*502 060 02*

# UNIVERSAL PROCESS INTERFACE

# PD 3221

# Manual

*GB* *July 1995*

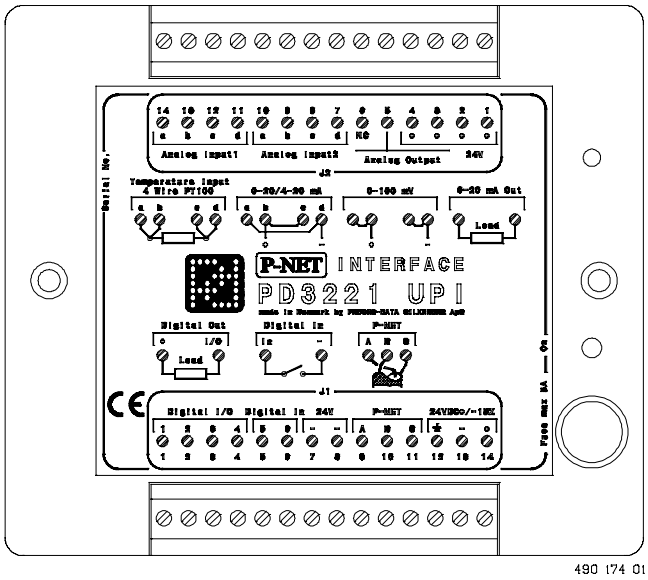
**Proces-Data Silkeborg ApS, Navervej 10, DK-8600 Silkeborg, Denmark, phone +45-87200300, fax +45-87200301**

## Contents

# 1 General information.

The PD 3221 Universal Process Interface is a member of **Proces-Data**'s module series 3000.

The PD 3221 module is equipped with an interface to P-NET®, which is a field-bus network designed for process control and data-collection. The PD 3221 module has been developed for interfacing directly to process signals. Configuration of the module for the functions required, and communication between the module and a control computer, is carried out via the P-NET. The PD 3221 module can be controlled via the P-NET, or it can operate as an autonomous unit.




## 1.1 Features.

* 4 Digital I/O's at 24 V 1 Amp, and 2 digital input.

* 2 high resolution (14 bit) analog input channels. Each channel can measure voltage (0-100 mV), current (0-20 or 4-20 mA) or derive temperature from a Pt-100 detector. The input signals are filtered, including 50 and 60 Hz interference suppression.

* Analog 4-20 mA current signal generation, with No-Load detection.

* Internal PID-regulator.

* Pulse Processor for fast pulse counting and pulse generation.

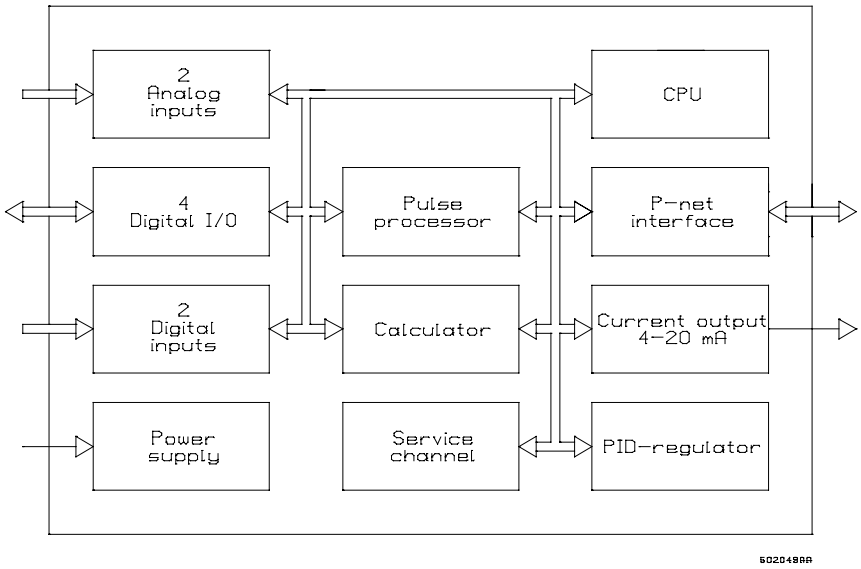
* 1 internal calculator channel.

* Advanced self testing facility, which can be monitored through the P-NET.

* All inputs and outputs are overload protected.

## 1.2 Channels/registers.

The PD 3221 module contains:




| | | | |
|---|---|---|---|
| 1 Service channel | (channel 0) | 4 Digital I/O‘s | (channel 1-4) |
| 2 Digital inputs | (channel 5-6) | 1 Common I/O channel | (channel 7) |
| 2 Analog inputs | (channel 8-9) | 1 Current output | (channel A) |
| 1 PID-regulator | (channel B) | 1 Calculator channel | (channel C) |
| 1 Pulse Processor | (channel D) | | |

A set of 16 variables, numbered from 0 - $F, is associated with each channel. For addressing a variable within a particular channel, a logical address called a SoftWire Number (SWNo), is used. The SWNo is calculated as:(channel number ∗ $10 + variable number within the channel).

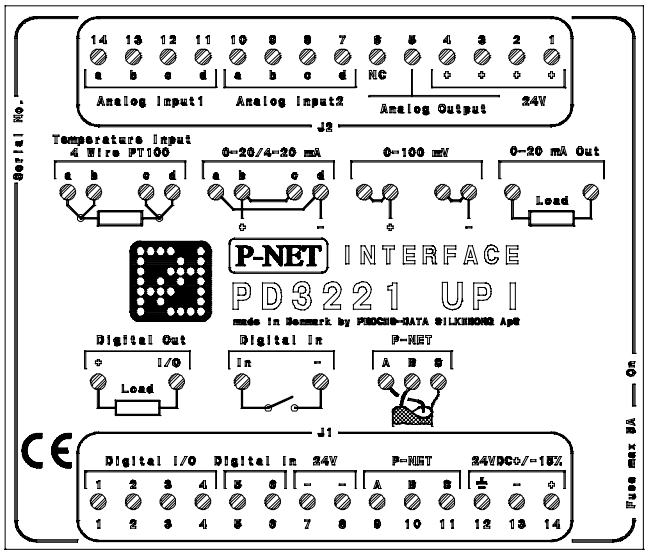
Example: Variable 4 on channel 3 needs to be addressed.
The SWNo will therefore be $34.

Throughout the manual the variables are depicted as tables.
Variables marked with a @ are called "indirect" variables, and can be configured as a normal variable or as an indirect variable. The indirect variable function is selected by pointing to another variable, by inserting the SWNo of this variable into ChConfig. If SWNo 00 is inserted, the variable becomes a normal independent variable. Reading or writing to an indirect variable is the same as reading or writing directly to the pointed variable. An indirect variable must not point at another indirect variable.

The variable names are standard identifiers, as defined in Process-Pascal.

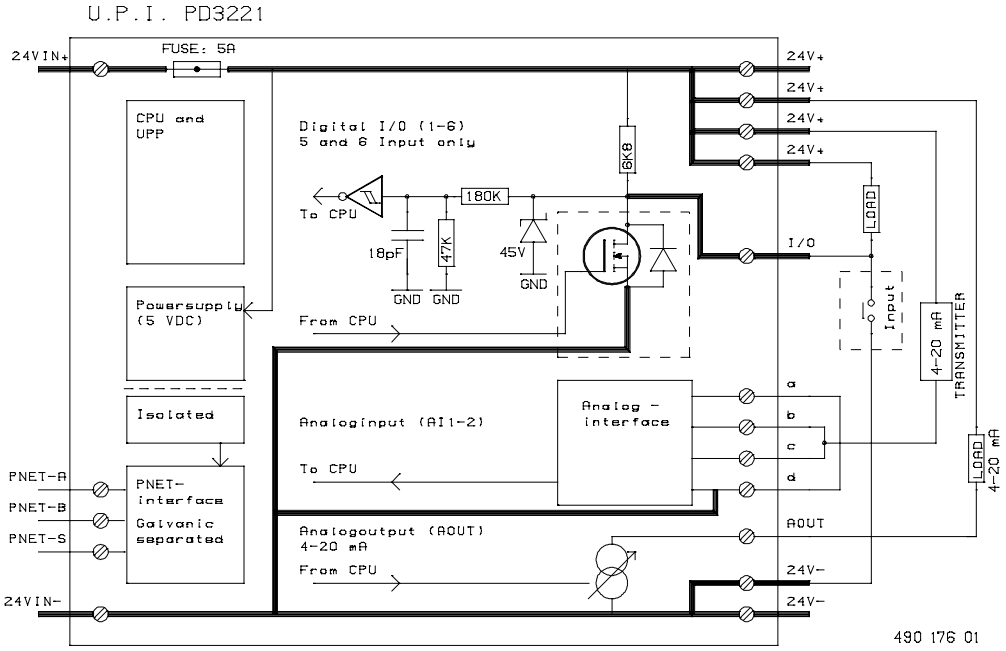
## 1.3 Connections.

The PD 3221 UPI is physically designed as a black box, having two 14 pin connectors for screw terminals. The connectors are removable and equipped with a key pin, to avoid reversed connections. The module has a built in fuse, which is used to protect the module, and externally connected wiring and equipment. When connecting external equipment, the additional fuse protected +24 V screw terminals should be used (see hardware diagram). The module wiring should be designed with a maximum of 2 wire connections in each screw terminal. Connection identities are printed on the top of the module.




490 175 01

**Hardware diagram, principle.**




490 176 01

## 1.4 Memory types.

The PD 3221 stores data in different types of memory depending on the value of a control variable following a reset or a power failure, and the state of write protection.

Some variables are stored in both non volatile memory and in volatile memory. The state of the module's WriteEnable register determines whether the contents are changed in both types of memory or only in the volatile type.

The following memory types are listed in the channel definition tables.

### Read Only

PROM ReadOnly
: The PROM is always write protected and can never be changed.

RAM ReadOnly
: The variable is stored in RAM and is only accessible for Reading.

### Read Protected Write

EEPROM RPW (Read, Protected Write)
: The EEPROM is always write protected directly following a reset. By setting WriteEnable to TRUE, the contents of the EEPROM can be changed. The contents of the EEPROM will remain unchanged during and after a power failure.

RAM RPW (Read, Protected Write)
: The variable is always write protected following a reset. By setting an input simulation boolean to TRUE, the contents of the RAM can be changed.
NB: This memory type is not utilised in the PD 3221 module.

### Read Write

RAM ReadWrite
: The variable can be changed instantly. After reset or a power failure, it's value is set to zero.

**Read Write, Protected BackUp Write**

RAM InitEEPROM

The variable is stored in both RAM and EEPROM. After a reset, the variable is copied from EEPROM into RAM. When the variable is changed via P-NET, the value is changed in RAM. If WriteEnable is TRUE, the value is changed in both RAM and EEPROM when the variable is changed via P-NET.

RAM AutoSave

Has the same function as RAM InitEEPROM, with the addition that the contents of RAM are automatically copied to EEPROM, at a frequency of approximately 10 hours.

## 2 Service channel.

PD 3221 contains a service channel containing variables and functions common to the entire module.

Variables on Service channel (channel 0). Channel identifier: **Service**

| SWNo | Identifier | Memory type | Read Out | Type |
|---|---|---|---|---|
| 0 | NumberOfSWNo | PROM Read Only | | Integer |
| 1 | DeviceID. | PROM Read Only | -------- | Record |
| 2 | | | | |
| 3 | Reset | RAM Read Write | Hex | Byte |
| 4 | PnetSerialNo | Special function | -------- | Record |
| 5 | | | | |
| 6 | | | | |
| 7 | FreeRunTimer | RAM Read Only | Decimal | Longinteger |
| 8 | WDTimer | RAM Read Write | Decimal | Real |
| 9 | ModuleConfig | EEPROM RPW | -------- | Record |
| A | WDPreset | EEPROM RPW | Decimal | Real |
| B | | | | |
| C | | | | |
| D | WriteEnable | RAM Read Write | Binary | Boolean |
| E | ChType | PROM Read Only | -------- | Record |
| F | CommonError | RAM Read Write | -------- | Record |

**SWNo 0: NumberOfSWNo**
This variable holds the highest SWNo in the module

**SWNo 1: DeviceID**
The purpose of this record is to be able to identify the device. The record includes a registered manufacturer number, the type number of the module and a string, identifying the manufacturer.

The record is of the following type:

```
Record
    DeviceNumber: Word;          (* Offset = 0 *)
    ProgramVersion: Word;        (* Offset = 2 *)
    ManufacturerNo: Word;        (* Offset = 4 *)
    Manufacturer: String[20];    (* Offset = 6 *)
end
```

An example of the field values in the DeviceID record is shown below:

DeviceNumber = 3221
ProgramVersion= 100 (the first version)
ManufacturerNo = 1
Manufacturer = Proces-Data DK

**SWNo 3: Reset**
By writing $FF to SWNo 3, the module performs a reset, and ExternalReset in CommonError SWNo $F is set TRUE.

**SWNo 4: PnetSerialNo**
This Variable is a record of the following type:

```
Record
    PnetNo: Byte; (* Node Address *)   (* Offset = 0 *)
    SerialNO: String[20];              (* Offset = 2 *)
end
```

The SerialNo is set by the manufacturer, and cannot be changed.
The serial number is used for service purposes and as a ’key’ to setting the module’s P-NET Nodeaddress.

A special function is included for identifying a module connected to a network containing many other modules, having the same or unknown node addresses, and to enable a change of the node address via the P-NET.

Setting a new node address via the P-NET is performed by writing the required node address together with the serial number of the module in question, into the PnetSerialNo at node address $7E (calling all modules). All modules on the P-NET will receive the message, but only the module with the transmitted serial number will store the P-NET node address.
An attempt to write data to node address $7E will give no reply. Consequently the calling master must disable the generation of a transmission error when addressing this node.

In the module, the SerialNo = "XXXXXXXPD", is set by **Proces - Data**, and cannot be changed. The seven X‘s indicate the serialnumber, and PD is the initials of Proces-Data.

**SWNo 7: FreeRunTimer**
FreeRunTimer is a timer, to which internal events are synchronized. The timer is of type Longinteger in 1 /256 Second.

**P-NET Watch Dog function**
PD 3221 UPI is equipped with a P-NET Watchdog, which switches off all the digital outputs, by clearing OutFlags and Control flags, if P-NET communication ceases. The P-NET watchdog uses SWNo 8 and SWNo A.

**SWNo 8: WDTimer [s]**
WDTimer is automatically preset with the value from WDPreset (SWNo A), either each time the module is called via P-NET, or following a power-up or module reset. If the WDTimer reaches zero before it is preset again, the PnetWDRunOut flag will be set, and all the outputs will switch OFF. The timer contains a value in sec.

**SWNo 9: ModuleConfig**
The variable is a record of the following type:

```
Record
    Enablebit   : Bit8;          (* Offset = 0 *)
    Functions   : BYTE;          (* Offset = 1 *)
    Ref_A       : BYTE;          (* Offset = 2 *)
    Ref_B       : BYTE;          (* Offset = 3 *)
end
```

The EnableBit field is not utilised in the module.

The watch-dog facility may be switched on and off by means of the field variable Functions as shown below.

| | |
|---|---|
| ModuleConfig.Functions = 0 | Watchdog |
| ModuleConfig.Functions = $10 | No watchdog |

The Ref_A and Ref_B fields are not utilised in the module.

**SWNo A: WDPreset [s]**
The maximum allowable time between two calls for the module, before the watch-dog is activated, is defined in seconds, in this register.

**SWNo D: WriteEnable**
Write protected variables can only be changed when WriteEnable is TRUE ("1"). After reset, WriteEnable is set to FALSE.

After modifying the contents of module EEPROM, WriteEnable should be set FALSE. An EEPROM sum check is calculated each time WriteEnable is changed from "TRUE" to "FALSE". This sum check calculation period is approximately 0.5 second. Consequently, the module should not be reset during this period, otherwise an EEPROM error can occur (see SWNo F: CommonError).

NB: Writing to EEPROM is limited to 10,000 cycles for each byte, including the sum check bytes.

**SWNo E: ChType**

Each channel in an interface module is described in an individual ChType variable. This is a Record, consisting of a unique number for the channel type and a TRUE boolean value for each of the registers which are represented within a channel. The register number in a channel, corresponds to the index number in the boolean array. In addition to these fields, various other fields can be found in the record, which depends on the channel type.

The record for the service channel has the following structure:

```
Record
    ChannelType: WORD;          (* Offset = 0 *)
    Exist: Bit16;               (* Offset = 2 *)
    Functions: Bit16;           (* Offset = 4 *)
end
```

For the service channel, ChType has the following value:

**ChannelType** = 1

**Exist** =

| 15 | | | | | | | | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |

**Functions** =

| 15 | | | | | | | | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

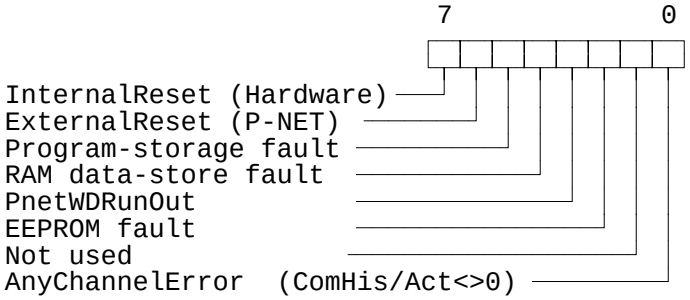
No Watchdog
Watchdog

**SWNo F: CommonError**

The CommonError variable holds error information on all Channels.
This variable is a record of the following type:

```
Record
      ChError: Record
                  His:Array[0..7] of Boolean;   (* Offset = 0 *)
                  Act:Array[0..7] of Boolean;   (* Offset = 2 *)
               End;
      ComHis:Array [0..$F] of Boolean;          (* Offset = 4 *)
      ComAct:Array [0..$F] of Boolean;          (* Offset = 6 *)
End
```

The 8 bits in ChError.His and ChError.Act have the following meaning:




Bit 7 InternalReset is set TRUE if a reset is caused by a power failure, or if the power has been disconnected.

Bit 6 ExternalReset is set TRUE if a reset is caused by writing $FF to SWNo 3, Reset, via P-NET.

Bit 5 Program-storage fault is set TRUE if the self test finds an error in the program memory (PROM).

Bit 4 RAM data-store fault is set TRUE if the self test finds an error in the data memory (RAM).

Bit 3 PnetWDRunOut is set TRUE if the WDTimer reaches zero and the Watchdog function is switched ON.

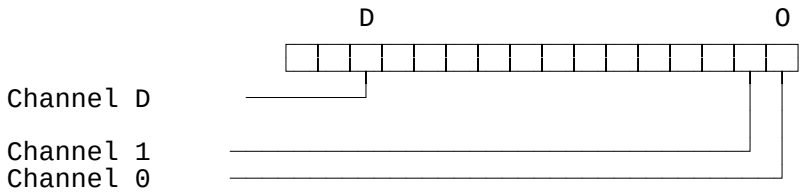
Bit 2 EEPROM fault is set to TRUE if the self test finds an error in the data memory (EEPROM). The error may be corrected by setting and resetting WriteEnable.

Bit 0 AnyChannelError = 1 means that an error or an unknowledged error exists, in one or more channels.

The following function of ChError.His and ChError.Act is analogous in all Channels:

**1** When an error occurs the corresponding bits in ChError.Act and ChError.His is set.
**2** When the error disappears the corresponding bit is reset in ChError.Act.
**3** After reading ChError.His, ChError.Act is copied to ChError.His.
**4** Transmission responses from a module will include the Actual Data Error bit (DataError) set TRUE if ChError.Act <> 0.
**5** The Historical Data Error bit (GeneralError) will be set TRUE in all responses from the module if ChError.His <> 0.

ComHis and ComAct are unique fields in the service channel, and hold an error status relating to all channels, where the bit number corresponds to the channel number. Each Channel has an error register, ChError. If ChError.His in a particular channel is<>0, the corresponding bit is set in ComHis. If ChError.Act in a particular channel is<>0, the corresponding bit is set in ComAct in the service channel. If the error disappears (ChError.Act = 0), the corresponding bit in ComAct is automatically cleared.
If the channels become error free, individual bits in ComHis will be cleared when reading ChError in each of the channels.
ComHis:=0 performs a special function, equivalent to reading all ChErrors in all channels.



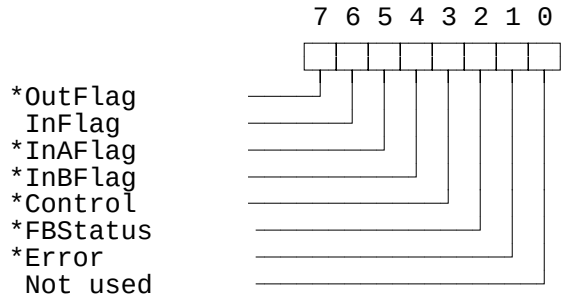
## 3 Digital I/O channel (channel 1 - 6).

Variables on analog input channel x. Channel identifier: **Digital_IO_x**

| SWNo | Identifier | Memory type | Read out | Type | SI Unit |
|---|---|---|---|---|---|
| x0 | FlagReg | RAM Read Write | Binary | Bit8 | - - - |
| x1 | OutTimer * | RAM Read Write | Decimal | Real | s |
| x2 | Counter | RAM Auto Save | Decimal | LongInteger | - - - |
| x3 | OutCurrent * | RAM Read Write | Decimal | Real | A |
| x4 | Operatingtime | RAM Auto Save | Decimal | Real | s |
| x5 | | | | | |
| x6 | FBTimer * | RAM Read Write | Decimal | Real | s |
| x7 | FBPreset * | EEPROM RPW | Decimal | Real | s |
| x8 | OutPreset * | EEPROM RPW | Decimal | Real | s |
| x9 | ChConfig | EEPROM RPW | - - - - - - | Record | - - - |
| xA | MinCurrent * | EEPROM RPW | Decimal | Real | A |
| xB | MaxCurrent * | EEPROM RPW | Decimal | Real | A |
| xC | | | | | |
| xD | Maintenance * | EEPROM RPW | - - - - - - | Record | - - - |
| xE | ChTyper | PROM Read Only | - - - - - - | Record | |
| xF | ChError | RAM Read Only | Binary | Record | - - - |

* Not available for CH 5 to CH 6.

**SWNo x0: FlagReg**




**Bit 7: OutFlag**

This flag controls the output if the P-NET WDRunOut bit is FALSE, and the channel is configured as an output. The special output functions control the output flag, in the same way as a P-NET transmission.

OutFlag:=True => Output ON

PnetWDRunOut = True => OutFlag:=False => Output OFF

**Bit 6: InFlag**

The input flag is controlled by the input detector, and shows the logic level on the input terminals. The input flag is true, when the input is connected to 24V-. If the channel is used as an output, the input flag will follow the output flag. If the output terminals are short-circuited, the input flag will not follow the output flag. The input signal can be simulated, by setting the channel in input simulation mode (ChConfig.Enablebit[0] = TRUE) and subsequently writing the state to the input flag.

**Feedback Control**

Many process components are equipped with feedback contacts. A typical example is a valve with 1 or 2 micro switches, which indicate the mechanical position of the piston. In this application, in addition to the output, one or two inputs are needed. The inputs to be used by the feedback-control, are selected in ChConfig.Ref_A and ChConfig.Ref_B in the output-Channel.

**Bit 5 and 4: InAFlag, InBFlag**

The InAFlag is identical to the input flag for the channel selected as Feedback-input A. Feedback-input A is the input signal which corresponds to the same state as the output signal. Therefore Feedback-input A must be TRUE when the output is TRUE to indicate a correct feedback signal. The Feedback-input A channel is selected in ChConfig.Ref_A. The InBFlag is identical to the input flag for the channel selected as Feedback-input B. Feedback-input B is the input signal which corresponds to the inverse state as the output signal. This input signal is selected in ChConfig.Ref_B. The feedback signals can be simulated by setting ChConfig.Enablebit[2] = TRUE. If feedback simulation is selected, the InAFlag and InBFlag are automatically set to the correct state to correspond to the current state of OutFlag.

**Bit 3: Control**

When the Control flag is set, a special output-function (one shot output, 50 % duty cycle output or output controlled by pulse processor) for the channel is enabled, which then controls the output. Clearing the Control flag will disable the special output-function and clears the output, which may then be controlled via the P-NET. After a power-up or a reset of the module, the Control flag is FALSE, unless the channel is configured to be controlled by the pulse processor, in which case, the Control flag is always set TRUE.

The state of the Control flag has no influence on the output if the channel is configured as ordinary output (ChConfig.Functions := $10).

**Bit 2: FBStatus**

The FBStatus indicates the current feedback condition. The value of FBStatus does not depend on the FBTimer, which means that the actual valve position for example, can be ascertained as correct, or incorrect, before the FBTimer has reached zero. If feedback is used (single or double), FBStatus is always set TRUE when the OutFlag is changed. If no feedback is used, FBStatus always reads FALSE.

FBStatus = TRUE, indicates that the feedback signal/s are incorrect.

**Bit 1: Error**
The purpose of Error is to indicate an error condition on the channel (incorrect feedback-signals, overload, underload, PrgError and hardware errors).
Error = TRUE, indicates ChError.Act <> 0. ( See SWNo $xF).

**SWNo x1: OutTimer [s]**
Each output channel has a timer, used with the special output-functions. The timer is either preset via P-NET, or from the preset register, depending on which function is selected for the channel. The timer counts down, with a resolution of 1/8 second. The count continues through negative values. The timer register is cleared after a power failure. The maximum value for the timer is approximately 97 days. Following an overflow, the timer continues from it's maximum value.

**SWNo x2: Counter**
The counter counts the number of pulses at the input. The maximum count frequency is 50 Hz. The counter counts up to a maximum of 2147483647 (a LongInteger).
When the counter exceeds +2147483647, it re-starts at -2147483648. The counter increments by one, every time the InFlag changes from "0" to "1".

**SWNo x3: OutCurrent [A]**
The OutCurrent register indicates the sink current in the output load. It is measured with a resolution of approximately 12 mA. The stability of the measurement depends on the power supply stability.

**SWNo x4: Operatingtime [s]**
This variable totalises the time period InFlag is True. The resolution for OperatingTime is 0.5 sec.

**SWNo x6: FBTimer [s]**
The FeedBack-timer is used to disable feedback error detection while the mechanical components are changing position. The FBTimer is preset from FBPreset when the output changes state. The timer counts down.

```
If FBTimer < 0 then
begin
  ChError.Act[FeedbackError] := FlagReg[FBStatus];
  ChError.His[FeedbackError] := FlagReg[FBStatus];
end
ELSE ChError.Act[FeedbackError] :=False.
```

**SWNo x7: FBPreset [s]**
This register holds a value equal to the maximum permitted time for incorrect feedback signals to be present, before an error is flagged. The value is passed to FBTimer when the output changes state.

**SWNo x8: OutPreset [s]**
This variable holds a preset value for the OutTimer. The preset value is passed to the OutTimer by the special output-functions.

**SWNo x9: ChConfig**
This variable selects the I/O type, the type of feedback control (single or double feedback) and a choice of special output-functions.

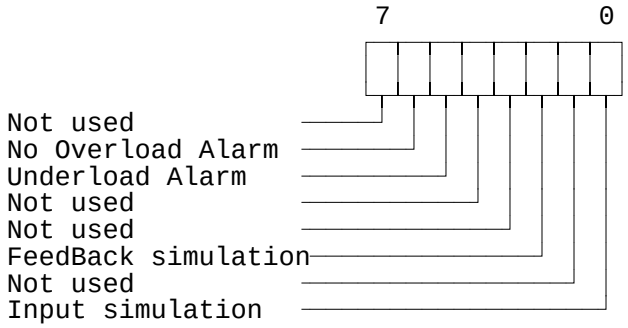
Feedback control: The correct feedback state is Input A = Output and Input B = NOT Output. The feedback inputs can be disabled by writing a 0 as the channel number, in ChConfig.Ref_A and/or ChConfig.Ref_B fields, if only one or no input channels are required.

The ChConfig variable is a record of the following type:

```
Record
    Enablebit   : Bit8;          (* Offset = 0 *)
    Functions   : BYTE;          (* Offset = 1 *)
    Ref_A       : BYTE;          (* Offset = 2 *)
    Ref_B       : BYTE;          (* Offset = 3 *)
end
```

where each field has the following interpretation:

**Enablebit** :



**Functions** :

Functions = $00 => Input only
Functions = $10 => Output
Functions = $20 => One shot output
Functions = $30 => 50% Duty-Cycle output
Functions = $40 => Output contr. by PP

**Ref_A** :

Channel No. for FeedBack input A (input = output).

**Ref_B** :

Channel No. for FeedBack input B (input ≠ output).

**Special output-functions:**

**One shot output.**

This automatic function is selected by setting ChConfig.Functions = $20. When the Control Flag is changed from FALSE to TRUE, the output will be set true for a period equal to the value of OutPreset. The time can be varied by reloading the OutTimer. Output is reset if the Control Flag is set to FALSE and the output may be controlled directly via P-NET.

Precise Function description:

```
After reset: InternalState:=False; FlagReg[Control]:=False;

Loop
    If NOT InternalState and FlagReg[Control] then (* Positive edge*)
      OutTimer:=OutPreset
    If InternalState and NOT FlagReg[Control] then (* Negative edge*)
      FlagReg[OutFlag]:=False;
    InternalState:=FlagReg[Control];
    If FlagReg[Control] then
      If OutTimer > 0 then
        FlagReg[OutFlag]:=true ELSE FlagReg[OutFlag]:=false
End
```

**50% Duty-cycle Output.**
This automatic function is selected by setting ChConfig.Functions = $30. If the Control Flag is set, the output is inverted with a time interval equal to OutPreset. The time for a period (one OFF period and one ON period) is twice the value of the contents of OutPreset.
If the Control Flag is reset, the output switches OFF and may then be controlled via P-NET.

Precise Function description:

```
After reset: InternalState:=False; FlagReg.Control:=False;

Loop
  If InternalState=False and FlagReg[Control]=True then (* Positive edge*)
    OutTimer:=OutPreset;
  If InternalState=True and FlagReg[Control]=False then (* Negative edge*)
    FlagReg[OutFlag]:=False;
  InternalState:=FlagReg[Control];
  If FlagReg[Control]=True and OutTimer <= 0 then
  Begin
    FlagReg[OutFlag]:=NOT FlagReg[OutFlag];
    OutTimer:=OutPreset;
  End
End
```

**Output controlled by the Pulse Processor (see PP channel).**
This automatic function is selected by setting ChConfig.Functions = $40. When the Control flag is set TRUE, the output is controlled by the pulse processor. If this function is selected, the Control flag is always set TRUE after a reset. The Feedback control does not function in this mode.

**SWNo xA: MinCurrent [A]**
This variable defines the minimum permitted current in the load when the output is ON. If Outcurrent is less than MinCurrent when the output is ON and the FBTimer < 0, an error can be generated. Error-code generation is enabled by setting ChConfig.-Enablebit[5] TRUE (underload alarm).

Precise Function description:

```
If (OutCurrent < MinCurrent) and (FlagReg.[OutFlag]=true)
       and (FBTimer < 0) and EnableBit[5] then
  ChError.Act[UnderLoad]:= true
else
  ChError.Act[UnderLoad]:= false
```

**SWNo xB: MaxCurrent [A]**
If Outcurrent exceeds MaxCurrent and FBTimer < 0, the output is switched off and an error is generated. For applications in which it is normal to let the max. current switch the output off, the error-code generation can be disabled, by setting ChConfig.-Enablebit[6] TRUE (no overload alarm). MaxCurrent can not exceed 1.0 Amp. Any output current > 2 Amp switches the output off instantly.

Precise Function description:

```
If ((OutCurrent > MaxCurrent) and (FlagReg[OutFlag] = true) and (FBTimer < 0))
or (OutCurrent > 2 Amp) then
Begin
  FlagReg[OutFlag]:=false;
  FlagReg[Control]:=false;
  If NOT EnableBit[6] then
  Begin
    ChError.Act[OverLoad]:= true;
    ChError.His[OverLoad]:= true;
  End
End
Else ChError.Act[OverLoad]:= false
```

**SWNo xD: Maintenance**
The Maintenance variable is used for service management and maintenance purposes, and holds the last date of service and an indication of the type of service.
Date, month, year, type.

The Maintenance is a record of the following type:

```
Record
      Date      : BYTE;          (* Offset = 0 *)
      Month     : BYTE;          (* Offset = 1 *)
      Year      : BYTE;          (* Offset = 2 *)
      Type      : BYTE;          (* Offset = 3 *)
end
```

**SWNo xE: ChType**

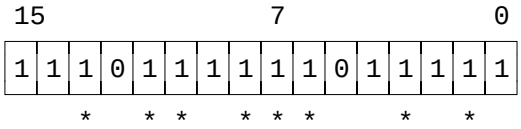
For the digital I/O channels, ChType is of the following type:

```
Record
    ChannelType: WORD;          (* Offset = 0 *)
    Exist: Bit16;               (* Offset = 2 *)
    Functions: Bit16;           (* Offset = 4 *)
    FeedBack: Bit16;            (* Offset = 6 *)
end
```

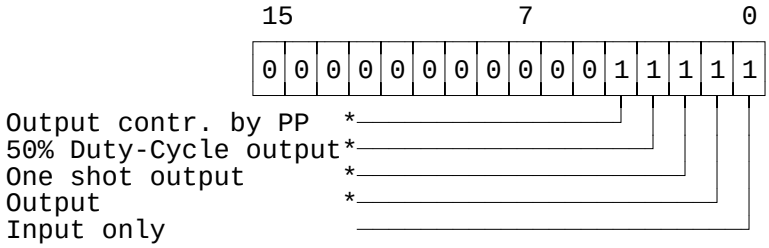
ChType has the following value:

**ChannelType** = 2

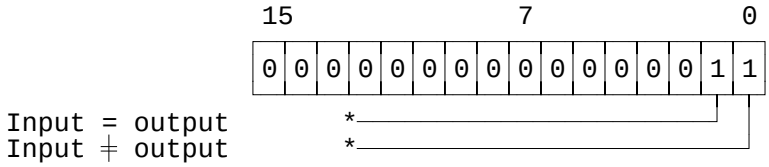
**Exist** =




**Functions** =




**Feedback** =

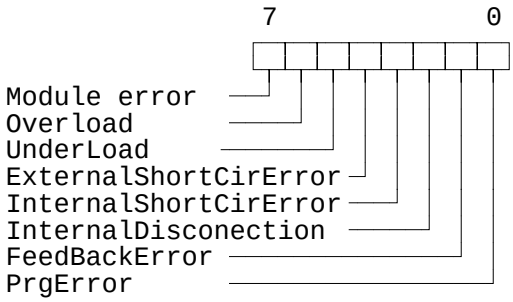



* These bits are not set for Ch 5 to Ch 6.

**SWNo xF: ChError**

```
ChError: Record
            His:Array[0..7] of Boolean;   (* Offset = 0 *)
            Act:Array[o..7] of Boolean;   (* Offset = 2 *)
        End;
```

The 8 bits in ChError.His and ChError.Act have the following meaning. When an error occurs, the corresponding bit is set in both ChError.His and ChError.Act. When the error disappears, the bit is cleared in ChError.Act.




Bit 7 Module error. If this bit is set, the rest of the bits have no meaning because a module error can cause random error codes on the individual channels (see also "Service channel").

Bit 6 OverLoad is set if the current in the output load exceeds MaxCurrent (default 1 A). The Overload alarm can be disabled (ChConfig.Enablebit[6]).

Bit 5 UnderLoad is set if the load is disconnected (OutCurrent < MinCurrent). The Underload alarm can be enabled by setting ChConfig.Enablebit[5] TRUE.

Bit 4 ExternalShortcirError is set if an external short circuit error is detected (InFlag=1 and OutFlag=0). The error bit can not be set on channels configured as input. This error bit will not appear in input simulation mode.

Bit 3 InternalShortCirError is set if the output transistor is short circuited (OutFlag=0 and OutCurrent > 0.1 Amp ). This error bit will not appear in input simulation mode.

Bit 2 InternalDisconection is set if the output transistor is disconnected (Out-Flag=1 and InFlag = 0 and NOT overload). This error bit will not appear in input simulation mode.

Bit 1 FeedBackError is set if there is a feedback error (see FBTimer).

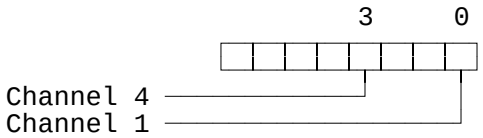
Bit 0 PrgError is set following attempts to set FlagReg[OutFlag] if the I/O is configured to be an input. (This bit is only set in ChError.His).

# 4 Common I/O channel (channel 7).

Variables on Common I/O channel. Channel identifier: **CommonIO**

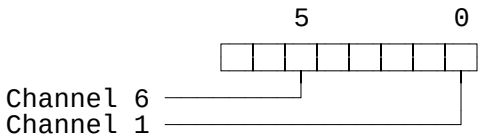
| SWNo | Contents | Memory type | Read out | Format |
|---|---|---|---|---|
| 70 | OutFlags | RAM Read Write | Hex | Bit8 |
| 71 | InFlags | RAM Read Write | Hex | Bit8 |
| 72 | | | | |
| 73 | | | | |
| 74 | | | | |
| 75 | | | | |
| 76 | | | | |
| 77 | | | | |
| 78 | | | | |
| 79 | | | | |
| 7A | | | | |
| 7B | | | | |
| 7C | | | | |
| 7D | | | | |
| 7E | ChType | PROM Read Only | - - - - - | Record |
| 7F | IOChError | RAM Read Only | Hex | Record |

**SWNo 70: OutFlags**




This variable contains all the OutFlag's from all I/O channels. This means that all digital outputs in the module can be controlled from this register.

**SWNo 71: InFlags**



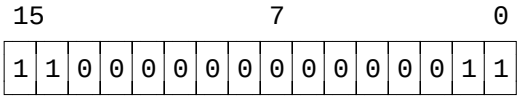

This variable contains all the InFlag's from all I/O channels. This means that all digital inputs in the module can be read in this register.

**SWNo 7E: ChType**
For the common channel, ChType is of following type:

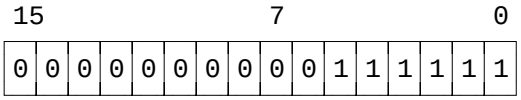
```
Record
    ChannelType: WORD;          (* Offset = 0 *)
    Exist: Bit16;               (* Offset = 2 *)
    ExistingChannels: Bit16;    (* Offset = 6 *)
end
```

ChType has the following value:

**ChannelType** = 3
**Exist** =



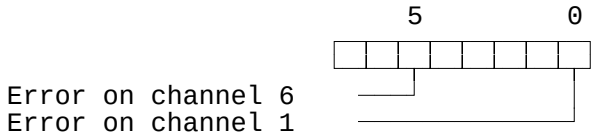

**ExistingChannels** =




**SWNo 7F: IOChError**

```
IOChError: Record
        His:Array[0..7] of Boolean;   (* Offset = 0 *)
        Act:Array[0..7] of Boolean;   (* Offset = 2 *)
    End;
```

Meaning of IOChError.His and IOChError.Act:




The IOChError variable indicates if there is an error, or an unacknowledged error, in one or more I/O channels (1-6). IOChError.His is set if ChError.His of any channel <> 0 and IOChError.Act is set if any Channel contains a ChError.Act <> 0.
Reading IOChError does not acknowledge the channel errors. This can only be done by reading ChError in the individual channels, or by means of SWNo $0F (channel 0).

# 5 Analog input channel (channel 8-9).

2 analog input signals may be connected to the PD 3221, which can be a current (0-20 or 4-20 mA), a voltage (0-100 mV) or a Pt-100 temperature detector. Signal type etc. is selected individually for each channel, by means of channel configuration (ChConfig). The input signals are filtered, and suppressed against 50 and 60 Hz interference.

Variables on analog input channel x. Channel identifier: **Analog_in_x**

| SWNo | Identifier | Memory type | Read out | Type | SI Unit |
|---|---|---|---|---|---|
| x0 | AnalogIn | RAM Read Write | Decimal | Real | * |
| x1 | | | | | |
| x2 | | | | | |
| x3 | | | | | |
| x4 | | | | | |
| x5 | | | | | |
| x6 | | | | | |
| x7 | HighLevel | RAM Init EEPROM | Decimal | Real | * |
| x8 | LowLevel | RAM Init EEPROM | Decimal | Real | * |
| x9 | ChConfig | EEPROM RPW | Hexadec. | Record | |
| xA | | | | | |
| xB | FullScale | EEPROM RPW | Decimal | Real | * |
| xC | ZeroPoint | EEPROM RPW | Decimal | Real | * |
| xD | Maintenance | EEPROM RPW | | Record | |
| xE | ChType | PROM Read Only | - - - - - - | Record | |
| xF | ChError | RAM Read Only | Binary | Record | |

* SI unit depends on the connected process component.

**SWNo x0: AnalogIn**

This variable holds the measurement result of the temperature, current or voltage input, (selected in ChConfig.Functions) as a scaled value in SI units, according to the contents of FullScale and ZeroPoint. If the value should attempt to exceed 110% of FullScale (220 °C for Pt100), the contents of this register will be held at 110% of FullScale (220 °C for Pt100) and the module will generate an error code (see ChError). A similar situation occurs if the output signal attempts to drop below -5% (-105 °C for Pt100).

**SWNo x7: HighLevel**

HighLevel is a "limitswitch" with the following function:
*If AnalogIn > HighLevel and ChConfig.Enablebit[4] = 1 then HighAlarm:=true else HighAlarm:=false.*

**SWNo x8: LowLevel**

LowLevel is a "limitswitch" with the following function:
*If AnalogIn < LowLevel and ChConfig.Enablebit[3] = 1 then LowAlarm:=true else LowAlarm:=false.*

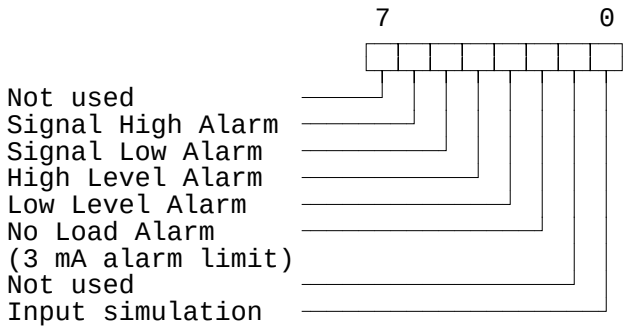
**SWNo x9: ChConfig**

The channel configuration for an analog input channel is stored in a record variable of the following type:

```
Record
    Enablebit   : Bit8;          (* Offset = 0 *)
    Functions   : BYTE;          (* Offset = 1 *)
    Ref_A       : BYTE;          (* Offset = 2 *)
    Ref_B       : BYTE;          (* Offset = 3 *)
end
```

where each field has the following interpretation:

**Enablebit** :




**Functions** :

The Functions field holds a 2 digit hexadecimal value, where the most significant digit is used to specify the input signal type, and the least significant digit is used to select a time constant for the input filter. The filter time constant defines that for a stepped change at the input terminals, the measured value (AnalogIn), will have only changed by 63 % of it's target value, at the end of the specified time period.

Input signal specification:

| | |
|---|---|
| Functions = $0x | => Channel disabled |
| Functions = $1x | => Temperature Pt-100 |
| Functions = $2x | => Current 0-20 mA |
| Functions = $3x | => Current 4-20 mA |
| Functions = $4x | => Voltage, 0-100 mV |

Filter constant specification:

| | |
|---|---|
| Functions = $x0 | => No filter |
| Functions = $xA | => Time constant = 1 sec. |
| Functions = $xB | => Time constant = 2 sec. |
| Functions = $xC | => Time constant = 5 sec. |
| Functions = $xD | => Time constant = 10 sec. |

**Ref_A** : Not used

**Ref_B** : Not used

**Note:** If a channel is not in use, "00" should be written in ChConfig.Functions, ("channel disable"), otherwise errors can occur.
No filter should be selected, if the AnalogIn value is to be used for regulation.

When the input is a (4-20 mA) current, an extra function can be used (3 mA low-limit alarm). If this function is selected, the module will generate an error as soon as the current through the input falls below 3 mA. The function is enabled by setting ChConfig.Enablebit[2] to TRUE (see register F).

When the channel is configured for input simulation mode (ChConfig.Enablebit[0] = TRUE), no measurement will be calculated, and it is possible for the user to insert any value in AnalogIn.

**SWNo xB: FullScale**
The resultant measured value expected in AnalogIn when the input signal is at it's maximum, e.g. 20 mA / 100 mV, should be placed in the FullScale variable. FullScale is not used for Pt-100 signals.

**SWNo xC: ZeroPoint**
Current and voltage signals: The resultant measured value expected in AnalogIn when the input signal is at it's minimum, e.g. 0 mA / 4 mA / 0 mV, should be placed in the ZeroPoint variable. The value is inserted in SI units corresponding to the result value in AnalogIn. If the channel has been configured as a Pt-100 input, ZeroPoint is used as an offset-adjustment for the temperature detector.

**SWNo xD: Maintenance**
The Maintenance variable is used for service management and maintenance purposes, and holds the last date of service and indicates the type of service.
Date, month, year, type (see also Digital I/O channel).

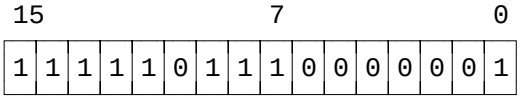
**SWNo xE: ChType**
For the analog input channels, ChType is of the following type:

```
Record
    ChannelType: WORD;          (* Offset = 0 *)
    Exist: Bit16;               (* Offset = 2 *)
    Functions: Bit16;           (* Offset = 4 *)
    FilterConstant: Bit16;      (* Offset = 6 *)
end
```

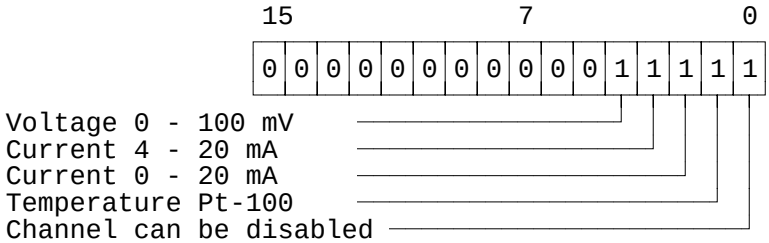
ChType has the following value:

**ChannelType** = 4

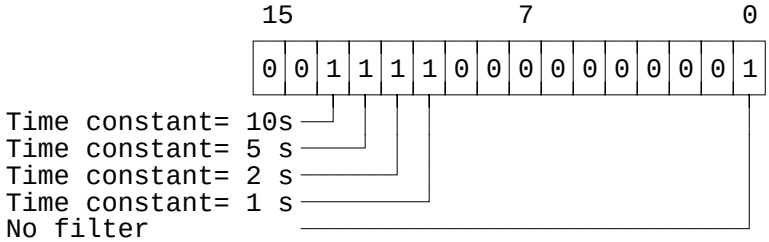
**Exist** =




**Functions** =



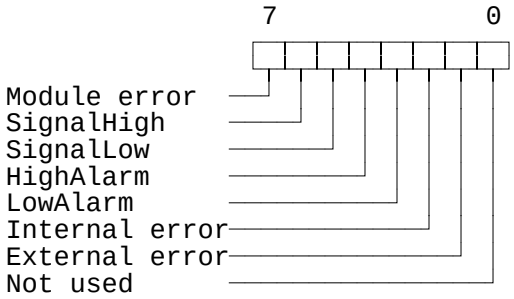

**FilterConstant** =




**SWNo xF: ChError**

```
ChError: Record
        His:Array[0..7] of Boolean;   (* Offset = 0 *)
        Act:Array[0..7] of Boolean;   (* Offset = 2 *)
    End;
```

The 8 bits in ChError.His and ChError.Act have the following meaning. When an error occurs, the corresponding bit is set in both ChError.His and ChError.Act, and when the error disappears, the bit is cleared in ChError.Act.




Bit 7 Module error. If this bit is set, the rest of the bits are insignificant, because a module error can lead to random error codes on individual channels (also see "Service channel").

Bit 6 SignalHigh is set if the input signal exceeds the maximum value (20mA, 100mV and 200°C) with more than 10% and ChConfig.Enablebit[6] = TRUE.

Bit 5 SignalLow is set if the input signal falls below the minimum value (0mA, 0mV and -100°C) with more than 5% and ChConfig.Enablebit[5] = TRUE.

Bit 4 HighAlarm is set if AnalogIn > HighLevel and ChConfig.Enablebit[4] = TRUE.

Bit 3 LowAlarm is set if AnalogIn < LowLevel and ChConfig.Enablebit[3] = TRUE.

Bit 2 An internal error is indicated. If the module continues to indicate internal error after a reset, the module is likely to require repair.

Bit 1 External error indicates different kinds of errors, depending on the channel configuration for the Input signal:

**4-20 mA:** External error is set as a 3 mA alarm limit if the channel is configured for current input (4 - 20 mA) and the current input signal falls below 3 mA and ChConfig.Enablebit[2] = TRUE.

**Pt-100:** External error is set if the current in the Pt-100 sensor is outside a legal range for measurement due to disconnection or a broken wire. This error can not be disabled.

Bit 0 Not used.

## 5.1 Connection to analog input channels.

The PD 3221 module has two sets of terminals marked "A-B-C-D", which are used for connection of the analog input signals. The connection for each signal is determined from the measurement type, and each signal processing is determined from the channel configuration.
When using a shielded cable, only terminal D should be connected to the screen, and only at this end of the cable. Refer to the figures below.

### 5.1.1 Temperature measurements.

Temperature detectors of the Pt-100 type can be connected to the PD 3221 module. A detector is connected to one of two sets of terminals marked "A-B-C-D", as shown in fig. 5.1.1.a. If twisted pair cable is used, then terminal connections B and C must constitute one pair, and terminal A and D the other pair.

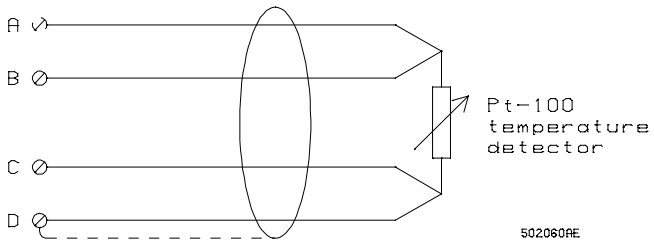



Fig. 5.1.1.a.

### 5.1.2 Current measurement, 0-20 or 4-20 mA.

Analog current signals of 0-20 mA or 4-20 mA characteristic can be connected to the PD 3221 module.
Such a signal is connected to one of two sets of terminals marked "A-B-C-D", as shown in fig. 5.1.2.a.

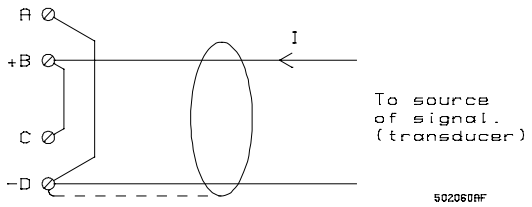



Fig. 5.1.2.a.

### 5.1.3 Voltage measurement, 0-100 mV.

Analog voltage signals of 0-100 mV characteristic can be connected to the PD 3221 module.
Such a signal is connected to one of two sets of terminals marked "A-B-C-D", as shown in fig. 5.1.3.a.

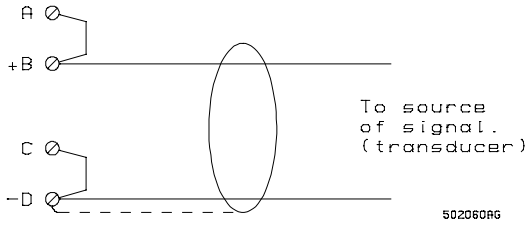



Fig. 5.1.3.a.

The input impedance on an analog channel is so high, that conventional voltage division using two resistors can be applied, if the signal to be measured is higher than 0-100 mV.

Example: Input signal 0-5 V. See fig. 5.1.3.b.

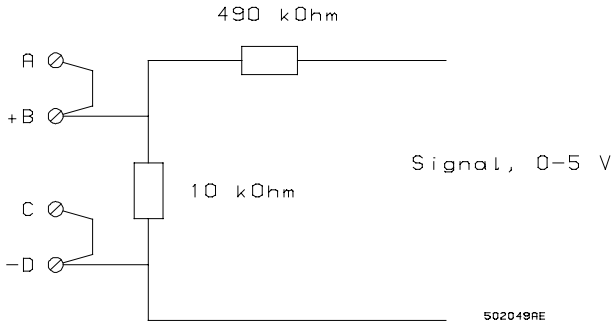
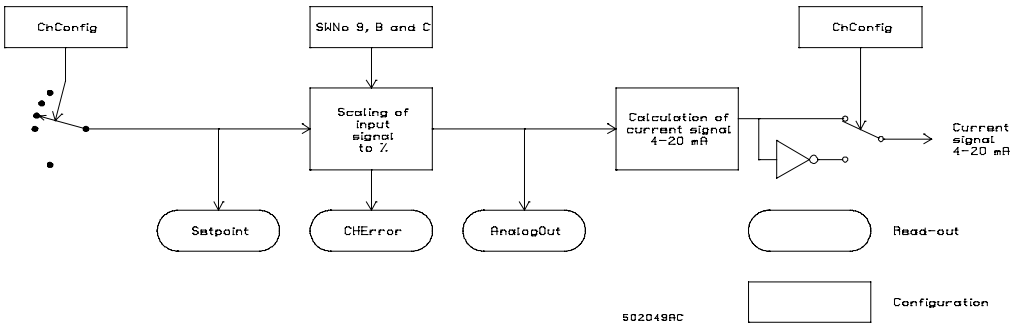



Fig. 5.1.3.b.

# 6 Current output channel (channel A).

The PD 3221 module includes a 4-20mA current output channel. The output current can be allied to the result of a measurement, a result from the calculator channel, the output signal from the internal PID-Regulator, or a value set by the P-NET.

**Block diagram of current output function:**




Variables on current output channel. Channel identifier: **Current_Out**

| SWNo | Identifier | Memory type | Read out | Format | SI Unit |
|---|---|---|---|---|---|
| A0 | AnalogOut | RAM Read Write | Decimal | Real | % |
| A1 | | | | | |
| A2 | | | | | |
| A3 | | | | | |
| A4 | | | | | |
| A5 | | | | | |
| A6 | Setpoint @ | RAM Init EEPROM | Decimal | Real | * |
| A7 | HighLevel | RAM Init EEPROM | Decimal | Real | * |
| A8 | LowLevel | RAM Init EEPROM | Decimal | Real | * |
| A9 | ChConfig | EEPROM RPW | Hexadec. | Record | |
| AA | | | | | |
| AB | FullScale | EEPROM RPW | Decimal | Real | * |
| AC | ZeroPoint | EEPROM RPW | Decimal | Real | * |
| AD | Maintenance | EEPROM RPW | | Record | |
| AE | ChType | PROM Read Only | - - - - - | Record | |
| AF | ChError | RAM Read Only | Binary | Record | |

@ Indirect variable, see section 1.2 * depend on application

**SWNo A0: AnalogOut**
This variable holds the value of the output signal as a percentage (0-100%) of span. If the value should attempt to exceed 100%, the contents of this register will be held at 100% and the module will generate an error code (see ChError). A similar situation occurs if the output signal attempts to drop below 0%.

**SWNo A6: Setpoint @**
Setpoint holds a process value in SI units, or the output value from the regulator-channel, from which the AnalogOut value is derived.

**SWNo A7: HighLevel**
HighLevel is a "limitswitch" with following function:
*If AnalogOut > HighLevel and ChConfig.Enablebit[4] = TRUE then HighAlarm:=true else HighAlarm:=false*

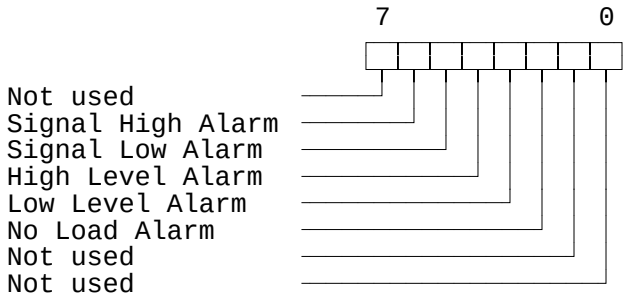
**SWNo A8: LowLevel**
LowLevel is a "limitswitch" with following function:
*If AnalogOut < LowLevel and ChConfig.Enablebit[3] = TRUE then LowAlarm:=true else LowAlarm:=false*

**SWNo A9: ChConfig**
The channel configuration for the current output channel is stored in a record variable of the following type:

```
Record
    Enablebit   : Bit8;          (* Offset = 0 *)
    Functions   : BYTE;          (* Offset = 1 *)
    Ref_A       : BYTE;          (* Offset = 2 *)
    Ref_B       : BYTE;          (* Offset = 3 *)
end
```

where each field has the following interpretation:

**Enablebit** :



**Functions** :
The Functions field is used for Output signal specification:

Functions = $00 => Output disabled
Functions = $10 => Current output 0-100% : 4-20 mA
Functions = $20 => Current output 0-100% : 20-4 mA

**Ref_A** : SWNo for Setpoint @
**Ref_B** : Not used

The two alarms, SignalHigh and SignalLow (see SWNo $AF) can be enabled by setting ChConfig.Enablebit[6] and ChConfig.Enablebit[5] TRUE.

The current-output is equipped with a current sensor that indicate whether the current is lower than expected. This can be caused by a disconnected load, too high a load impedance or too low a supply voltage. If No Load Alarm is enabled, (ChConfig.Enablebit[2] = TRUE), NoLoad in ChError will be set if any of these conditions occur.

The current output can be disabled by setting ChConfig.Functions = 0. When disabled, no error-codes will be generated.

ChConfig.Functions is used to select whether 100% corresponds to 20 mA or 4 mA, depending on the connected process component.

**SWNo AB: Fullscale**
Fullscale is set to be equal to the Setpoint value which will produce the max. output signal (100%).

**SWNo AC: Zero Point**
Zeropoint is set to be equal to the Setpoint value which will produce the min. output signal (0%)

**SWNo AD: Maintenance**
The Maintenance variable is used for service management and maintenance purposes, and holds the last date of service and indicates the type of service.
Date, month, year, type (see also Digital I/O channel).

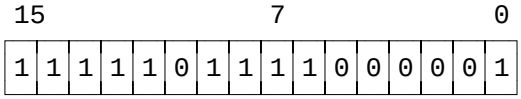
**SWNo AE: ChType**
For the analog output channel, ChType is of the following type:

```
Record
    ChannelType: WORD;          (* Offset = 0 *)
    Exist: Bit16;               (* Offset = 2 *)
    Functions: Bit16;           (* Offset = 4 *)
end
```

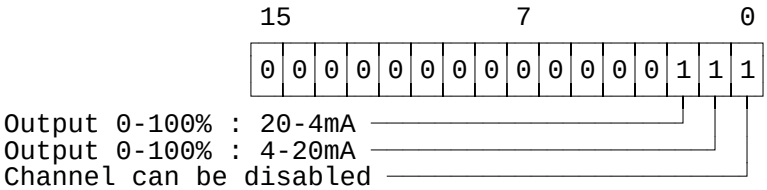
ChType has the following value:

**ChannelType** = 5

**Exist** =



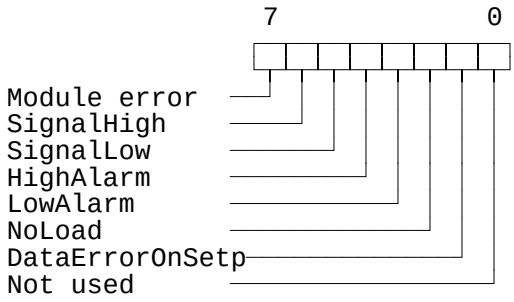

**Functions** =




**SWNo AF: ChError**

```
ChError: Record
        His:Array[0..7] of Boolean;   (* Offset = 0 *)
        Act:Array[0..7] of Boolean;   (* Offset = 2 *)
    End;
```

The 8 bits in ChError.His and ChError.Act have the following meaning. When an error occurs, the corresponding bit is set in both ChError.His and ChError.Act. When the error disappears, the bit is cleared in ChError.Act.



Bit 7 Module error. If this bit is set, the rest of the bits are insignificant, because a module error can lead to random error codes on individual channels (also see "Service channel").

Bit 6 SignalHigh is set if AnalogOut exceeds 100% and ChConfig.Enablebit[6] = TRUE.

Bit 5 SignalLow is set if AnalogOut falls below 0% and ChConfig.Enablebit[5] = TRUE.

Bit 4 HighAlarm is set if AnalogOut > HighLevel and ChConfig.Enablebit[4] = TRUE.

Bit 3 LowAlarm is set if AnalogOut < LowLevel and ChConfig.Enablebit[3] = TRUE.

Bit 2 NoLoad is set if the output current falls below a preset limit and ChConfig.Enablebit[2] = TRUE.

Bit 1 DataErrorOnSetp is set when the setpoint is an indirect variable, and the channel of the variable that is pointed to has a ChErrror.Act <> 0.

Bit 0 Not used

## 6.1 Current Output, Electrical.

The current output is protected against incorrect polarization, by a zenerdiode and a current limiting resistor. This resistor is rated so that current limit occurs at approximately 35 mA. The output must be disconnected totally for a few seconds, before the output can be used again.

The external control equipment must have a specification, such that the voltage at the current output terminal will always be greater than 5 V.

For details of electrical connection for the current output, refer to the hardware diagram at page 3.

# 7 PID-regulator.

The PD 3221 module is equipped with an internal PID-regulator (channel B), which can be used for a variety of control purposes.

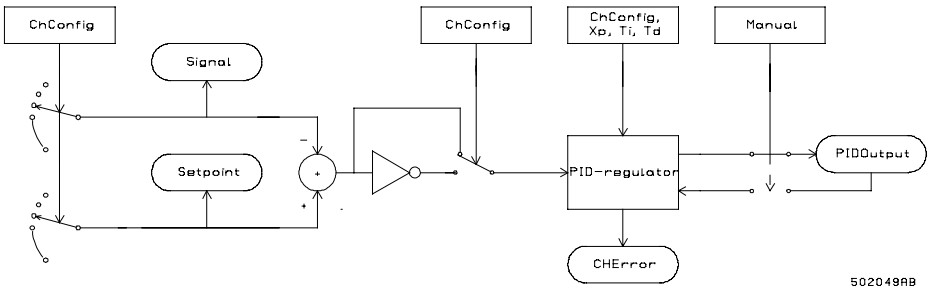
The PID-regulator can be used either to control a current signal (4- 20mA) or a variable duty cycle digital signal.

The regulator is set with the P, I and D parameters, and a code, which defines the regulator input signals.

The Process Variable and the Setpoint can either be derived from an input channel, a calculator channel or P-NET.

The regulator can be set to "Manual".

**Block diagram of PID-regulator:**



**Variables for PID-regulator (channel B).**

Variables on PID-regulator channel. Channel identifier: **PID**

| SWNo | Identifier | Memory type | Read out | Format | SI Unit |
|---|---|---|---|---|---|
| B0 | PIDOutput | RAM Read Write | Decimal | Real | % |
| B1 | Signal @ | RAM Read Write | Decimal | Real | * |
| B2 | | | | | |
| B3 | | | | | |
| B4 | | | | | |
| B5 | | | | | |
| B6 | Setpoint @ | RAM Init EEPROM | Decimal | Real | * |
| B7 | | | | | |
| B8 | Manual | RAM Init EEPROM | Binary | Boolean | |
| B9 | ChConfig | EEPROM RPW | Hexadec. | Record | |
| BA | | | | | |
| BB | Xp | EEPROM RPW | Decimal | Real | * |
| BC | Ti | EEPROM RPW | Decimal | Real | s |
| BD | Td | EEPROM RPW | Decimal | Real | s |
| BE | ChType | PROM Read Only | - - - - | Record | |
| BF | ChError | RAM Read Only | Binary | Record | |

@ Indirect variables see section 1.2. * depend on application

**SWNo B0: PIDOutput**
The output signal from the PID-regulator is normally represented by this variable, as a value between 0 and 100%. If the value should attempt to exceed 100%, the contents of this register will be held at 100% and the module will generate an error code (see ChError). A similar situation occurs if the output signal attempts to drop below 0%. The PIDOutput value is updated at the same cyclic rate as analog channel input signals.

**SWNo B1: Signal @**
This register holds the value of the selected Process variable.

**SWNo B6: Setpoint @**
This variable holds the selected Setpoint value, to be used for PID-regulator control.

**SWNo B8: Manual**
This boolean variable can be used to switch between automatic and manual control, by writing FALSE or TRUE into the register. TRUE corresponds to MANUAL.

When the regulator is in "Manual", it is possible to preset the output value. When the regulator is switched to automatic, the output will continue from the manual value. This facility is called "Bump-less transfer". After presetting a manual output value, a delay of at least 1 sec. must pass before the regulator is set back to automatic operation.

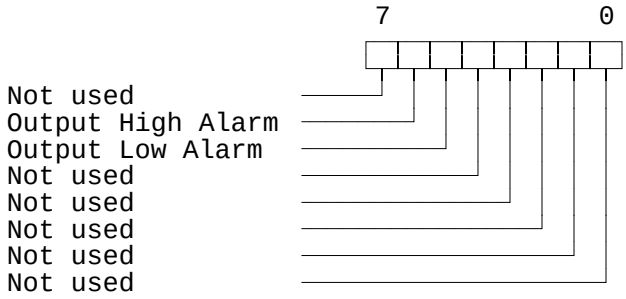
The "Manual" facility can also be used to quickly switch off the output signal, for example in an alarm situation.

**SWNo B9: ChConfig.**
The channel configuration for a PID regulator channel is stored in a record variable of the following type:

```
Record
    Enablebit   : Bit8;          (* Offset = 0 *)
    Functions   : BYTE;          (* Offset = 1 *)
    Ref_A       : BYTE;          (* Offset = 2 *)
    Ref_B       : BYTE;          (* Offset = 3 *)
end
```

where each field has the following interpretation:

**Enablebit** :
The Enable bits are used to enable/disable the overflow/underflow alarm.




**Functions** :
The Functions field holds a 2 digit hexadecimal value, where the most significant digit is used to specify the regulator type and the least significant digit is used to specify a regulator function.
Regulator type specification:

Functions = $0x => Regulator disabled
Functions = $1x => P or PD regulator
Functions = $2x => PID regulator

Regulator function specification:

Functions = $x0 => Heating
Functions = $x1 => Cooling

**Ref_A** : SWNo for Signal @
**Ref_B** : SWNo for Setpoint @

The regulator can be used in four different modes, as a simple P regulator, a PD regulator, a PI regulator or as a PID regulator. The simple P regulator, or PI regulator is achieved by setting the value of the D-component for the regulator to 0.

The regulator function specification, is used to select "Heating" or "Cooling". "Heating" means that if signal > setpoint, PIDOutput will decrease.

If the regulator is not required, then ChConfig.Functions must be set to 0, to avoid possible errors in the channel.

**SWNo BB: Xp**
Xp is the means of defining the proportional band (or gain) of the regulator. It is input as a value, in the same units as the input signal, which if seen as a change at the input, would cause PIDOutput to change by 100%, assuming no I or D components are considered.

**SWNo BC: Ti [s]**
Ti defines the integration time of the I component in the regulator, which is set to a time in seconds, for a stepped change at the input of the same value as Xp, to cause the 100% change in PIDOutput. This has the effect of reducing any offset error, i.e. the difference between the actual control level and the desired setpoint level.

**SWNo BD: Td [s]**
Td defines the differentiation time of the D component in the regulator, which is set to a time in seconds that a constantly rising input signal will take to increase by a value equivalent to Xp, to give a constant output signal of 100%, assuming no proportional or integral components are considered.
If Td is negative, Td has the effect as acting as a "brake" on the output signal, and is most useful in overcoming the undesirable effects of long lags in the control loop, or rapid changes in the controlled condition.

If problems arise in the setting of the three parameters P, I and D, it is advisable to refer to technical literature on the subject.

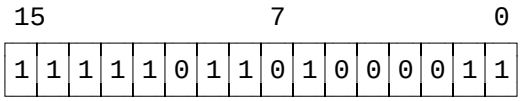
**SWNo BE: ChType**
For the PID regulator channel, ChType is of the following type:

```
Record
    ChannelType: WORD;          (* Offset = 0 *)
    Exist: Bit16;               (* Offset = 2 *)
    Functions: Bit16;           (* Offset = 4 *)
    RegulatorFunction: Bit16;   (* Offset = 6 *)
end
```

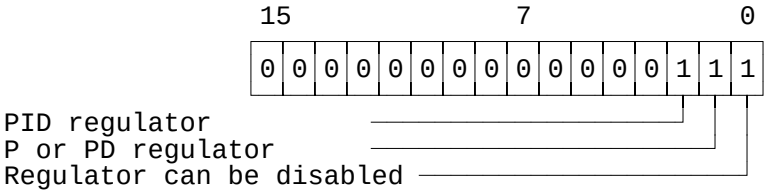
ChType has the following value:

**ChannelType** = 6

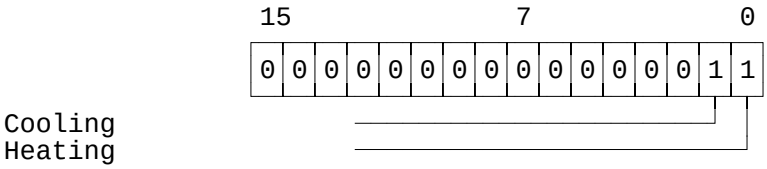
**Exist** =




**Functions** =



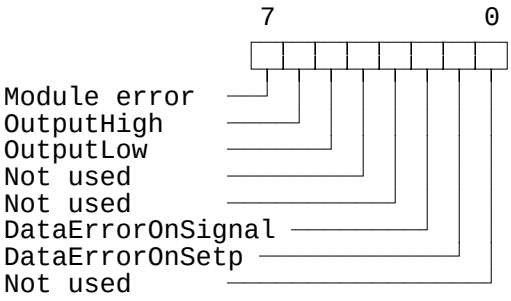

**RegulatorFunction** =




**SWNo BF: ChError**

```
ChError: Record
        His:Array[0..7] of Boolean;   (* Offset = 0 *)
        Act:Array[0..7] of Boolean;   (* Offset = 2 *)
    End;
```

The 8 bits in ChError.His and ChError.Act have the following meaning. When an error occurs the corresponding bit is set in both ChError.His and ChError.Act. When the error disappears, the bit is cleared in ChError.Act.




Bit 7 Module error. If this bit is set, the rest of the bit have no significance, because an error can lead to random error codes in the individual channels (also see "Service channel").

Bit 6 OutputHigh is set if PIDOutput exceeds 100% and ChConfig.Enablebit[6] = TRUE.

Bit 5 OutputLow is set if PIDOutput is below 0% and ChConfig.Enablebit[5] = TRUE.

Bit 4 Not used

Bit 3 Not used

Bit 2 DataErrorOnSignal is set when the signal is an indirect variable, and the variable that is pointed to has an ChError.Act <> 0.

Bit 1 DataErrorOnSetp is set when the setpoint is an indirect variable, and the register that is pointed to has an ChErrror.Act <> 0.

Bit 0 Not used

# 8 Calculator channel (channel C).

The PD 3221 is able to perform arithmetical and boolean functions by means of the calculator channel. All variables within the module can be used in the expressions.

The calculator has a real type accumulator, a boolean type accumulator, two channel pointers, two index registers and a bit index register. The instruction set includes Move, Compare, Jump, logical operations and arithmetical operations. Some of the instructions can operate either on variables via SWNo (channel no. and register no.) or on immediate values.

It is not possible to write into EEPROM by means of the Calculator programme.

Variables for Calculator. Channel identifier: **Calculator**

| SWNo | Identifier | Memory type | Read out | Format |
|---|---|---|---|---|
| C0 | UniversalA | RAM Auto Save | Decimal | Real |
| C1 | UniversalB | RAM Auto Save | Decimal | Real |
| C2 | UniversalC | RAM Init EEPROM | Decimal | Real |
| C3 | UniversalD | RAM Init EEPROM | Decimal | Real |
| C4 | UniversalE | RAM Init EEPROM | Decimal | Real |
| C5 | UniversalF | RAM Init EEPROM | Decimal | Real |
| C6 | UniversalG | RAM Init EEPROM | Decimal | Real |
| C7 | Universal | RAM Read Write | - - - - - | Array20Real |
| C8 | UserTimer | RAM Read Write | Decimal | Real |
| C9 | RunEnable | RAM Init EEPROM | Binary | Boolean |
| CA | LookUp1 | EEPROM RPW | - - - - - | Array20Real |
| CB | LookUp2 | EEPROM RPW | - - - - - | Array10Real |
| CC | LookUp3 | EEPROM RPW | - - - - - | Array10Real |
| CD | ProgramStep | EEPROM RPW | - - - - - | Array200Word |
| CE | ChType | PROM Read Only | - - - - - | Record |
| CF | ChError | RAM Read Only | Binary | Record |

**SWNo C0-C6:UniversalA-UniversalG**
These variables are universal variables of type real, used by the calculator for input/output values. The first 2 variables are automatically saved in EEPROM at a certain predetermined frequency.

**SWNo C7: Universal**
This variable is an array which can hold 20 real values. The structure of the Universal variable can be useful to the calculator programmer, for accessing variables within a program loop, by means of an index pointer. The array is declared as:

ARRAY[0..19] OF REAL

**SWNo C8: UserTimer**
This register holds a timer variable, which can be used by the calculator program. The timer counts down with a resolution of 1/8 second. The count continues through negative values. The timer register is cleared after a power failure or reset. The maximum value for the timer is approximately 97 days. After an overflow, the timer continues from the maximum value.

**SWNo C9: RunEnable**
This variable is used to start and stop program execution in the calculator channel. When RunEnable is set to TRUE the program is able to operate. RunEnable should be set to FALSE before a program is downloaded.

**SWNo CA: LookUp1**
This variable is declared as a lookup table with the following format:

```
Coordinate:     Record
                  X:Real;
                  Y:Real;
                End;

LookUp: Array[1..10] of Coordinate;
```

This variable represents a line through 10 pairs of x,y coordinates. LookUp1 performs a function that returns an interpolated Y value when called with an X value. The X coordinates must be in increasing order. For X-values below X1 the function will return the Y1 value and for X-values higher than X10, it returns Y10.

**SWNo CB-CC: LookUp2 - LookUp3**
These variables are declared as lookup tables with the following format:

```
Coordinate:     Record
                  X:Real;
                  Y:Real;
                End;

LookUp: Array[1..5] of Coordinate;
```

The variables represents each a line through 5 pairs of x,y coordinates. Each LookUp performs a function that returns an interpolated Y value when called with an X value. The X coordinates must be in increasing order. For X-values below X1 the function will return the Y1 value and for X-values higher than X5, it returns Y5.

**SWNo CD:ProgramStep**
This variable is defined as ARRAY[1..200] OF WORD, and holds the calculator program as a number of instructions, operating on various variables. The total number of program steps in a calculator program is 200.

The execution time for each instruction is approximately 1 ms. By disabling the automatic functions at the other channels which are not in use, the operating speed of the calculator program can be increased.

Some typical instructions, and the execution time, are shown below. The conditions for the time measurements are: All the channels in the module are enabled, the digital outputs are configured as 50 % duty cycle output, analog inputs are configured as Pt100, one analog input with filter, current output configured with indirect setpoint, and regulator channel configured as PID. The minimum time is found with little P-NET communication, and the maximum time is found with significant P-NET communication.

| **Instruction** | **Min time** | **Typ. time** | **Max time** |
|---|---|---|---|
| Move CR1:6, Acc | 0.3 ms | 0.8 ms | 1.4 ms |
| Move Acc, CR1:4 | 0.3 ms | 0.8 ms | 1.4 ms |
| Move 4, CR1 | 0.3 ms | 0.5 ms | 1.0 ms |
| Div 123.4 | 0.8 ms | 1.3 ms | 1.9 ms |
| Mul 123.4 | 0.8 ms | 1.3 ms | 1.9 ms |
| Sub 123.4 | 0.6 ms | 1.0 ms | 1.4 ms |
| Add 123.4 | 0.6 ms | 1.0 ms | 1.4 ms |
| LookUp CR1:#A | 5.8 ms | | 57 ms |

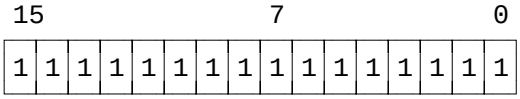
Refer to the PD Calculator Assembler Manual for a list of the available instructions and information on how to programme the calculator.

**SWNo CE: ChType**
For the calculator channel, ChType is of the following type:

```
Record
    ChannelType: WORD;          (* Offset = 0 *)
    Exist: Bit16;               (* Offset = 2 *)
    NoOfProgramStep: WORD;      (* Offset = 4 *)
end
```

ChType has the following value:

**ChannelType** = 7
**Exist** =




**NoOfProgramStep** = 200

**SWNo CF: ChError**
No Error messages will automatically appear in the calculator channel, and therefore ChError normally reads 0. However, the calculator can be programmed to set any of the error bits, to simulate an error condition. This feature can be useful in indicating erroneous data. The register is declared as follows:

```
ChError: Record
            His:Array[0..7] of Boolean;    (* Offset = 0 *)
            Act:Array[0..7] of Boolean;    (* Offset = 2 *)
         End;
```

# 9 Pulse Processor channel (channel D).

The pulse processor is a programmable pulse I/O device which, can utilise the digital I/O channels within the module. A versatile pulse I/O system can be realized for applications such as encoder signal interpretation, pulses generation for stepper motor control, adjustable duty-cycle pulse generation, fast pulse counting and a variety of similar functions.

The pulse processor utilizes 24 16-bit universal registers, each one selectable as counter register, compare register, capture register or timer register. Furthermore it utilizes a 24 bit I/O register for pulse I/O control. Some of these I/O bits are connected to the OutFlags and InFlags in the digital channels. The rest of the I/O registers are accessible by both the pulse processor and the Calculator program. The pulse processor automatically performs complex pulse control, using a selection of 15 different instructions (functions), which dramatically reduces the burden on the main microprocessor in the module. The pulse width resolution, which is derived from the pulse processor internal clock, is 5 µs or better, depending on the number of functions selected.

The pulse processor occupies one channel in the module. This channel is used as a direct interface to some of the universal registers, the internal I/O register bits, the PROGRAM instructions, and access to all the universal registers by means of an array variable.

Registers for Pulse Processor. Channel identifier: **PulseProcessor**

| SWNo | Contents | Memory type | Read out | Format |
|---|---|---|---|---|
| D0 | PPReg0 | RAM Read Write | Decimal | Word |
| D1 | PPReg1 | RAM Read Write | Decimal | Word |
| D2 | PPReg2 | RAM Read Write | Decimal | Word |
| D3 | PPReg3 | RAM Read Write | Decimal | Word |
| D4 | PPReg4 | RAM Read Write | Decimal | Word |
| D5 | PPReg5 | RAM Read Write | Decimal | Word |
| D6 | PPReg6 | RAM Read Write | Decimal | Word |
| D7 | PPReg7 | RAM Read Write | Decimal | Word |
| D8 | PPReg8 | RAM Read Write | Decimal | Word |
| D9 | RunEnable | RAM Init EEPROM | Binary | Boolean |
| DA | Registers | RAM Read Write | Decimal | Array24Word |
| DB | IOFlags | RAM Read Write | Binary | Bit24 |
| DC | NoOfInst | EEPROM RPW | Decimal | Byte |
| DD | PPProgram | EEPROM RPW | - - - - - | Array56Word |
| DE | ChType | PROM Read Only | - - - - - | Record |
| DF | ChError | RAM Read Only | Binary | Record |

The pulse processor is programmable with up to 16 program steps, which are complete functions which are stored in an internal function table. The pulse processor program can be written, edited and downloaded by means of a PC based utility program, designed by Proces-Data Silkeborg ApS.

**SWNo D0-D8: PPReg0 - PPReg8**
These variables are the first 9 universal registers in the pulse processor. The registers can be accessed via the P-NET and directly by the pulse processor. Each register is programmable as counter register, compare register, capture register, data register or timer register.

A register can be specified by more than one function. For example, a register which is specified as a counter by one function, can be specified as a compare register by another function.

A COUNTER register counts clocks specified by the I/O assignment within the program step. The counter can count on a rising edge, a falling edge or both edges. The value of the counter can be decremented or incremented, depending on the specified function and operating mode.

A COMPARE register is used to make a comparison between itself and another specified register. Compare can be performed using ( < ) or ( >= ) operators. A comparison result can be related to a specific flag bit, e.g. an output.

A CAPTURE register is used to latch a value from another specified register, when a specific polarity transition is detected on a specific flag bit.

A DATA register is used in shift functions. Functions can be selected to shift bits into a register, shift bits out of a register, rotate within a register, or simultaneously transfer the contents of a part of a register to a section of the I/O flag register.

A TIMER register value represents a count of internal clock periods. This time period (pulse width resolution) is determined by the cycle time of the pulse processor program, which in turn depends on the number of program steps used. The pulse width resolution is inversely proportional to the internal clock frequency. This value is automatically calculated and displayed in the Pulse Processor Program Editor. The value of a timer can be decremented or incremented, depending on the specified function.

**SWNo D9: RunEnable**
This variable is used to start or stop program execution of functions held in the pulse processor function table. When RunEnable is set TRUE the program is running.

**SWNo DA: Registers**
All 24 universal registers in the pulse processor can be accessed in this array. Register0 to Register8 can also be directly accessed in SWNo $D0-$D8.

**SWNo DB: IOFlags**
The 24 bit I/O registers (bits 0 to 23) in the pulse processor, used for pulse I/O control, can be accessed by the calculator and the P-NET by using this variable.

The higher 8 bits (bit 16 to bit 23), have no external input/output connection, therefore pulses cannot be input to or output from them. However, these bits can be used as a handshake between the pulse processor and calculator programs or P-NET. To avoid any problems in accessing these bits, it is advisable that the pulse processor program is designed to only read the state of these bits, except in specific functional circumstances. These bits are read/write.

Bit 0 to 15 holds the digital output and input flags, and a number of accessible bits for the pulse processor. These bits are READ ONLY when accessed by the Calculator channel and by P-NET.

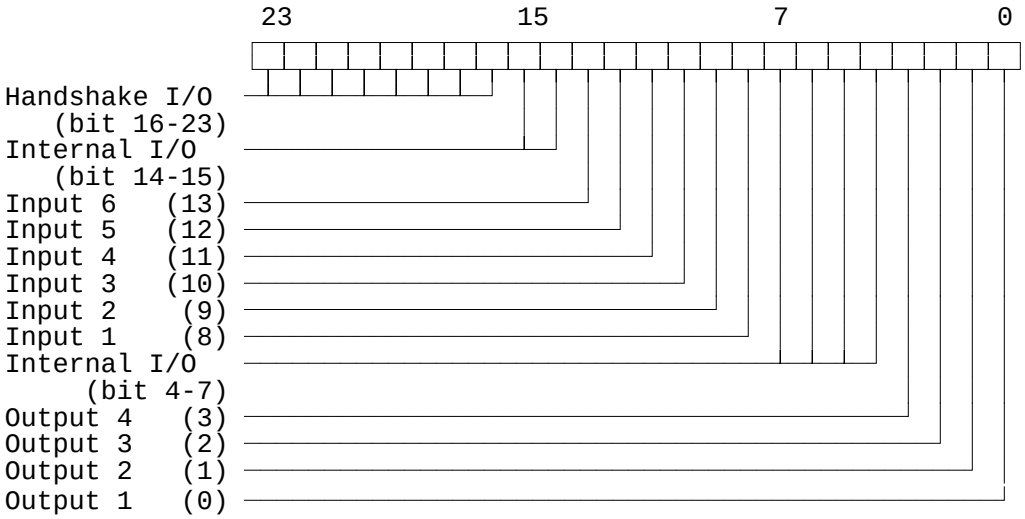
Bit 8 to 13 hold the data from the input pins, corresponding to the InFlags in channels 1 to 6. These bits do always correspond to the state of the input signals.

Bit 0 to 3 holds the data from the output pins, corresponding to the OutFlags in channels 1 to 4. These bits do always correspond to the state of the output signals.

Bit 4 to 7 and bit 14 to 15, have no external input/output connection, hence pulses cannot be input to or output from them. However, these bits can be used to relay signals within the pulse processor program, so the output from one function can be used as input to another function.

When creating a pulse processor program, only these bit numbers should be used when accessing module inputs and outputs, handshake and internal I/O bits.

The structure of the 24 bit I/O registers is shown below.




**SWNo DC: NoOfInst**

This variable holds the total number of instructions which are to be executed by the pulse processor. The number of functions to be executed and the pulse width resolution are determined by this value. Refer to the Pulse Processor Program Editor for this value.

**SWNo DD: PPProgram**

The program for the pulse processor is stored in this array variable. The pulse processor program consists of up to 16 program steps. Each step uses 7 bytes of memory. The program should only be downloaded to the pulse processor function table following a reset.

Refer to the Pulse Processor Program Editor Manual (Manual no. 50 20 65) for a list of the available functions and information on how to program the pulse processor.

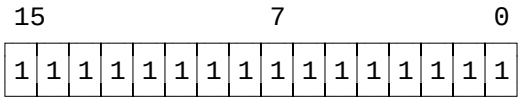
**SWNo DE: ChType**

For the pulse processor channel, ChType is of the following type:

```
Record
    ChannelType: WORD;          (* Offset = 0 *)
    Exist: Bit16;               (* Offset = 2 *)
end
```

ChType has the following value:

ChannelType = 8
Exist =




**SWNo DF: ChError**

No Error messages appear in ChError in the pulse processor channel, and will therefore always contain a value of zero. The register is declared as follow.

```
ChError: Record
            His:Array[0..7] of Boolean;    (* Offset = 0 *)
            Act:Array[0..7] of Boolean;    (* Offset = 2 *)
         End;
```

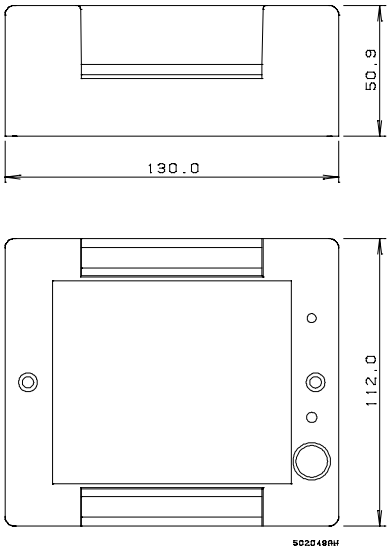
## 10 Construction, Mechanical.

The PD 3221 module is housed in a black plastic case. The case measures W x H x D = 130.0 x 112.0 x 50.9 mm (tolerance to DIN 16901 ).

The module is designed for plugging directly on to a mounting rail (EN 50 022 / DIN 46277). The module incorporates two snap connectors, which provide the terminals for field connection, power and communications.

The module may be DIN rail mounted for a panel mounted configuration and contained in a sealed box designed for the plant environment. It may be removed for service, without interfering with operational activities on the rest of the network.

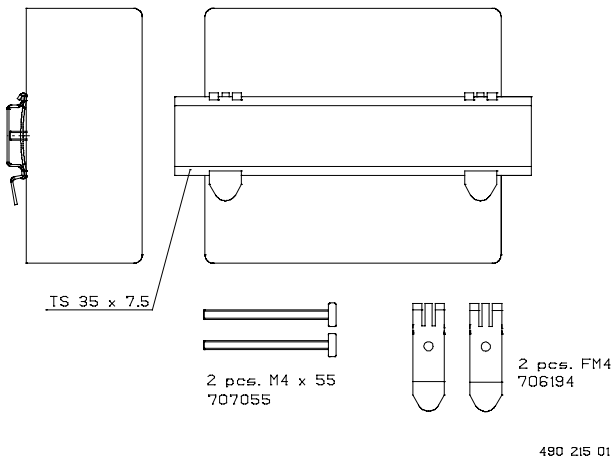
**Scale drawing (in mm):**




**Materials**

| | |
|---|---|
| Case | : Black NORYL GFN ( injection moulded ) |
| Front foil | : Polycarbonate. |
| Back plate | : Black anodized aluminium. |
| Weight | : 400 gram. |

**Rail mounting:**



# 11 Specifications.

All electrical characteristics are valid at an ambient temperature -25 °C - 70 °C, unless otherwise stated.

## 11.1 Power supply.

| | | |
|---|---|---|
| Power supply DC: | nom. | 24.0 V |
| | min. | 20.0 V |
| | max. | 28.0 V |
| Ripple : | max. | 5 % |
| Power consumption (All outputs = OFF) : | max. | 1.3 W |
| Power consumption (All outputs = ON ) : | max. | 3.0 W |
| Current at power up : | max. | 250 mA |

Fuse 5 A time delay.

## 11.2 Digital Input.

| | | |
|---|---|---|
| Input voltage at ON : | | < 4.1 V |
| Input voltage at OFF : | | > 15.1 V |
| Hysteresis : | min. | 3.2 V |
| Input current at ON : | max. | 5.0 mA |
| Counter frequency for counter in channel : | max. | 50 Hz |
| Counter frequency input controlled by the pulse processor : | max. | 100 kHz |

## 11.3 Digital Output.

| | | |
|---|---|---|
| Load current at ON ( Sink ) : | max. | 1.0 A |
| Leakage current at OFF : | max. | 0.5 mA |
| Short circuit cut off delay time (cut off at outputcurrent > 2 A): | max. | 10 ms |
| One-shot and duty-cycle resolution : | | 125 ms |
| Feedback check time : | | 500 ms |

Load current measurement :

| | |
|---|---|
| Accuracy : | ± 19 mA |
| Resolution : | 12.5 mA |
| Repeatability : | ± 12.5 mA |
| Current measurement update time : | 125 ms |

## 11.4 Analog input.

Current input (0-20mA, 4-20mA):

| | | |
|---|---|---|
| Measurement error: | | |
| @ 0°C to +50°C: | max. | ± 0.1 % of actual current ± 2 µA |
| @ -25°C to +70°C: | max. | ± 0.3 % of actual current ± 2 µA |
| Resolution : | | 1 µA |
| Repeatability : | | ± 2 µA |
| Voltagedrop across input at 20 mA : | max. | 1.4 V |
| Current input measurement update time (one channel enabled) : | | 0.6 s |
| Current input measurement update time (two channels enabled) : | | 0.9 s |

Voltage input (0-100 mV):

| | | |
|---|---|---|
| Measurement error: | | |
| @ 0°C to +50°C: | max. | ± 0.1 % of actual voltage ± 10 µV |
| @ -25°C to +70°C: | max. | ± 0.3 % of actual voltage ± 10 µV |
| Resolution : | | 5 µV |
| Repeatability : | | ± 10 µV |
| Input impedance : | min. | 5.0 MΩ |
| Voltage input measurement update time (one channel enabled) : | | 0.6 s |
| Voltage input measurement update time (two channels enabled) : | | 0.9 s |

Temperature input with Pt-100 sensor (IEC 751). Specifications exclude the accuracy of the Pt-100 sensor.

| | | |
|---|---|---|
| Accuracy @ -100 °C : | max. | ± 0.19 °C |
| Accuracy @ 20 °C : | max. | ± 0.29 °C |
| Accuracy @ 200 °C : | max. | ± 0.51 °C |
| Resolution : | | 0.05 °C |
| Repeatability : | | ± 0.1 °C |
| Temperature range : | | -100 °C - 200 °C |
| Temperature input measurement update time (one channel enabled) : | | 0.6 s |
| Temperature input measurement update time (two channels enabled) : | | 0.9 s |
| Power dissipation in temperature detector : | max. | 0.05 mW |

## 11.5 Analog output.

Current output (4-20 mA):

| | | | |
|---|---|---|---|
| Accuracy : | max. | ± 0.22 % of actual current | ± 19.2 µA |
| Output voltage at 20 mA : | | min. | 5.0 V |
| Resolution : | | | 15.3 µA |
| Repeatability : | | | ± 20 µA |

Current output update time :

| | |
|---|---|
| No analog input channels enabled : | 0.5 s |
| One analog input channel enabled : | 0.6 s |
| Two analog input channels enabled : | 0.9 s |

## 11.6 Ambient Temperature.

| | |
|---|---|
| Operating temperature : | -25 °C - 70 °C |
| Storage temperature : | -40 °C - 85 °C |

## 11.7 Humidity.

| | | |
|---|---|---|
| Relative humidity : | max. | 95 % |

## 11.8 Approvals.

| | |
|---|---|
| Compliance with EMC-directive no.: | 89/336/ECC |

Generic standards for emission:

| | |
|---|---|
| Residental, commercial and light industry | EN 50081-1 |
| Industry | PrEN 50081-2 |

Generic standards for immunity:

| | |
|---|---|
| Residental, commercial and light industry | EN 50082-1 |
| Industry | PrEN 50082-2 |

| | |
|---|---|
| Vibration (sinusoidal): | IEC 68-2-6 Test Fc |

# 12 Survey of variables in the PD 3221 module.

| SWNo | Service channel 0 | Digital I/O 1–4 | Digital input 5–6 | Common I/O 1–6 7 | Analog input 8–9 | Current output A | PID-regulator B | Calculator C | Pulse Processor Run D |
|---|---|---|---|---|---|---|---|---|---|
| x0 | NumberOfSWNo | FlagReg | FlagReg | OutFlags | AnalogIn | AnalogOut | PIDOutput | UniversalA | PPReg0 |
| x1 | DeviceID | OutTimer | | InFlags | | | Signal | UniversalB | PPReg1 |
| x2 | | Counter | Counter | | | | | UniversalC | PPReg2 |
| x3 | Reset | OutCurrent | | | | | | UniversalD | PPReg3 |
| x4 | PnetSerialNo | OperatingTime | OperatingTime | | | | | UniversalE | PPReg4 |
| x5 | | | | | | | | UniversalF | PPReg5 |
| x6 | | FBTimer | | | | SetPoint | SetPoint | UniversalG | PPReg6 |
| x7 | FreeRunTimer | FBPreset | | | HighLevel | HighLevel | | Universal | PPReg7 |
| x8 | WDTimer | OutPreset | | | LowLevel | LowLevel | Manual | UserTimer | PPReg8 |
| x9 | ModuleConfig | ChConfig | ChConfig | | ChConfig | ChConfig | ChConfig | RunEnable | RunEnable |
| xA | WDPreset | MinCurrent | | | | | | LookUp1 | Registers |
| xB | | MaxCurrent | | | FullScale | FullScale | Xp | LookUp2 | IOFlags |
| xC | | | | | ZeroPoint | ZeroPoint | Ti | LookUp3 | NoOfInst |
| xD | WriteEnable | Maintenance | | | Maintenance | Maintenance | Td | ProgramStep | PPProgram |
| xE | ChType | ChType | ChType | ChType | ChType | ChType | ChType | ChType | ChType |
| xF | CommonError | CHError | CHError | IOChError | CHError | CHError | CHError | CHError | ChError |

502060AD

## Contents

Page