502 067 01

Digital I/O

PD 3120

Manual

GB *December 1993*

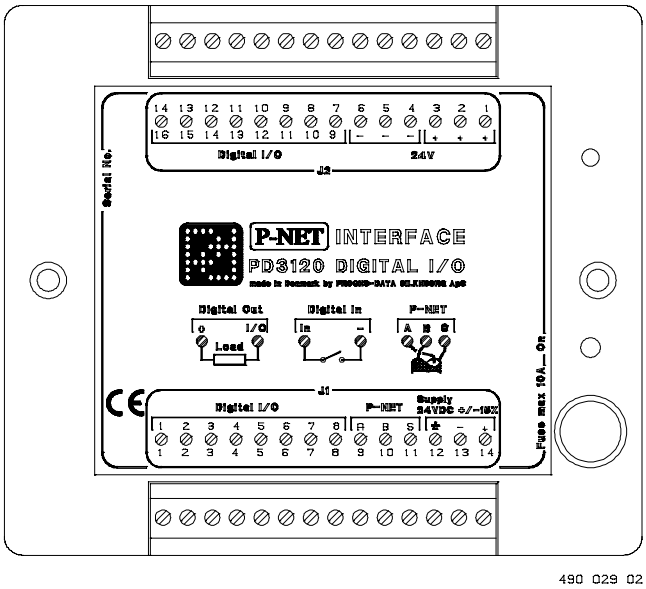
Proces-Data Silkeborg ApS, Navervej 10, DK-8600 Silkeborg, Denmark, phone +45-86814033, fax +45-86814088

Contents

1 General information.

The PD 3120 Digital I/O is a member of Proces-Data's module series 3000.

It provides a versatile interface between valves, switches, pulses, lamps, alarms, motors, level detectors, etc., and distributed master control computers. The PD 3120 Digital I/O Module is an intelligent module, provided with 16 input/output channels for 24 VDC, an interface for the P-NET Fieldbus and an internal programmable calculator for local control.



490 029 02

The P-NET is a field-bus network designed for process control and data-collection. The PD 3120 module has been developed for interfacing directly to digital 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 3120 module can be controlled via the P-NET, or it can operate as an autonomous unit.

The module possesses a programmable Calculator, which can be purpose programmed to control the digital outputs and monitor the digital inputs. The Calculator operates with different types of variables, such as reals, integers, bytes, booleans, timers and arrays. With the utilization of user programmes, application specific functions, such as digital control loop and PLC functions may be set up for use in a wide variety of local autonomous process applications.

The compact design and the outstanding environmental specifications for the Digital I/O module, makes it an ideal process component in industrial as well as other environments.

1.1 Features.

- 16 Digital Input/Output Channels at 24 VDC 1 Amp.
- Pulse or Contact Counting
- Pulse and one-shot on all outputs
- Output Feedback Facility
- Automatic output Functions
- Overload Protection
- Current measuring on each output
- Programmable Calculator
- Internal Data Channel
- Continuous Selftest, which can be monitored through the P-NET.
- P-NET Fieldbus Communication
- Watch Dog Timer
- Rail mounting module (DIN / EN)
- EMC approved (89/336/ECC)

1.2 System description.

Various automatic functions can be selected on each digital channel, such as automatic feedback control (single as well as double), one-shot output and pulse output, to reduce the basic operations from the central control system or enable the unit to operate autonomously. The unit offers comprehensive self-testing features, which enables reporting of disconnection, overload and process failure. All outputs are protected against overload. The watchdog timer ensures the safe shut down of a process following a communication error or power failure.

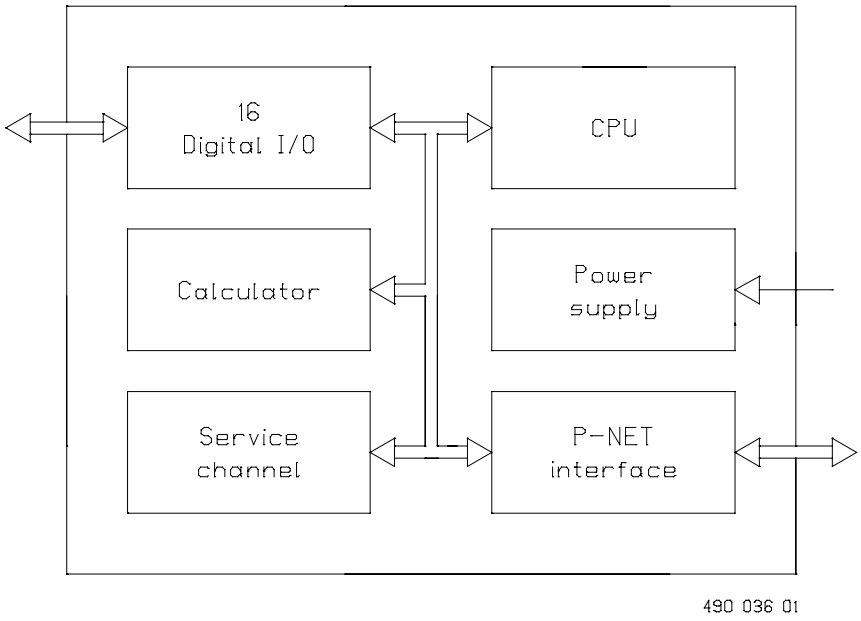
The output current (Sink current) is measured continuously on each channel and can be read as a value in Amps. If the current exceeds the specified max value, the output is switched off and an errorcode (overload) is generated in the module. This feature may be used on purpose, eg. to open a window using a DC motor, where the motor is stopped automatically when reaching the end position because of the increasing current in the motor.

Each channel is automatically summarizing OperatingTime as a value in seconds and counting the number of pulses on the input. Data for maintenance may be stored directly on each channel.
A common channel in the module provides the possibility to read/set all inputs/outputs or errorflags in one P-NET transmission.

PD 3120 is approved in compliance with the **EMC-directive no 89/336/ECC**. Test limits are determined by the generic standards **EN 50081-1** for emission and **PrEN 50082-2** for immunity.
PD 3120 is approved in compliance with the **IEC 68-2-6 Test Fc** standard for vibration.

1.3 Channels/registers.

The PD 3120 module contains:



1 Service channel	(channel 0)	16 Digital I/O's	(channel 1-$10)
1 Common I/O channel	(channel $11)	1 Program channel	(channel $12)
1 Data channel	(channel $13)		

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).

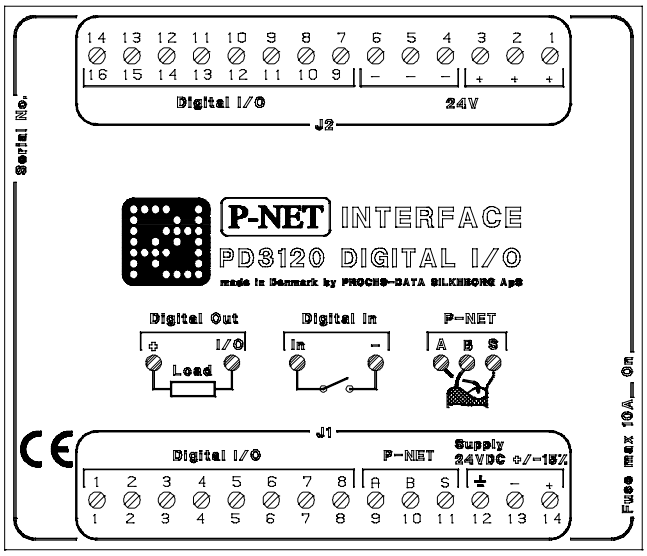
Example: Variable 4 on channel 3 needs to be addressed.
The SWNo will therefore be $34.

Throughout the manual the variables are depicted as tables. The variable names are standard identifiers, as defined in Process-Pascal.

The channel names are depicted on the corresponding channel tables as standard identifiers, defined in Process-Pascal.

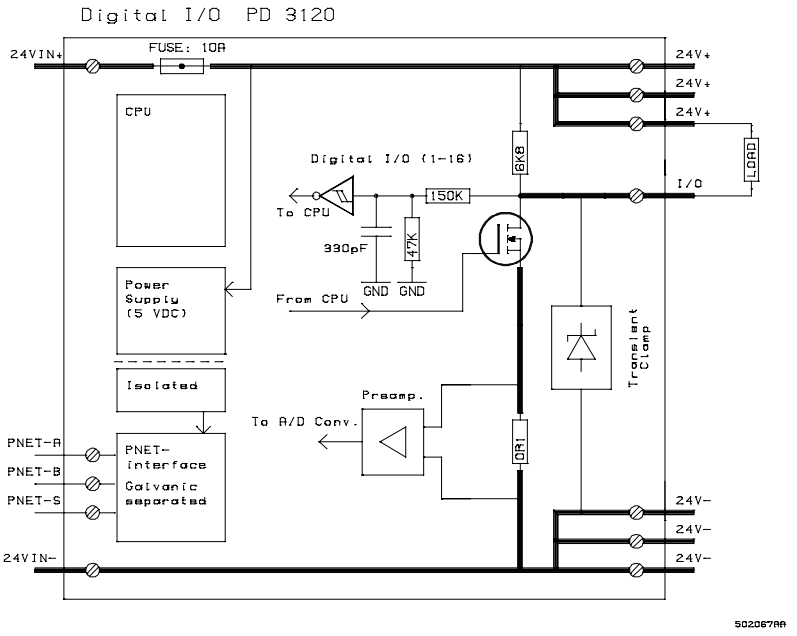
1.4 Connections.

The PD 3120 Digital I/O 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.



490 030 02

Hardware diagram, principle.



1.5 Memory types.

The PD 3120 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.

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.

After copying to EEPROM, a new sumcheck value is calculated. The new sumcheck value is stored in the EEPROM, unless an EEPROM error was set already (CommonError.ChError.Act[2]).

2 Service channel.

PD 3120 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 = 3120
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 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 3120 Digital I/O 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 watchdog is activated, is defined in seconds, in this register.

SWNo $D: WriteEnable

Write protected variables can only be changed when WriteEnable is TRUE. 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 0.25 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

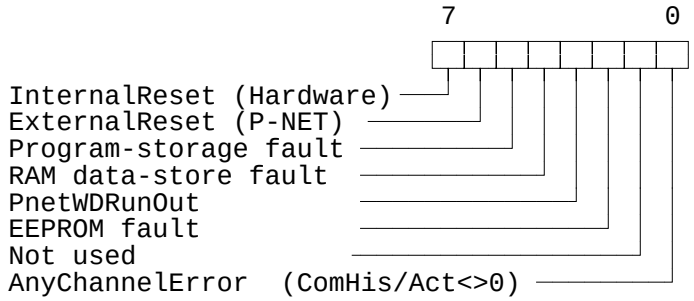
No Watchdog
Watchdog

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;
    ComHis24:Array [0..23] of Boolean;         (* Offset = 4 *)
    ComAct24:Array [0..23] of Boolean;         (* Offset = 8 *)
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.

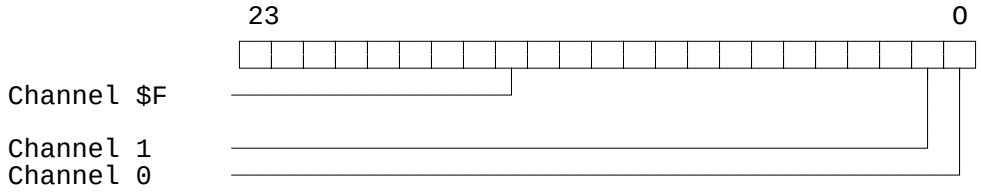
Bit 2 EEPROM fault is set to TRUE if the self test finds an error in the data memory (EEPROM). The error may be corrected by setting and resetting WriteEnable.

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.His in all channels.

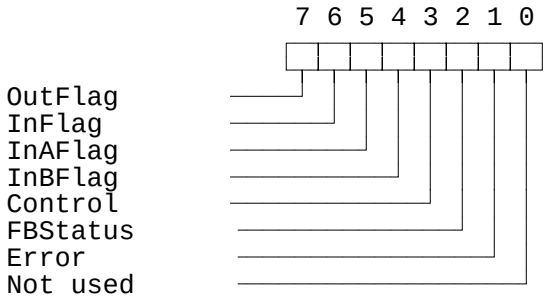


3 Digital I/O channel (channel 1 - $10).

Variables on digital I/O 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 AutoSave	Decimal	Longinteger	- - -
x3	OutCurrent	RAM Read Write	Decimal	Real	A
x4	OperatingTime	RAM AutoSave	Decimal	Real	s
x5	UserByteArray	RAM Init EEPROM	- - - - - -	Array4Byte	- - -
x6	FBTimer	RAM Read Write	Decimal	Real	s
x7	FB Preset	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	UserRealArray	EEPROM RPW	- - - - - -	Array2Real	- - -
xD	Maintenance	EEPROM RPW	- - - - - -	Record	- - -
xE	ChType	PROM Read Only	- - - - - -	Record	- - -
xF	ChError	RAM Read Only	Binary	Record	- - -

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. The input flag is FALSE after reset in simulation mode.

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 Bit 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 timer output) 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.

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 this Error bit 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 1/32 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 0.125 sec.

SWNo x5: UserByteArray
This variable may be used for temporary values or special configuration parameters, in particular in connection with a calculator program. The variable has no effect on the standard functions for a digital I/O channel.

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.

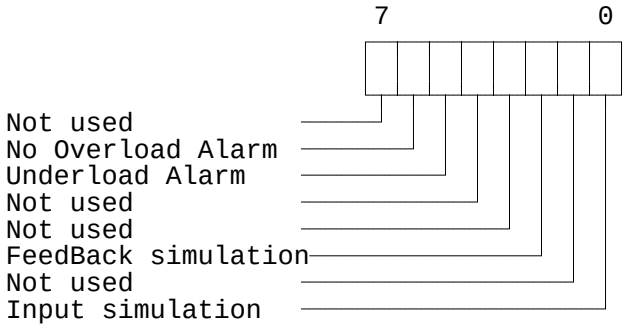
Feedback control: The correct feedback state is Input A = Output and Input B = NOT Output. The feedback inputs can be disabled by writing a 0 as the channel number, in ChConfig.Ref_A and/or ChConfig.Ref_B fields, if only one or no input channels are required.

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 = $50 => Timer Output

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 ON, 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*)
  Begin
    OutTimer:=OutPreset;
    FlagReg[OutFlag]:=True
  End;
  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
```

Timer output.
This automatic function is selected by setting ChConfig.Functions = $50. When the Control Flag is TRUE, the output will be set true if OutTimer is greater than zero. 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 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
```

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 may 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 ChConfig.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 ChConfig.EnableBit[6] then
  Begin
    ChError.Act[OverLoad]:= true;
    ChError.His[OverLoad]:= true;
  End
End
```

SWNo xC: UserRealArray
This variable may be used for special configuration parameters, in particular in connection with a calculator program. The variable has no effect on the standard functions for a digital I/O channel.

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:

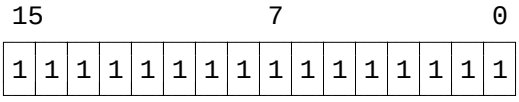
```
Record
    Date      : BYTE;        (* Offset = 0 *)
    Month     : BYTE;        (* Offset = 1 *)
    Year      : BYTE;        (* Offset = 2 *)
    Type      : BYTE;        (* Offset = 3 *)
end
```

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

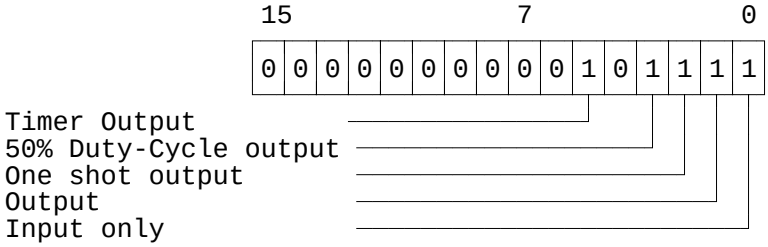
```
Record
    ChannelType: WORD;       (* Offset = 0 *)
    Exist: Bit16;            (* Offset = 2 *)
    Functions: Bit16;        (* Offset = 4 *)
    FeedBack: Bit16;         (* Offset = 6 *)
end
```

ChType has the following value:

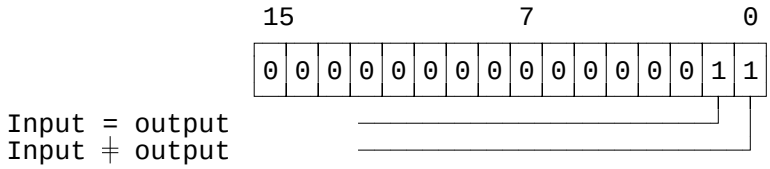
ChannelType = 2
Exist =



Functions =



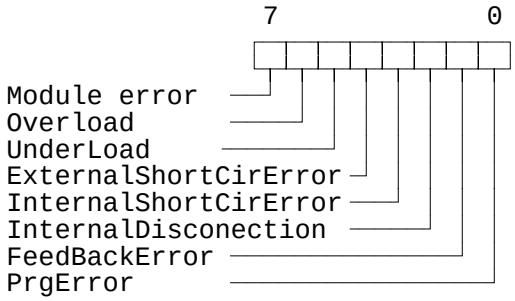
Feedback =



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 by setting ChConfig.Enablebit[6] TRUE.
ChError.Act[OverLoad] remains set until a **Write** operation is performed on the **FlagReg** variable.

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 (OutFlag=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).

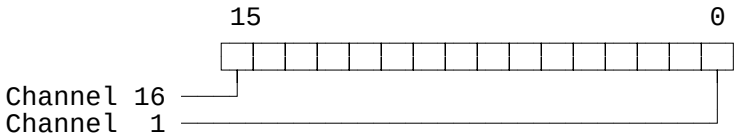
Bit 0 PrgError is set following attempts to set FlagReg[OutFlag] if the I/O is configured to be an input, or following attempts to set FlagReg[Control] if the I/O is configured to be an input or an output without any automatic functions. ChError.Act[PrgError] remains set until a **Write** operation is performed on the **FlagReg** variable.

4 Common I/O channel (channel $11).

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

SWNo	Identifier	Memory type	Read out	Type
110	OutFlags	RAM Read Write	Binary	Bit16
111	InFlags	RAM Read Write	Binary	Bit16
112	IOChError	RAM Read Only	Binary	Record
113				
114				
115				
116				
117				
118				
119				
11A				
11B				
11C				
11D				
11E	ChType	PROM Read Only	- - - - - -	Record
11F	ChError	RAM Read Only	Binary	Record

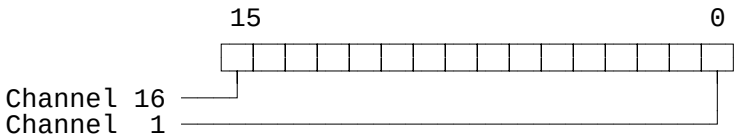
SWNo $110: OutFlags



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.

NOTE: When setting an outflag TRUE via the Common I/O channel, the FlagReg[FBStatus] will not be set TRUE and the FBTimer will not be preset on the corresponding channels.

SWNo $111: InFlags

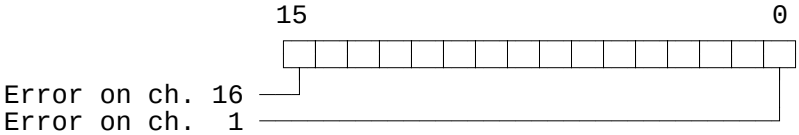


This variable contains all the InFlag's from all I/O channels. This means that all digital inputs in the module can be read in this register.

SWNo $112: IOChError
The IOChError is a variable of the following type:

```
IOChError: Record
             His:Array[0..15] of Boolean;  (* Offset = 0 *)
             Act:Array[0..15] of Boolean;  (* Offset = 2 *)
           End;
```

Meaning of IOChError.His and IOChError.Act:



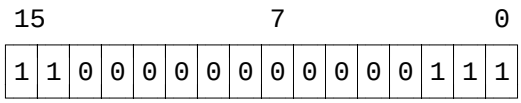
The IOChError variable indicates if there is an error, or an unacknowledged error, in one or more I/O channels.
IOChError.His is set if ChError.His of any channel <> 0 and IOChError.Act is set if any channel contains a ChError.Act <> 0.
Reading IOChError does not acknowledge the channel errors. This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

SWNo $11E: ChType
For the common channel, ChType is of following type:

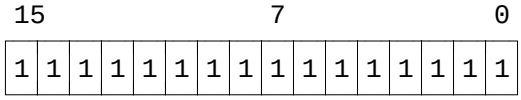
```
Record
    ChannelType: WORD;          (* Offset = 0 *)
    Exist: Bit16;               (* Offset = 2 *)
    ExistingChannels: Bit16;    (* Offset = 6 *)
end
```

The Common I/O Channel is a company specific channel, where ChType has the following value:

ChannelType = $8002
Exist =



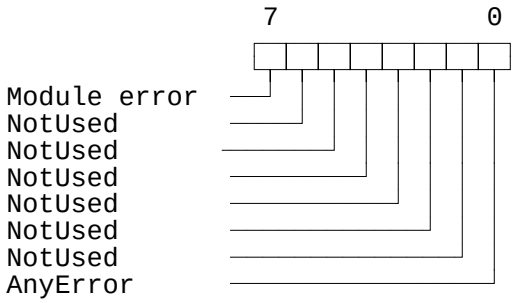
ExistingChannels =



SWNo $11F: 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 0 AnyError is set if there is an error, or an unacknowledged error, in one or more I/O channels.
ChError.His[0] is set if any bit is set in IOChError.His, indicating any historical error on any I/O channel.
ChError.Act[0] is set if any bit is set in IOChError.Act indicating any actual error on any I/O channel.
Reading ChError does not acknowledge the channel errors. This can only be done by reading ChError in the individual channels, or by means of SWNo $F (channel 0).

5 Calculator program.

The PD 3120 is able to perform arithmetical and boolean functions by means of the calculator program. 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 directly into EEPROM by means of the Calculator program. However, it is possible to save all variables with memory type **RAM AutoSave** to EEPROM directly from the Calculator program. The variables on the DataChannel may be used for common configuration parameters and temporary values.

The calculator program is stored (download and upload) and controlled (start, stop etc.) on the Program Channel (channel $12). The program may be downloaded from a PC, eg. the Calculator Assembler program or from a master controller unit on the P-NET, eg. PD 3000 with a Process-Pascal program.

The execution time for each instruction is approximately 0.3 ms. By disabling the automatic functions at the channels which are not in use, the operating speed of the calculator program may be sligthly 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 and all the control bit are set. The minimum time is found with little P-NET communication, and the maximum time is found with significant P-NET communication.

Instruction	**Min time**	**Max time**
Move CR2:0, Acc	0.22 ms	0.24 ms
Move Acc, CR2:7[IR2]	0.30 ms	0.32 ms
Move #13, CR2	0.14 ms	0.14 ms
Div 123.4	0.34 ms	0.35 ms
Mul 123.4	0.32 ms	0.33 ms
Sub 123.4	0.27 ms	0.29 ms
Add 123.4	0.26 ms	0.28 ms
LookUp CR2:#C (each index pair)	1.70 ms	1.80 ms

Refer to the PD Calculator Assembler Manual, PD no. **502 061**, for a list of the available instructions and information on how to program the calculator.

6 Program Channel (channel $12).

The Program Channel utilizes Program Invocation Management and specifies some general information for a Calculator program in the PD 3120. The Calculator program is an application program, which can be downloaded or uploaded via this channel and the status for the program can be supervised and changed.

The program in the library is indexed by a number, LibraryIndex, which is found in Library-Control. The total number of programmes, which can be stored in the library-memory is stated in MaxLibraryIndex, which is found in LibraryStatus. The total number of programmes in the PD 3120 module is 1.

The selected program can be controlled and supervised. The commands and program-states conform to the instructions specified in Manufacturing Message Specification (MMS).

Registers on Program Channel (channel n). Channel identifier: **Calculator**

SWNo	Identifier	Memory Type	Read Out	Type
120	ProgramControl	RAM Read Write	- - - - - - -	Record
121	ProgramStatus	RAM Read Only	- - - - - - -	Record
122	ProgramID	Read Only	- - - - - - -	Record
125	SystemPointer	Read Only	Hex	LongInteger
127	MemoryInfo	Read Only	- - - - - - -	Record
128	IDAndCode	Special function	- - - - - - -	Record
129	ChConfig	EEPROM RPW	- - - - - - -	Record
12A	LibraryControl	RAM Read Write	- - - - - - -	Record
12B	LibraryStatus	RAM Read Only	- - - - - - -	Record
12C	LibraryProgramID	Read Only	- - - - - - -	Record
12D	Maintenance	EEPROM RPW	- - - - - - -	Record
12E	ChType	PROM Read Only	- - - - - - -	Record
12F	ChError	RAM Read Only	- - - - - - -	Record

SWNo $120: ProgramControl

ProgramControl is used to set and change state for the current program which has been selected and invoked via the Program Channel. The selected program number is inserted and indicated as a part of ProgramControl. Commands can be sent to ProgramControl to stop/start or reset the program.

ProgramControl is a record of the following type:

```
Record
    Command         : BYTE;     (* Offset = 0 *)
    ProgramToSelect : Word;     (* Offset = 2 *)
    ErrorStatus     : Bit32;    (* Offset = 4 *)
End
```

Command is used to send a command to the Program Channel for changing the state of the current program. A list of possible commands is given below. The commands and the corresponding numbers conform to the Request Instructions used by MMS.

Command		**Purpose**
38	SelectProgram	Selects a program from the library, resulting in ProgramStatus.State = Idle if the program is OK. (MMS = CreateProgramInvocation)
39	UnSelectProgram	Sets SelectedProgram to 0, resulting in ProgramStatus.State = Non-selected. (MMS =DeleteProgramInvocation)
40	Start	Starts the selected program. (MMS=Start)
41	Stop	Stops the selected program. (MMS=Stop)
42	Resume	Continues program execution in the selected program. (MMS=Resume)
43	Reset	Resets the selected program. (MMS=Reset)
44	Kill	Stops the selected program instantly and sets the ProgramStatus.State = Unrunable. (MMS=Kill)

Command is automatically set to 0 by the Program Channel after writing to the variable. ProgramStatus.State is updated immediately to one of the corresponding temporary states, Starting, Stopping, Resuming or Resetting each time a command is sent to the Command variable. By the change in state it is possible to read the variable ProgramStatus.State to check if the Command was executed successfully or the operation failed. If an error occurs following a Start or Resume command, the specific error may be read in ChError.

ProgramToSelect holds the library index for the program to select or the already selected program. ProgramToSelect is copied from ChConfig following a module reset or power up.

ErrorStatus does not indicate any errors in the application program and consequently always reads FALSE for all bits.

SWNo $121: ProgramStatus

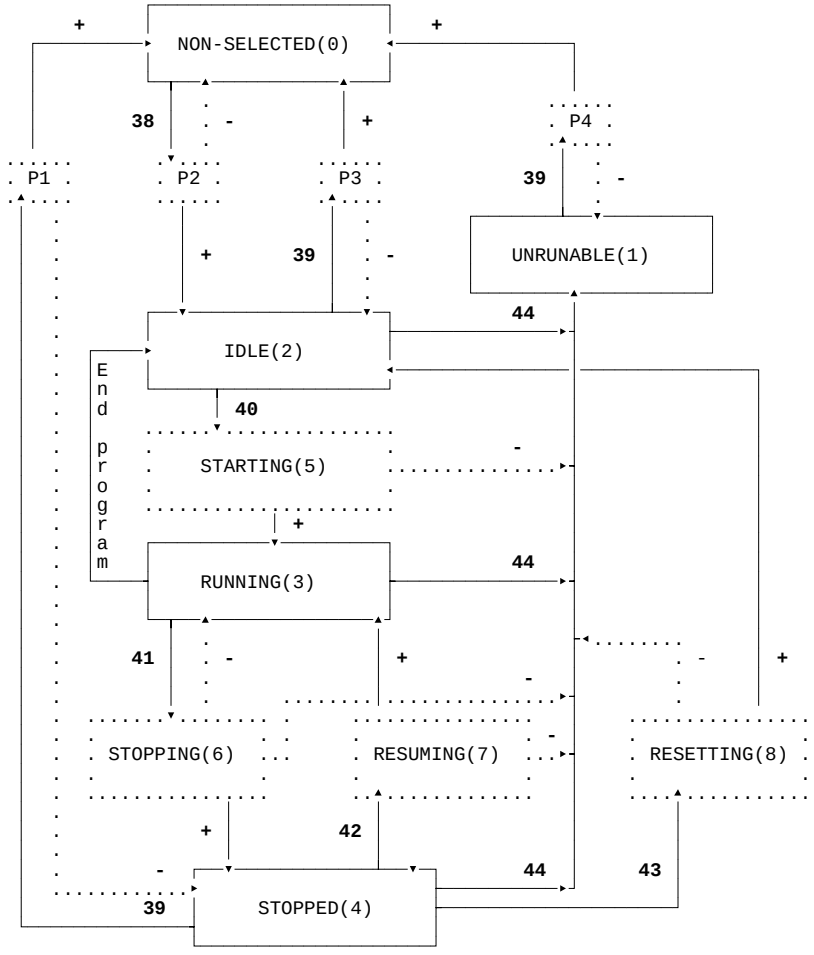
ProgramStatus summarizes the state and error conditions for the selected program. The library index for the selected program is also indicated.

ProgramStatus is a record of the following type:

```
Record
    State           : BYTE;     (* Offset = 0 *)
    SelectedProgram : Word;     (* Offset = 2 *)
    ErrorStatus     : Bit32;    (* Offset = 4 *)
End
```

State indicates the current state for the selected program, eg. stopping, stopped, running, idle, non-selected etc. A state diagram and a list of possible states is given below. The states and the corresponding numbers conform to the states for Program Invocation Management used by MMS.

PROGRAM INVOCATION STATE DIAGRAM



\+ State transition succeeded - State transition failed.

State		Explanation
0	Non-selected	No program selected. (MMS = Non-selected)
1	Unrunable	The program can not run. (MMS = Unrunable)
2	Idle	The program is stopped and reset. (MMS = Idle)
3	Running	The program is running. (MMS = Running)
4	Stopped	The program is stopped. (MMS = Stopped)
5	Starting	The program is changing state from idle to running. (MMS = Starting)
6	Stopping	The program is changing state from running to stopping. (MMS = Stopping)
7	Resuming	The program is changing state from stopped to running. (MMS = Resuming)
8	Resetting	The program is changing state from stopped to idle. (MMS = Resetting)

SelectedProgram holds the library index for the selected program. SelectedProgram is 0 if State is Non-selected.
ErrorStatus is identical to ProgramControl.ErrorStatus, but only read access is possible. ErrorStatus always reads FALSE for all bits

SWNo $122: ProgramID
ProgramID is used to identify the selected program. The record includes a name for the program, version number, the required version number for the interpreter program and a name for the Softwarehouse, which created the program. Compile time, compiler version and actual size for the program is also a part of ProgramID as well as SumCheck (2's complement word addition without carry) and a code type identifier. The SumCheck value is used for check sum calculations when the program is started, i.e. following a Start command (40) or a Resume command (42). CodeType must match CodeType found in ChType. The CodeType check is performed at the same time as the SumCheck.

ProgramID is a record of the following type:

```
Record
    ProgramName     : STRING[20];    (* Offset = 0 *)
    Version         : Word;          (* Offset = 22 *)
    InterpreterVers : Word;          (* Offset = 24 *)
    SoftwareHouse   : STRING[20];    (* Offset = 26 *)
    CompileTime     : DateTimeRec;   (* Offset = 48 *)
    CompilerVersion : Word;          (* Offset = 56 *)
    ActualSize      : LongInteger;   (* Offset = 58 *)
    SumCheck        : Word;          (* Offset = 62 *)
    CodeType        : Word;          (* Offset = 64 *)
    NoOfTask        : Word;          (* Offset = 66 *)
End
```

CodeType is 2, corresponding to calculator program code.
NoOfTask is always 1.

SWNo $125: SystemPointer
This variable holds a pointer to system specific data. These system data may be used for debugging, read out of kernel data etc. Only Proces-Data know of the information stored in the SystemPointer.

SWNo $127: MemoryInfo
For the program segment stored in the library, selected by LibraryControl.LibraryIndex, a corresponding memory information can be read. The memory information holds the actual size for the program, the max. size for the program segment and a code for the memory type in which the program is stored.

MemoryInfo is a record of the following type:

```
MemoryInfo : Record
    ActualSize      : LongInteger;    (* Offset = 0 *)
    MaxSize         : LongInteger;    (* Offset = 4 *)
    MemoryType      : Word;           (* Offset =8 *)
End
```

ActualSize includes the size of the program code and the header with the ProgramID.

MaxSize indicates the max size for the program segment and is the max size for a program to download within the memory area specified by **MemoryType**.
MaxSize = 7000 and **MemoryType = 4** in PD 3120.

SWNo $128: IDAndCode
This SoftWire number is used for up- or download of programmes from/to the Program Channel. When a program is downloaded to the Program Channel, the entire program and a header with the ProgramID and size indicator, is stored in IDAndCode by means of a LongStore instruction.

The program and the header with the ProgramID, a size indicator and a code-type indicator is a variable of the following type:

```
IDAndCode = ARRAY[1..ActualSize] OF BYTE;
```

The format for the data stored in the first part of IDAndCode matches exactly the format for ProgramID, and IDAndCode is interpreted as a record of the following type:

```
Record
    ProgramName     : STRING[20];     (* Offset = 0 *)
    Version         : Word;           (* Offset = 22 *)
    InterpreterVers : Word;           (* Offset = 24 *)
    SoftwareHouse   : STRING[20];     (* Offset = 26 *)
    CompileTime     : DateTimeRec;    (* Offset = 48 *)
    CompilerVersion : Word;           (* Offset = 56 *)
    ActualSize      : LongInteger;    (* Offset = 58 *)
    SumCheck        : Word;           (* Offset = 62 *)
    CodeType        : Word;           (* Offset = 64 *)
    NoOfTask        : Word;           (* Offset = 66 *)
    Reserved1       : LongInteger;    (* Offset = 68 *)
    Reserved2       : LongInteger;    (* Offset = 72 *)
    Reserved3       : LongInteger;    (* Offset = 76 *)
    ProgramCode     : ARRAY[1..(ActualSize-HeaderSize)] OF BYTE;
                                      (* Offset = 80 *)
End
```

Before the program can be downloaded to the channel, the download program must ensure that the necessary memory area is available, the code-type for the program code is in accordance with the code-type specified for the channel and that the interpreter program in the operating system is of the right version.

To download or upload a program, the corresponding command must be sent to the LibraryControl.Command register to initiate the sequence. The download program must wait for LibraryStatus.State = Loading before the download is executed.

ActualSize for a program is given in LibraryProgramID.ActualSize after a complete download.

SWNo $129: ChConfig

The specification for how the selected program must operate after a reset or power failure is held in ChConfig. This configuration includes a number specifying which program must be invoked after reset.

ChConfig is a record of the following type:

```
Record
    Enablebit   : ARRAY[0..7] OF BOOLEAN;    (* Offset = 0 *)
    Functions   : BYTE;                      (* Offset = 1 *)
    Ref_A       : BYTE;                      (* Offset = 2 *)
    Ref_B       : BYTE;                      (* Offset = 3 *)
End
```

Enablebit[0] indicates how the selected program must operate after a power failure or module reset.

Enablebit[0] = TRUE indicates that the selected program must perform an autostart, resulting in ProgramStatus.State = Running.

Enablebit[0] = FALSE indicates that the selected program must go to ProgramStatus.State = Idle.

The Program Invocation Management on the Program Channel sends Commands to the selected program after the module reset to change State to the state specified in Enablebit[0] (Idle or Running). If an error occurs during the autostart procedure, the error code may be read in ChError.His.

Functions is not used.

Ref_A holds the selected program number which must be invoked after module reset or power up. No program is selected if Ref_A = 0. Following a successful autostart, Ref_A is copied to ProgramStatus.SelectedProgram.

Ref_B is not used.

SWNo $12A: LibraryControl

LibraryControl is used to set and change state for a program in the library. The program in the library is choosen with LibraryIndex. Commands can be sent to LibraryControl to control a download or upload sequence.

LibraryControl is a record of the following type:

```
Record
    Command       : BYTE;    (* Offset = 0 *)
    LibraryIndex  : Word;    (* Offset = 2 *)
End
```

Command is used to send a command to the Program Channel for changing the state of the program choosen by LibraryIndex. A list of possible commands is given below. The commands and the corresponding numbers conform to the Request Instructions for download used by (MMS).

Command		Purpose
26	InitiateDownloadSequence	Prepare for download. (MMS = InitiateDownloadSequence)
28	TerminateDownloadSequence	Cancel download and end sequence. (MMS = TerminateDownloadSequence)
29	InitiateUploadSequence	Prepare for upload (MMS = InitiateUploadSequence)
31	TerminateUploadSequence	Cancel upload and end sequence. (MMS = TerminateUploadSequence)
36	DeleteProgram	Delete program from the library. (MMS = DeleteProgram)

Command is automatically set to 0 by the Program Channel after writing to the variable. State is immediately updated to one of the corresponding temporary states, Complete or Incomplete each time a command is sent to the Command variable. By the change in state it is possible to read the variable LibraryStatus.State to check if the Command was executed successfully or the operation failed.

LibraryIndex chooses one of the programmes in the program library. When a program is choosen, all data concerning this program may be accessed. The data for the choosen program may be read at the variables LibraryProgramID, LibraryStatus and MemoryInfo. Upload and download of the complete program, including the program code, is done via the IDAndCode variable.

If LibraryIndex is equal to ProgramStatus.SelectedProgram, download is not possible.

The value of LibraryIndex must be in the range from 1 to MaxLibraryIndex. MaxLibraryIndex is found in LibraryStatus. MaxLibraryIndex is 1 in the PD 3120 module.

SWNo $12B: LibraryStatus

LibraryStatus is used to read the current state for a program in the library. The choosen program in the library is indicated by LibraryIndex. The max number of programmes in the library is stated by MaxLibraryIndex.

LibraryStatus is a record of the following type:

```
Record
    State           : BYTE;   (* Offset = 0 *)
    LibraryIndex    : Word;   (* Offset = 2 *)
    MaxLibraryIndex : Word;   (* Offset = 4 *)
End
```

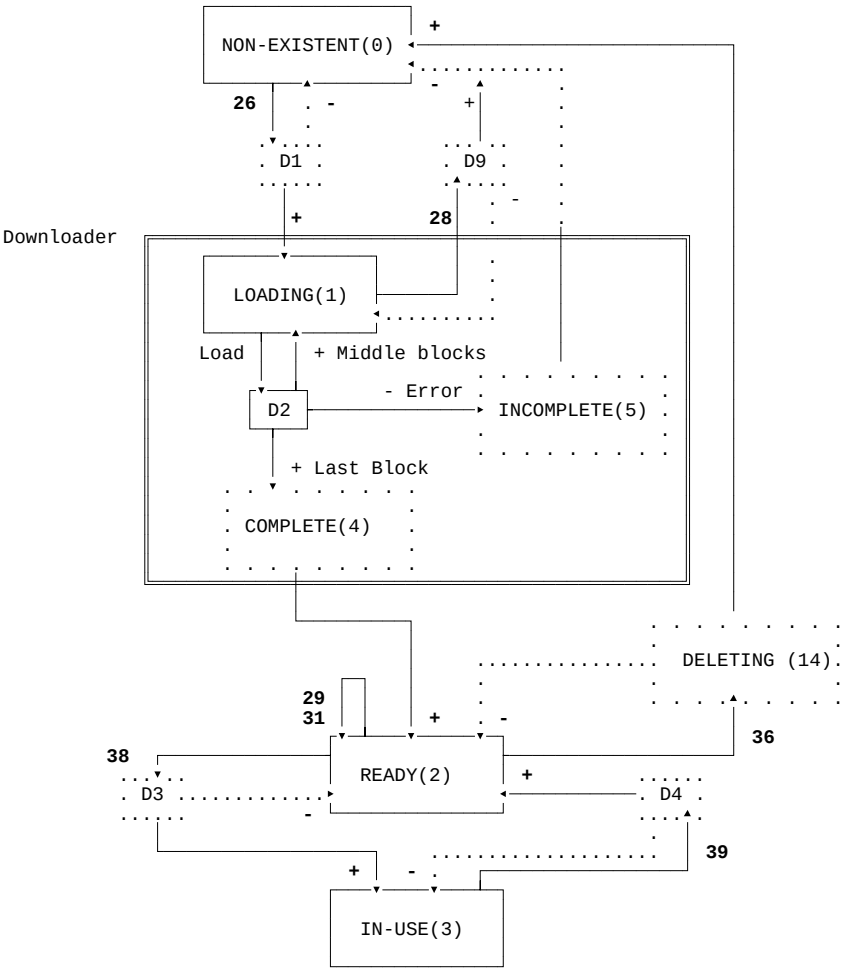
State indicates the current state for the program, eg. loading, ready, nonexistent etc. A list of possible states is given below. The states and the corresponding numbers conform exactly to the states for download domain used by The Manufacturing Message Specification.

State		**Explanation**
0	Non-existent	No program in this memory type or program segment. (MMS = Non-existent)
1	Loading	Download in progress. (MMS = Loading)
2	Ready	The program is ready to be selected to run. (MMS = Ready)
3	In-use	This program is selected in ProgramControl. The state change to/from In-use is entirely controlled in ProgramControl. Download is not allowed. (MMS = In-use)
4	Complete	The program is completely downloaded and will change state to ready. (MMS = Complete)
5	Incomplete	An error has occurred during download and the program changes state to Non-existent. (MMS = Incomplete)
14	Deleting	Deletion in progress, eg. clearing Flash. (MMS = Deleting)

LibraryIndex is identical to LibraryControl.LibraryIndex.

MaxLibraryIndex indicates the max. number of programmes which can be or are stored on the program channel. 1 type of memory and only 1 program is implemented in the PD 3120 module. MaxLibraryIndex therefore always reads 1.

PROGRAM STATE DIAGRAM



\+ State transition succeeded - State transition failed.

SWNo $12C: LibraryProgramID

LibraryProgramID corresponds completely to ProgramID, but is used to identify the program in the program library. LibraryProgramID is a record of the same type as ProgramID, which is found at SWNo $122.

A program from the library is choosen by a number, LibraryIndex, which is found in Library-Control.

SWNo $12D: 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, and category.

Maintenance is a Record of the following type:

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

SWNo $12E: ChType

For the Program Channel, ChType is of the following type:

```
Record
    ChannelType      : Word;                  (* Offset = 0 *)
    Exist            : Array[0..15] OF Boolean; (* Offset = 2 *)
    InterpreterVers  : Word;                  (* Offset = 4 *)
    CodeType         : Word;                  (* Offset = 6 *)
End
```

For the Program Channel, ChType has the following value:

ChannelType = 11

Exist =

15							7							0
1	1	1	1	1	1	1	1	1	0	1	0	0	1	1	1

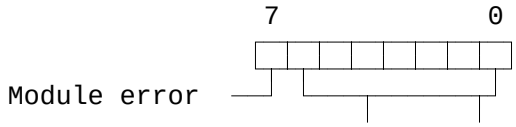
CodeType indicates which kind of program code that can be executed on the Program Channel.

InterpreterVers = 100 (first version)
CodeType = 2 (calculator program code)

SWNo $12F: 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 is a combination of an error bit and an error code and have the following meaning. When an error occurs, the corresponding bits are set in both ChError.His and ChError.Act. An UnselectProgram command (39) will clear the error code in ChError.Act.



No prg. after reset	0000001
SumCheck error	0000010
WrongInterpr.vers	0000011
No program code	0000100

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").

No prg. after reset. This error code can only occur after a reset because of one of the following conditions:
If the program number stated in ChConfig.Ref_A=0 and ChConfig.EnableBit[0]=TRUE.
If the program number stated in ChConfig.Ref_A>1 (MaxLibraryIndex).
If the program number stated in ChConfig.Ref_A=1, but the program is non-existing (Actualsize <= 0).

The error code is only set in ChError.His.

SumCheck error. When the program is started, following a Start or Resume command or an autostart after reset, a sum check is calculated and compared to the sum check in ProgramID. Any inconsistency will set this error code.

WrongInterpr vers. When the program is started, following a Start or Resume command or an autostart after reset, the interpreter version in ChType.InterpreterVers is compared to the interpreter version stated in ProgramID. Any inconsistency will set this error code.

No program code. When the program is started, following a Start or Resume command or an autostart after reset, Actualsize for the program is compared to the headersize. If (Actualsize<=Headersize), this error code will be set.

7 Data Channel (channel $13).

A selection of various universal variables are found on the Data Channel. These variables can be used as temporary variables or for saving calculated results.

Registers on Data Channel . Channel identifier: **DataChannel**

SWNo	Identifier	Memory Type	Read Out	Type
130	RealA	RAM Auto Save	Decimal	Real
131	RealB	RAM Init EEPROM	Decimal	Real
132	IntegerA	RAM Auto Save	Decimal	Integer
133	WordA	RAM Auto Save	Decimal	Word
134	BooleanA	RAM Auto Save	Binary	Bit8
135	BooleanB	RAM Init EEPROM	Binary	Bit8
136	RealArray	RAM Read Write	Decimal	Array8Real
137	IntegerArray	RAM Read Write	Decimal	Array8Integer
138	BooleanArray	RAM Read Write	Binary	Bit32
139	ChConfig	EEPROM RPW	- - - - - - -	Record
13A	TimerA	RAM Read Write	Decimal	Real
13B	TimerB	RAM Read Write	Decimal	Real
13C	LookUp1	EEPROM RPW	Decimal	Record
13D	LookUp2	EEPROM RPW	Decimal	Record
13E	ChType	PROM Read Only	- - - - - - -	Record
13F	ChError	RAM Read Write	- - - - - - -	Record

SWNo $130 - $133:
These variables are universal variables of simple type. The variables may be used by the calculator program for storing temporary values or calculated results. Some of these variables are saved in EEPROM at a certain predetermined frequency.

SWN0 134 - 138:
These variables are universal variables of array type and may be accessed by using the index registers in the calculator.

SWN0 139: ChConfig
This variable is the same type as the standard channel configuration variable. If you design an application specific channel, eg. special regulator with ramp up/down facilities, it may be used as to select between different configurations.

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
```

SWNo $13A and $13B: TimerA and TimerB

These registers holds timer variables, which can be used by the calculator program. The timers counts down with a resolution of 1/32 second. The count continues through negative values. The timer registers are cleared after a power failure or reset. The maximum value for the timer is approximately 97 days. After a overflow, the timer continues from the maximum value.

SWNo $13C and $13D: LookUp1 and LookUp2

These variable is declared as lookup tables with the following format:

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

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

Each variable represents a line through 10 pairs of x,y coordinates. 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 X10, it returns Y10.

SWNo $13E: ChType

For the data channel, ChType is of the following type:

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

The Data Channel is a company specific channel where ChType has the following value:

ChannelType = $8001

Exist =

15								7							0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

SWNo $13F: 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 may be useful in indicating erroneous data. The register is declared as follows:

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

Any error bit set in ChError will result in the corresponding channel-bit set in ComHis / ComAct in CommonError at the Service channel.

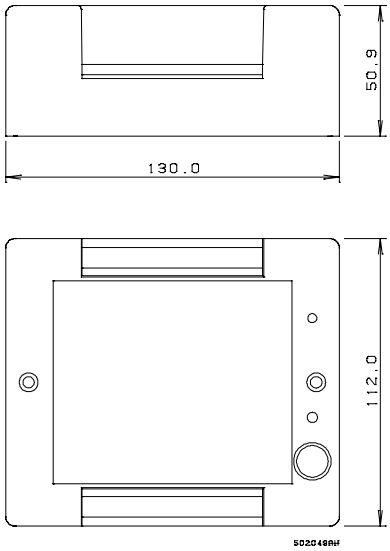
8 Construction, Mechanical.

The PD 3120 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, or 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.

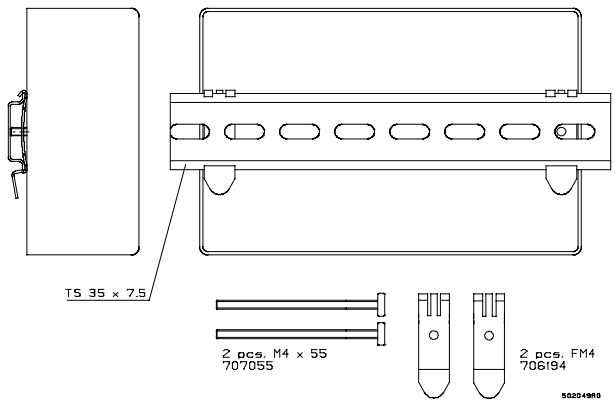
Scale drawing (in mm):



Materials

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

Rail mounting:



9 Specifications.

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

All specifications are respected in the approved EMI conditions. EMC test specifications for PD 3120 are available in a separate document, PD no. **506 021**.

9.1 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 10 A time delay.

9.2 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

9.3 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 :		31.25 ms
FeedBack update time:		125 ms

Digital Output.

Load current measurement :	
Accuracy :	± 19 mA
Resolution :	12.5 mA
Repeatability :	± 12.5 mA
Current measurement update time :	31.25 ms

9.4 Calculator program.

Memory size:	7000 bytes
Instruction time:	typical 0.3 ms

9.5 Ambient Temperature.

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

9.6 Humidity.

Relative humidity :	max.95 %

9.7 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

10 Survey of variables in the PD 3120 module.

SWNo	**Service** 0	**Digital_IO_x** 1 - $10	**Common_IO** $11	**Calculator** $12	**DataChannel** $13
x0	NumberOfSWNo	FlagReg	OutFlags	ProgramControl	RealA
x1	DeviceID	OutTimer	InFlags	ProgramStatus	RealB
x2		Counter	IOChError	ProgramID	IntegerA
x3	Reset	OutCurrent			WordA
x4	PnetSerialNo	OperatingTime			BooleanA
x5		UserByteArray		SystemPointer	BooleanB
x6		FBTimer			RealArray
x7	FreeRunTimer	FB Preset		MemoryInfo	IntegerArray
x8	WDTimer	OutPreset		IDAndCode	BooleanArray
x9	ModuleConfig	ChConfig		ChConfig	ChConfig
xA	WDPreset	MinCurrent		LibraryControl	TimerA
xB		MaxCurrent		LibraryStatus	TimerB
xC		UserRealArray		LibraryProgramID	LookUp1
xD	WriteEnable	Maintenance		Maintenance	LookUp2
xE	ChType	ChType	ChType	ChType	ChType
xF	CommonError	ChError	ChError	ChError	ChError

Contents