit integer value (packed boolean)</td>
</tr>
<tr>
<td><strong>TON</strong></td>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td>Timing function</td>
</tr>
</tbody>
</table>

**Illustration**

```
+-----------+----------+----------+
| BLOCK x   | TON      | INPUT    |
|           |          | TD       |
|           |          | OUT      |
```

**Operation**
The output is true when INPUT has been true for a time equal or longer than TD. The output is false when the INPUT is false.

![Diagram](https://via.placeholder.com/150)

Values on display: true = -1, false = 0.

**Connections**

- **INPUT:** Boolean value
- **Input TD:** 24-bit integer value (23 bits + sign). 1 corresponds to 1 ms.
- **Output OUT:** 24-bit integer value (packed boolean)
## TRIGG

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timing function</td>
<td>This block is used for reducing impulse times at the start of automatic procedures and for calculating functions.</td>
</tr>
</tbody>
</table>

### Illustration

![Block Diagram](image)

### Operation

The rising edge of input I1 sets output bit 0 for one program cycle. The rising edge of input I2 sets output bit 1 for one program cycle. The rising edge of input I3 sets output bit 2 for one program cycle.

### Example

- **Input I1**
  - Input 1
  - Output, Bit 0
  - $T_c = $ Program cycle time

![Example Graph](image)

### Connections

- Input I1, I2 and I3: Boolean values
- Output OUT: 24-bit integer value (23 bits + sign)
<table>
<thead>
<tr>
<th>WR-I</th>
<th>Type</th>
<th>This block writes an integer value to the parameter in the RAM memory of the control board.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Illustration</td>
<td><strong>BLOCK x</strong>&lt;br&gt;<strong>WR-I</strong>&lt;br&gt;<strong>GROUP</strong>&lt;br&gt;<strong>INDEX</strong>&lt;br&gt;<strong>OUT</strong>&lt;br&gt;<strong>IN</strong></td>
</tr>
</tbody>
</table>
|      | Operation | This function writes an integer value to the integer type of AMC table index.  
**Note:** The function does not consider whether another device e.g. fieldbus is writing into the same location, which would cause oscillation of signal. It is not possible to write into the middle of the reference chain.  
See the **Firmware manual** for the parameter type (I or PB). |
|      | Connections | Inputs **GROUP**, **INDEX** and **IN**: 24-bit integer value (23 bits + sign)  
Input **GROUP**: Parameter group number  
Input **INDEX**: Parameter index number  
Input **IN**: Data input pin to read a new value for the parameter  
Output **OUT**: Error code (24-bit integer value)  
**Error codes:**  
0 Successful write  
131073 Group protected  
131074 Index protected  
131075 Illegal group  
131076 Undefined group  
131077 Illegal index  
131078 Undefined index  
131079 Illegal format  
131080 Min max limitation  
131088 Illegal selection  
**Example:** For parameter **24.01 Q POWER REF**, **GROUP** is 24 and **INDEX** is 01.
<table>
<thead>
<tr>
<th><strong>WR-PB</strong></th>
<th><strong>Type</strong></th>
<th>This block writes a packed boolean value to the parameter in the RAM memory of the control board.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Illustration</strong></td>
<td>BLOCK x&lt;br&gt;WR-PB&lt;br&gt;GROUP&lt;br&gt;INDEX&lt;br&gt;IN&lt;br&gt;OUT</td>
<td>Writes a packed boolean value to the packed boolean type of AMC table index e.g. command word. <strong>Note:</strong> The function does not consider whether another device e.g. fieldbus is writing into the same location, which would cause oscillation of signal. See the Firmware manual for the parameter type (l or PB).</td>
</tr>
<tr>
<td><strong>Operation</strong></td>
<td>Writes a packed boolean value to the packed boolean type of AMC table index e.g. command word.</td>
<td></td>
</tr>
<tr>
<td><strong>Connections</strong></td>
<td>Inputs GROUP, INDEX and IN: 24-bit integer value (23 bits + sign)&lt;br&gt;Input GROUP: Parameter group number&lt;br&gt;Input INDEX: Parameter index number&lt;br&gt;Input IN: Data input pin to read A new value for the parameter&lt;br&gt;Output OUT: Error code (24-bit integer value)&lt;br&gt;Error codes:&lt;br&gt;0 Successful write&lt;br&gt;131073 Group protected&lt;br&gt;131074 Index protected&lt;br&gt;131075 Illegal group&lt;br&gt;131076 Undefined group&lt;br&gt;131077 Illegal index&lt;br&gt;131078 Undefined index&lt;br&gt;131079 Illegal format&lt;br&gt;131080 Min max limitation&lt;br&gt;131088 Illegal selection&lt;br&gt;<strong>Example:</strong> For parameter 07.01 MAIN CTRL WORD, GROUP is 7 and INDEX is 01.</td>
<td></td>
</tr>
</tbody>
</table>
**XOR**

**Summary**
This block is used to generate combinatory expressions with boolean variables.

**Illustration**

```
BLOCK x
  XOR
  I1
  I2
  I3
    OUT
```

**Operation**
The output is true if only one or all connected inputs are true. Otherwise, the output is false. The truth table is shown below.

<table>
<thead>
<tr>
<th>I1</th>
<th>I2</th>
<th>I3</th>
<th>OUT (Binary)</th>
<th>OUT (Value on display)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>True (All bits 1)</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>True (All bits 1)</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>True (All bits 1)</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>False (All bits 0)</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>True (All bits 1)</td>
<td>-1</td>
</tr>
</tbody>
</table>

**Connections**
Input I1, I2 and I3: Boolean values
Output OUT: 24-bit integer value (packed boolean)
I/O and communication blocks

Execution time of these blocks has no relation with execution time of function blocks.

<table>
<thead>
<tr>
<th>A/F WORD</th>
<th>Type</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/F WORD</td>
<td>Application-based alarm and fault word. Execution interval is 560 ms.</td>
<td></td>
</tr>
</tbody>
</table>

**Illustration**

![A/F WORD Diagram]

**Operation**
This block is used to compile application specific alarms and faults into a packed boolean word further to be read e.g. by an overriding system from **09.13 AP FW**.

Use e.g. BSET function blocks to set the desired bits of A/F WORD block according to the application needs.

**Connections**
- IN: 24-bit integer value (23 bits + sign)

<table>
<thead>
<tr>
<th>AI1</th>
<th>Type</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI1</td>
<td>Analog input AI1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This block is used to read analog input AI1 of the control unit. The resolution is 10 bits + sign, voltage type of input. The updating interval is 10 ms.</td>
<td></td>
</tr>
</tbody>
</table>

**Illustration**

![AI1 Diagram]

**Operation**
- CONV MODE: See parameter **13.01 AI1 CONV MODE**.
- FILTER ms: See parameter **13.02 AI1 FILTER ms**. A filter time constant for AI1.
- OUT: See parameter **05.01 BASIC AI1**.

**Connections**
- Input CONV MODE: Integer value 1…3
- Input FILTER ms: Integer value 0…30000
- Output: Integer value on range -20000…20000
<table>
<thead>
<tr>
<th><strong>AI2</strong></th>
<th><strong>Type</strong></th>
<th>Analog input AI2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
<td>This block is used to read analog input AI2 of the control unit. Resolution is 10 bits + sign, current type of the input is 0(4)…20 mA.</td>
<td></td>
</tr>
<tr>
<td><strong>Illustration</strong></td>
<td><img src="image" alt="Illustration" /></td>
<td></td>
</tr>
<tr>
<td><strong>Operation</strong></td>
<td>CONV MODE: See parameter 13.03 AI2 CONV MODE. FILTER ms: See parameter 13.04 AI2 FILTER ms. OUT: See signal 05.01 BASIC AI1.</td>
<td></td>
</tr>
<tr>
<td><strong>Connections</strong></td>
<td>Input CONV MODE: Integer value 1…2 Input FILTER ms: Integer value 0…30000 Output: Integer value on range -20000…20000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>AI3</strong></th>
<th><strong>Type</strong></th>
<th>Analog input AI3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
<td>This block is used to read analog input AI3 of the control unit. Resolution is 10 bits + sign, current type of the input is 0(4)…20 mA.</td>
<td></td>
</tr>
<tr>
<td><strong>Illustration</strong></td>
<td><img src="image" alt="Illustration" /></td>
<td></td>
</tr>
<tr>
<td><strong>Operation</strong></td>
<td>CONV MODE: See parameter 13.05 AI3 CONV MODE. FILTER ms: See parameter 13.06 AI3 FILTER ms. OUT: See signal 05.03 BASIC AI3.</td>
<td></td>
</tr>
<tr>
<td><strong>Connections</strong></td>
<td>Input CONV MODE: Integer value 1…2 Input FILTER ms: Integer value 0…30000 Output: Integer value on range -20000…20000</td>
<td></td>
</tr>
</tbody>
</table>
### AO1

**Type** | Analog output AO1  
---|---
**Summary** | This block is used to write data to analog output AO1 of the control unit. Execution interval is 10 ms.

**Illustration**

![AO1 Illustration](image)

**Operation**

- OFFSET: See parameter 15.01 AO1 OFFSET.
- SCALE: See parameter 15.02 AO1 SCALE.
- IN: Input pin to control analog output AO1.

**Connections**

- Input OFFSET: Integer value 0…20000 = 0…20 mA  
- Input SCALE: Integer value 0…30000  
- IN: 24-bit integer value (23 bits + sign)

---

### AO2

**Type** | Analog output AO2  
---|---
**Summary** | This block is used to write data to analog output AO2 of the control unit. Execution interval is 10 ms.

**Illustration**

![AO2 Illustration](image)

**Operation**

- OFFSET: See parameter 15.03 AO2 OFFSET.  
- SCALE: See parameter 15.04 AO2 SCALE.  
- IN: Input pin to control analog output AO2.

**Connections**

- Input OFFSET: Integer value 0…20000 = 0…20 mA  
- Input SCALE: Integer value 0…30000  
- IN: 24-bit integer value (23 bits + sign)
## CW

<table>
<thead>
<tr>
<th>Type</th>
<th>Control Word in adaptive program. This block is used to control the PVS800 from the adaptive program. See parameter 07.02 USED MCW in the Firmware manual. Execution interval is 10 ms.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Illustration</td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>See parameter 66.01 CW in the Firmware manual.</td>
</tr>
<tr>
<td>Connections</td>
<td>IN: 24-bit integer value (23 bits + sign)</td>
</tr>
</tbody>
</table>

## DI1...DI6, DI IL

<table>
<thead>
<tr>
<th>Type</th>
<th>Digital input These blocks are used to read the status of digital inputs DI1...DI6 and DIIL (DI7) of the control unit. Execution interval is 10 ms.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Illustration</td>
<td></td>
</tr>
</tbody>
</table>
| Operation | Output of the block corresponds to 08.02 DI STATUS WORD.  
bit 0 = not in use  
bit 1 = status of DI1  
bit 2 = status of DI2  
bit 3 = status of DI3  
bit 4 = status of DI4  
bit 5 = status of DI5  
bit 6 = status of DI6  
bit 7 = status of DIIL |
| Connections | OUT: 16-bit packed boolean value.                                                                                                                                                                   |
### DO1
**Type**
Digital output DO1

**Summary**
This block is used to control relay output RO1 of the control unit.
Execution interval is 10 ms.

**Illustration**

![DO1 Block Diagram](image1)

**Operation**
IN: State TRUE energises relay output RO1 and state FALSE de-energises the relay.

**Connections**
IN: 24-bit integer value (23 bits + sign).

### DO2
**Type**
Digital output DO2

**Summary**
This block is used to control relay output RO2 of the control unit.
Execution interval is 10 ms.

**Illustration**

![DO2 Block Diagram](image2)

**Operation**
IN: State TRUE energises relay output RO2 and state FALSE de-energises the relay.

**Connections**
IN: 24-bit integer value (23 bits + sign).

### DO3
**Type**
Digital output DO3

**Summary**
This block is used to control relay output RO3 of the control unit.
Execution interval is 10 ms.

**Illustration**

![DO3 Block Diagram](image3)

**Operation**
IN: State TRUE energises relay output RO3 and state FALSE de-energises the relay.

**Connections**
IN: 24-bit integer value (23 bits + sign).
<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EXT1</strong></td>
<td><strong>DI…DI3</strong> Digital inputs</td>
</tr>
<tr>
<td></td>
<td>This block is used to read the status of digital inputs DI1…DI3 of digital I/O extension module 1.</td>
</tr>
<tr>
<td></td>
<td>Execution interval is 100 ms.</td>
</tr>
<tr>
<td><strong>Illustration</strong></td>
<td><img src="#" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>Operation</strong></td>
<td>Output equals <strong>08.03 EXT DI STATUS WOR</strong> bits 1…3.</td>
</tr>
<tr>
<td><strong>Connections</strong></td>
<td>OUT: 16-bit packed boolean value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EXT2</strong></td>
<td><strong>DI…DI3</strong> Digital inputs</td>
</tr>
<tr>
<td></td>
<td>This block is used to read the status of digital inputs DI1…DI3 of digital I/O extension module 2.</td>
</tr>
<tr>
<td></td>
<td>Execution interval is 100 ms.</td>
</tr>
<tr>
<td><strong>Illustration</strong></td>
<td><img src="#" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>Operation</strong></td>
<td>Output equals <strong>08.03 EXT DI STATUS WOR</strong> bits 4…6.</td>
</tr>
<tr>
<td><strong>Connections</strong></td>
<td>OUT: 16-bit packed boolean value.</td>
</tr>
</tbody>
</table>
### EXT3 DI1...DI3

| Type Summary | Extension module 3 digital inputs  
|--------------|--------------------------------------------------------------------------------|
|              | This block is used to read the status of digital inputs DI1...DI3 of digital I/O extension module 3.  
|              | Execution interval is 100 ms.  

#### Illustration

![Diagram](image)

#### Operation

Output is the same as **08.03 EXT DI STATUS WOR** bits 7...9.

#### Connections

**OUT:** 16-bit packed boolean value.

### EXT4 DI1...DI3

| Type Summary | Extension module 4 digital inputs  
|--------------|--------------------------------------------------------------------------------|
|              | This block is used to read the status of digital inputs DI1...DI3 of digital I/O extension module 4.  
|              | Execution interval is 100 ms.  

#### Illustration

![Diagram](image)

#### Operation

Output is the same as **08.03 EXT DI STATUS WOR** bits 10...12.

#### Connections

**OUT:** 16-bit packed boolean value.
### EXT5 DI...DI3

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extension module 5 digital inputs</td>
<td></td>
</tr>
<tr>
<td>This block is used to read the status of digital inputs DI1...DI3 of digital I/O extension module 5.</td>
<td></td>
</tr>
<tr>
<td>Execution interval is 100 ms.</td>
<td></td>
</tr>
</tbody>
</table>

#### Illustration

![Diagram](image)

#### Operation

Output is the same as **08.03 EXT DI STATUS WOR** bits 13...15.

#### Connections

**OUT:** 16-bit packed boolean value.

### EXT DO

<table>
<thead>
<tr>
<th>Type</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extension module digital output control</td>
<td></td>
</tr>
<tr>
<td>This block is used to control the relay outputs of digital I/O extension modules 1...5.</td>
<td></td>
</tr>
<tr>
<td>Execution interval is 100 ms.</td>
<td></td>
</tr>
</tbody>
</table>

#### Illustration

![Diagram](image)

#### Operation

This block writes packed boolean value into parameter **66.05 EXT DO**. State TRUE of each bit energises relay output and state FALSE de-energises the relay. Use **BSET** blocks connected in series to set each bit of parameter **66.05 EXT DO**.

#### Connections

**IN:** 16-bit packed boolean value.
<table>
<thead>
<tr>
<th><strong>EXT1...5 AI1...AI2</strong></th>
<th><strong>Type</strong></th>
<th>Extension module analogue inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
<td>This block is used to read analogue inputs AI1 and AI2 of analogue I/O extension modules 1… 5. Execution interval is 100 ms.</td>
<td></td>
</tr>
<tr>
<td><strong>Illustration</strong></td>
<td>Example: EXT1 AI1 and EXT1 AI2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>![Diagram](Note: Diagram not provided in text format.)</td>
<td></td>
</tr>
<tr>
<td><strong>Operation</strong></td>
<td>HW MODE: See parameter 13.15 EXT1 AI1 HW MODE.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CONV MODE: See parameter 13.17 EXT1 AI1 CONV MOD.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FILTER ms See parameter 13.19 EXT1 AI1 FILT ms.</td>
<td></td>
</tr>
<tr>
<td><strong>Connections</strong></td>
<td>Input HW MODE: Integer value 1…2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Input CONV MODE: Integer value 1…4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Input FILTER ms: Integer value 0…30000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Output OUT: Integer value on range –20000…20000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hardware settings See Analogue I/O Extension User’s Manual RAIO-01 (3AFE64484567 English).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>EXT1...5 AO1...AO2</strong></th>
<th><strong>Type</strong></th>
<th>Extension module analogue outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
<td>This block is used to write data to analogue output AO1 and AO2 of the analogue I/O extension modules 1…5. Execution interval is 100 ms.</td>
<td></td>
</tr>
<tr>
<td><strong>Illustration</strong></td>
<td>Example: EXT5 AO1 and EXT5 AO2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>![Diagram](Note: Diagram not provided in text format.)</td>
<td></td>
</tr>
<tr>
<td><strong>Operation</strong></td>
<td>OFFSET: See parameter 15.05 EXT1 AO1 OFFSET.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCALE: See parameters 15.06 EXT1 AO1 SCALE.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IN: Input pin to control analogue output</td>
<td></td>
</tr>
<tr>
<td><strong>Connections</strong></td>
<td>Input OFFSET: integer value 0…20000 = 0…20 mA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Input SCALE: integer value 0…30000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IN: 24-bit integer value (23 bits + sign)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hardware settings See Analogue I/O Extension User’s Manual RAIO-01 (3AFE64484567 English).</td>
<td></td>
</tr>
</tbody>
</table>
### Application blocks

<table>
<thead>
<tr>
<th><strong>FUNG IN</strong></th>
<th><strong>Type</strong></th>
<th><strong>Summary</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Function generator input</td>
<td>This block is a part of an internally built 5-step function generation. It is executed every 100 ms.</td>
</tr>
</tbody>
</table>

**Illustration**

![FUNG IN Diagram](image)

**Operation**

Input for function generator. See parameter group **65 FUNCTION GENERATOR**.

Function generator is enabled by parameter **65.01 ENABLE** and it includes:

**IN:** Function block **FUNG IN**

A function curve is set by parameters:

- **65.04** X1
- **65.05** Y1
- **65.06** X2
- **65.07** Y2
- **65.08** X3
- **65.09** Y3
- **65.10** X4
- **65.11** Y4
- **65.12** X5
- **65.13** Y5

**OUT:** Function block **FUNG OUT**

**Connections**

**IN:** 24-bit integer value (23 bits + sign)

<table>
<thead>
<tr>
<th><strong>FUNG OUT</strong></th>
<th><strong>Type</strong></th>
<th><strong>Summary</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Function generator output</td>
<td>This block is a part of internally built 5 step function generation. It is executed every 100 ms.</td>
</tr>
</tbody>
</table>

**Illustration**

![FUNG OUT Diagram](image)

**Operation**

Output of the function generator. See parameter group **65 FUNCTION GENERATOR**.

Function generator is enabled by parameter **65.01 ENABLE** and it includes:

**IN:** Function block **FUNG IN**

A function curve is set by parameters:

- **65.04** X1
- **65.05** Y1
- **65.06** X2
- **65.07** Y2
- **65.08** X3
- **65.09** Y3
- **65.10** X4
- **65.11** Y4
- **65.12** X5
- **65.13** Y5

**OUT:** Function block **FUNG OUT**

**Connections**

**OUT:** 24-bit integer value (23 bits + sign)
**PZD3 OUT**

<table>
<thead>
<tr>
<th><strong>Type</strong></th>
<th>Communication input block</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
<td>This block is used with RPBA fieldbus module to read PROFIBUS process data PZD3 OUT for block application program.</td>
</tr>
</tbody>
</table>

**Illustration**

![Diagram of PZD3 OUT block](image_url)

**Operation**

When this block is inserted, it automatically sets parameter 51.05 PZD3 OUT to 1901. Thereafter, data PZD3 OUT received from the master device is written to parameter 19.01 DATA 1 (output pin of the block). When this block is deleted, it automatically writes zero to parameter 19.01 DATA 1 once. If this block is not activated, parameter 19.01 DATA 1 can be used for other purposes.

In on-line mode PZD3 OUT … PZD10 OUT blocks are available only if RPBA module is connected to Slot 1. In off-line mode, use the PROFIBUS template file.

**Connections**

Output OUT: 16-bit integer value (15 bits + sign)
### PZD4 OUT

<table>
<thead>
<tr>
<th>Type</th>
<th>Communication input block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>This block is used with RPBA fieldbus module to read PROFIBUS process data PZD4 OUT for block application program.</td>
</tr>
</tbody>
</table>

#### Illustration

![Diagram of PZD4 OUT]

#### Operation

When this block is inserted, it automatically sets parameter 51.07 PZD4 OUT to 1902. Thereafter, data PZD4 OUT received from the master device is written to parameter 19.02 DATA 2 (output pin of the block). When this block is deleted, it automatically writes zero to parameter 19.02 DATA 2 once. If this block is not activated, parameter 19.02 DATA 2 can be used for other purposes.

In on-line mode PZD3 OUT … PZD10 OUT blocks are available only if RPBA module is connected to Slot 1. In off-line mode, use the PROFIBUS template file.

#### Connections

- **OUT**: 16-bit integer value (15 bits + sign)

### PZD5 OUT

<table>
<thead>
<tr>
<th>Type</th>
<th>Communication input block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>This block is used with RPBA fieldbus module to read PROFIBUS data PZD5 OUT for block application program.</td>
</tr>
</tbody>
</table>

#### Illustration

![Diagram of PZD5 OUT]

#### Operation

When this block is inserted, it automatically sets parameter 51.09 PZD5 OUT to 1903. Thereafter, data PZD5 OUT received from the master device is written to parameter 19.03 DATA 3 (output pin of the block). When this block is deleted, it automatically writes zero to parameter 19.03 DATA 3 once. If this block is not activated, parameter 19.03 DATA 3 can be used for other purposes.

In on-line mode PZD3 OUT … PZD10 OUT blocks are available only if RPBA module is connected to Slot 1. In off-line mode, use the PROFIBUS template file.

#### Connections

- **Output OUT**: 16-bit integer value (15 bits + sign)
**PZD6 OUT**

<table>
<thead>
<tr>
<th>Type</th>
<th>Communication input block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>This block is used with RPBA fieldbus module to read PROFIBUS process data PZD6 OUT for block application program.</td>
</tr>
</tbody>
</table>

**Illustration**

![Diagram of PZD6 OUT block]

**Operation**

When this block is inserted, it automatically sets parameter 51.11 PZD6 OUT to 1904. Thereafter, data PZD6 OUT received from the master device is written to parameter 19.04 DATA 4 (output pin of the block). When this block is deleted, it automatically writes zero to parameter 19.04 DATA 4 once. If this block is not activated, parameter 19.04 DATA 4 can be used for other purposes.

In on-line mode PZD3 OUT … PZD10 OUT blocks are available only if RPBA module is connected to Slot 1. In off-line mode, use the PROFIBUS template file.

**Connections**

OUT: 16-bit integer value (15 bits + sign)

---

**PZD7 OUT**

<table>
<thead>
<tr>
<th>Type</th>
<th>Communication input block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>This block is used with RPBA fieldbus module to read PROFIBUS process data PZD7 OUT for block application program.</td>
</tr>
</tbody>
</table>

**Illustration**

![Diagram of PZD7 OUT block]

**Operation**

When this block is inserted, it automatically sets parameter 51.13 PZD7 OUT to 1905. Thereafter, data PZD7 OUT received from the master device is written to parameter 19.05 DATA 5 (output pin of the block). When this block is deleted, it automatically writes zero to parameter 19.05 DATA 5 once. If this block is not activated, parameter 19.05 DATA 5 can be used for other purposes.

In on-line mode PZD3 OUT … PZD10 OUT blocks are available only if RPBA module is connected to Slot 1. In off-line mode, use the PROFIBUS template file.

**Connections**

OUT: 16-bit integer value (15 bits + sign)
### PZD8 OUT

<table>
<thead>
<tr>
<th>Type</th>
<th>Communication input block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>This block is used with RPBA fieldbus module to read PROFIBUS process data PZD8 OUT for block application program.</td>
</tr>
</tbody>
</table>

**Illustration**

![Diagram of PZD8 OUT block](image)

**Operation**

When this block is inserted, it automatically sets parameter 51.15 PZD8 OUT to 1906. Thereafter, data PZD8 OUT received from the master device is written to parameter 19.06 DATA 6 (output pin of the block). When this block is deleted, it automatically writes zero to parameter 19.06 DATA 6 once. If this block is not activated, parameter 19.06 DATA 6 can be used for other purposes.

In on-line mode PZD3 OUT … PZD10 OUT blocks are available only if RPBA module is connected to Slot 1. In off-line mode, use the PROFIBUS template file.

**Connections**

| OUT: | 16-bit integer value (15 bits + sign) |

### PZD9 OUT

<table>
<thead>
<tr>
<th>Type</th>
<th>Communication input block</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>This block is used with RPBA fieldbus module to read PROFIBUS process data PZD9 OUT for block application program. Execution interval is 20 ms.</td>
</tr>
</tbody>
</table>

**Illustration**

![Diagram of PZD9 OUT block](image)

**Operation**

When this block is inserted, it automatically sets parameter 51.17 PZD9 OUT to 1907. Thereafter, data PZD9 OUT received from the master device is written to parameter 19.07 DATA 7 (output pin of the block). When this block is deleted, it automatically writes zero to parameter 19.07 DATA 7 once. If this block is not activated, parameter 19.07 DATA 7 can be used for other purposes.

In on-line mode PZD3 OUT … PZD10 OUT blocks are available only if RPBA module is connected to Slot 1. In off-line mode, use the PROFIBUS template file.

**Connections**

| OUT: | 16-bit integer value (15 bits + sign) |
### PZD10 OUT

**Type**
Communication input block.

**Summary**
This block is used with RPBA fieldbus module to read PROFIBUS process data PZD10 OUT for block application program.

**Illustration**

![Diagram of PZD10 OUT]

**Operation**
When this block is inserted, it automatically sets parameter 51.19 PZD10 OUT to 1908. Thereafter, data PZD10 OUT received from the master device is written to parameter 19.08 DATA 8 (output pin of the block). When this block is deleted, it automatically writes zero to parameter 19.08 DATA 8 once. If this block is not activated, parameter 19.08 DATA8 can be used for other purposes.

In on-line mode PZD3 OUT … PZD10 OUT blocks are available only if RPBA module is connected to Slot 1. In off-line mode, use the PROFIBUS template file.

**Connections**
- OUT: 16-bit integer value (15 bits + sign)

### PZD3 IN

**Type**
Communication output block from the PVS800.

**Summary**
This block is used with RPBA fieldbus module to write process data value PZD3 IN to the PROFIBUS master device.

**Illustration**

![Diagram of PZD3 IN]

**Operation**
INPUT: Source signal is connected to INPUT and sent to the PROFIBUS master. Assignment of source signal is written automatically to parameter 51.06 PZD3 IN during the input connection.

**Note:** The maximum index number to Profibus is 99. Therefore, use only blocks 1…19 as a source from the block application to this block.

**Note:** Function block outputs are 24-bit integer values and PROFIBUS outputs 16-bit integer values. In on-line mode PZD3 IN … PZD10 IN blocks are available only if RPBA module is connected to Slot 1. In off-line mode, use the PROFIBUS template file.

**Connections**
- INPUT: 16-bit integer value (15 bits + sign)
### PZD4 IN

**Type**

Communication output block from the PVS800.

**Summary**

This block is used with RPBA type of fieldbus module to write process data value PZD4 IN to the PROFIBUS master device.

**Illustration**

Block 1…19

![Diagram](image1)

**Operation**

- **INPUT**: Source signal is connected to INPUT and sent to the PROFIBUS master. Assignment of source signal is written automatically to parameter 51.08 PZD4 IN during the input connection.
- **Note**: The maximum index number to PROFIBUS is 99. Therefore, use only blocks 1…19 as a source from the block application to this block.
- **Note**: Function block outputs are 24-bit integer values and PROFIBUS outputs 16-bit integer values. In on-line mode PZD3 IN … PZD10 IN blocks are available only if RPBA module is connected to Slot 1. In off-line mode, use the PROFIBUS template file.

**Connections**

- **INPUT**: 16-bit integer value (15 bits + sign)

### PZD5 IN

**Type**

Communication output block from the PVS800.

**Summary**

This block is used with RPBA type of fieldbus module to write process data value PZD5 IN to the Proﬁbus master device.

**Illustration**

Block 1…19

![Diagram](image2)

**Operation**

- **INPUT**: Source signal is connected to INPUT and sent to the PROFIBUS master. Assignment of source signal is written automatically to parameter 51.10 PZD5 IN during the input connection.
- **Note**: The maximum index number to PROFIBUS is 99. Therefore, use only blocks 1…19 as a source from the block application to this block.
- **Note**: Function block outputs are 24-bit integer values and PROFIBUS outputs 16-bit integer values. In on-line mode PZD3 IN … PZD10 IN blocks are available only if RPBA module is connected to Slot 1. In off-line mode, use the PROFIBUS template file.

**Connections**

- **INPUT**: 16-bit integer value (15 bits + sign)
PZD6 IN

Type
Summary
Communication output block from the PVS800.
This block is used with RPBA type of fieldbus module to write process data value PZD6 IN to the PROFIBUS master device.

Illustration
Block 1…19

Operation
INPUT: Source signal is connected to INPUT and sent to the PROFIBUS master. Assignment of source signal is written automatically to parameter 51.12 PZD6 IN during the input connection.

Note: The maximum index number to PROFIBUS is 99. Therefore, use only blocks 1…19 as a source from the block application to this block.

Note: Function block outputs are 24-bit integer values and PROFIBUS outputs 16-bit integer values. In on-line mode PZD3 IN … PZD10 IN blocks are available only if RPBA module is connected to Slot 1. In off-line mode, use the PROFIBUS template file.

Connections
INPUT: 16-bit integer value (15 bits + sign)

PZD7 IN

Type
Summary
Communication output block from the PVS800.
This block is used with RPBA type of fieldbus module to write process data value PZD7 IN to the PROFIBUS master device.

Illustration
Block 1…19

Operation
INPUT: Source signal is connected to INPUT and sent to the PROFIBUS master. Assignment of source signal is written automatically to parameter 51.14 PZD7 IN during the input connection.

Note: The maximum index number to PROFIBUS is 99. Therefore, use only blocks 1…19 as a source from the block application to this block.

Note: Function block outputs are 24-bit integer values and PROFIBUS outputs 16-bit integer values. In on-line mode PZD3 IN … PZD10 IN blocks are available only if RPBA module is connected to Slot 1. In off-line mode, use the PROFIBUS template file.

Connections
INPUT: 16-bit integer value (15 bits + sign)
## PZD8 IN

### Type
Communication output block from the PVS800.

### Summary
This block is used with RPBA type of fieldbus module to write process data value PZD8 IN to the PROFIBUS master device.

### Illustration
![Diagram of PZD8 IN block](image)

### Operation
**INPUT:** Source signal is connected to INPUT and sent to the PROFIBUS master. Assignment of source signal is written automatically to parameter 51.16 PZD8 IN during the input connection.

**Note:** The maximum index number to PROFIBUS is 99. Therefore, use only blocks 1…19 as a source from the block application to this block.

**Note:** Function block outputs are 24-bit integer values and PROFIBUS outputs 16-bit integer values. In on-line mode PZD3 IN … PZD10 IN blocks are available only if RPBA module is connected to Slot 1. In off-line mode, use the PROFIBUS template file.

### Connections
**INPUT:** 16-bit integer value (15 bits + sign)

## PZD9 IN

### Type
Communication output block from the PVS800.

### Summary
This block is used with RPBA type of fieldbus module to write process data value PZD9 IN to the PROFIBUS master device.

### Illustration
![Diagram of PZD9 IN block](image)

### Operation
**INPUT:** Source signal is connected to INPUT and sent to the PROFIBUS master. Assignment of source signal is written automatically to parameter 51.18 PZD9 IN during the input connection.

**Note:** The maximum index number to PROFIBUS is 99. Therefore, use only blocks 1…19 as a source from the block application to this block.

**Note:** Function block outputs are 24-bit integer values and PROFIBUS outputs 16-bit integer values. In on-line mode PZD3 IN … PZD10 IN blocks are available only if RPBA module is connected to Slot 1. In off-line mode, use the PROFIBUS template file.

### Connections
**INPUT:** 16-bit integer value (15 bits + sign)
**PZD10 IN**

**Type** Communication output block from the PVS800.

**Summary** This block is used with RPBA type of fieldbus module to write process data value PZD10 IN to the PROFIBUS master device.

**Illustration** Block 1…19

**Operation** INPUT: Source signal is connected to INPUT and sent to the PROFIBUS master. Assignment of source signal is written automatically to parameter 51.20 PZD10 IN during the input connection.

**Note:** The maximum index number to PROFIBUS is 99. Therefore, use only blocks 1…19 as a source from the block application to this block.

**Note:** Function block outputs are 24-bit integer values and PEOFIBUS outputs 16-bit integer values. In on-line mode PZD3 IN … PZD10 IN blocks are available only if RPBA module is connected to Slot 1. In off-line mode, use the PROFIBUS template file.

**Connections**

| INPUT | 16-bit integer value (15 bits + sign) |

---

**SUB**

**Type** Arithmetic subtraction function.

**Summary**

**Illustration**

**Operation** Subtraction can be implemented by using ADD block in which the subtracter input is inverted (multiplied by -1).

\[ \text{OUT} = \text{ADD1} + (-\text{ADD2}) + (-\text{ADD3}) \]

**Example**
- \( \text{ADD1} = 3000 \)
- \( \text{ADD2} = 1000 \) (inverted in Connect Pin window of DriveAP)
- \( \text{ADD3} = 0 \) (inverted in Connect Pin window of DriveAP)

\[ \text{OUT} = 3000 + (-1000) + 0 \]

2000 = 3000 + (-1000) + (-0)

**Connections**

| Inputs ADD1, ADD2 and ADD3: | 24-bit integer values (23 bits + sign) |
| Output OUT: | 24-bit integer (23 bits + sign) |
Further information

More information about ABB products for solar applications on the Internet:
www.abb.com/solar