502 063 02

WEIGHT TRANSMITTER

PD 3230 / PD 3235

Manual

GB *August 1996*

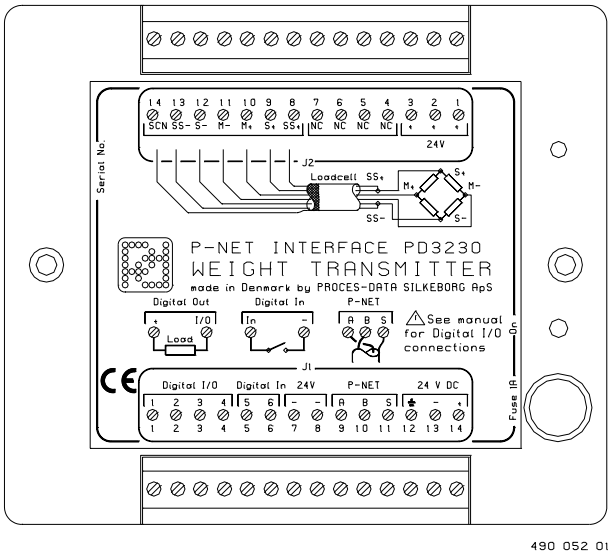
PROCES-DATA A/S, Navervej 8, 8600 Silkeborg, Denmark, phone +45-87 200 300, fax + 45-87 200 301

Contents

1 General information.

The PD 3230 and PD 3235 Weight Transmitters are a members of PROCES-DATA's module series 3000. In this manual, illustrations are shown for PD 3230 but descriptions and specifications applies to both weight transmitter types. Where differences appear in specifications it is clearly stated for each module independently.

The Weight Transmitter provides a versatile interface between weighing elements, the associated digital elements such as valves, switches, motors etc. and the P-NET fieldbus.



490 052 01

It is an intelligent module, provided with 1 Weight Channel, 6 Digital Input Channels of which 4 can be configured as Digital Outputs and an internal user programmable Calculator Channel for local control.

The P-NET is a field-bus network designed for process control and data collection.

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 module can be controlled via the P-NET, or it can operate as an autonomous unit.

The unit provides high resolution internal conversion of load measurements into engineering units with single and double taring of the measured value. The weight change per time (Flow) is continuously calculated. The module can be configured to perform Belt Weight functions with either an active (belt running) signal or a pulse frequency proportional to the belt velocity. The continuous fully integrating non sampling principle offers high noise immunity. The influence of vibrations can be damped with the adjustable averaging function.

The module possesses a programmable calculator channel, which can be used to control the digital outputs. With the utilization of user programmes application specific autonomous functions such as dosing etc. can be added.

The compact design and the outstanding environmental specifications for the Weight Transmitter makes it an ideal process component in industrial as well as other environments.

1.1 Features.

- 1 Weight Channel with Weight and Belt Weight functions
- High Resolution non sampling, fully integrating principle
- High Accuracy factory calibration
- 6 Digital channels: 2 I and 4 I/O
- Pulse or Contact Counting
- Pulse and one-shot on all outputs
- Output Feedback Facility
- Automatic output Functions
- Current measuring and Overload Protection on each output
- 1 Programmable Calculator 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/EEC)

1.2 System description.

Various configurable automatic functions can be selected, such as automatic calculation of fullscale and zeropoint during calibration, automatic calculation of tare and presettable limit switches. These functions combined with the user programmable calculator reduce the basic operations in the central control system and enable the unit to operate autonomously in a stand-alone system. The unit offers comprehensive self-testing features, which enables reporting of disconnection, overload and process failure.

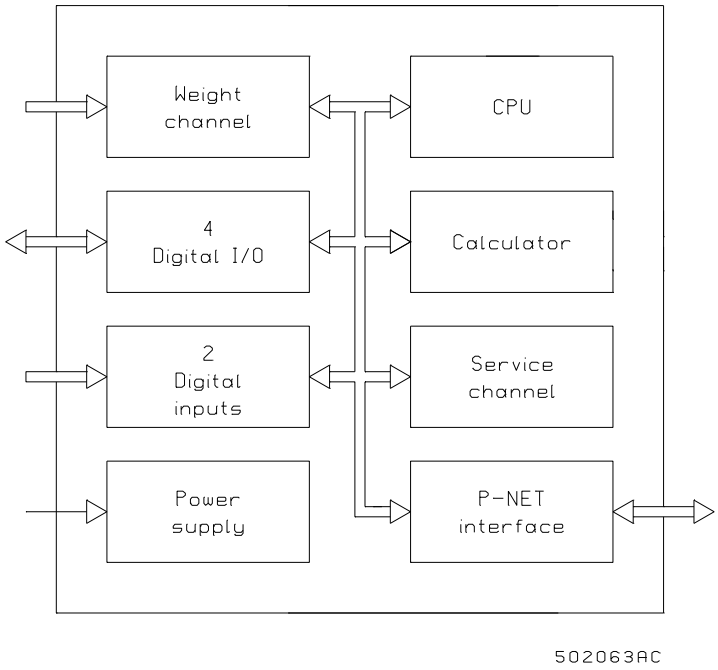
All digital outputs are protected against overload. The selectable watchdog timer ensures the safe shut down of a process during a communications or power failure. The output current (Sink current) is measured continuously on each digital output channel and may be read as a value in Amps. If the current exceeds the specified max value, the output is switched off and an errorcode (overload) is generated in the module. 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.

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

A pattern approved weighing system according to the European Standard **EN 45501** for non-automatic weighing systems can be designed with PD 3230 Weight Transmitters and PD 4000 Weight Displays. Refer to PD Manual 502 073 01 for further information.

1.3 Channels/registers.

The Weight Transmitter module contains:



1 Service channel	(channel 0)	1 Common I/O channel	(channel 7)
4 Digital I/O's	(channel 1-4)	1 Weight input channel	(channel 8)
2 Digital inputs	(channel 5-6)	1 Calculator channel	(channel 9)

A set of 16 variables numbered from 0 - $F, are 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 ®.

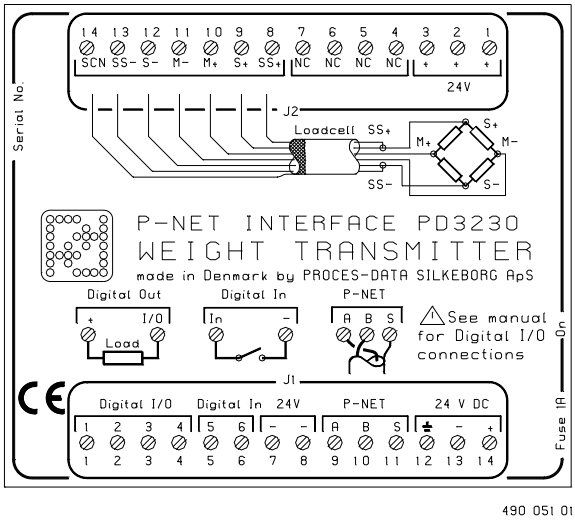
1.4 Connections.

The Weight Transmitter 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 wiring should be designed with a maximum of 2 wire connections in each screw terminal.

The module is protected by a replaceable built in fuse.

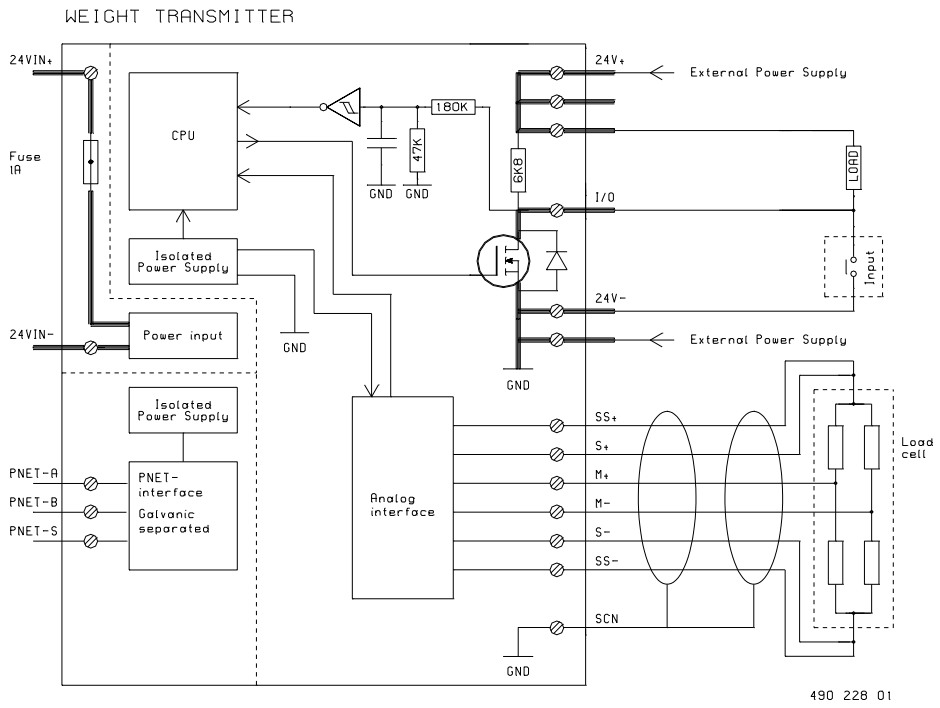
When using the digital inputs or outputs the, 24V+ and 24V- terminals can be used for connection of external equipment and a power supply. To avoid ground loops, an external isolated power supply is recommended. If the connected load cell is isolated from ground, the 24VIN+ and 24VIN- terminals can be used to supply the 24V+ and 24V- terminals.

Connection identities are printed on top of the module.



490 051 01

Hardware diagram, principle.



490 228 01

1.5 Memory types.

The Weight Transmitter 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. AutoSave is suspended if there is an unacknowledged EEPROM error.

2 Service channel (channel 0).

The Weight Transmitter 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	Decimal	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 = 3230
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

Weight Transmitter 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.

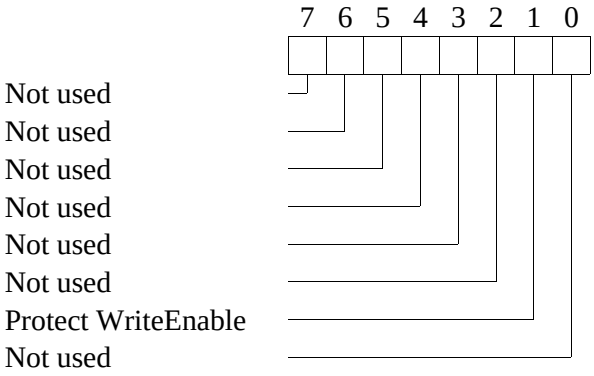
SWNo. 9: ModuleConfig

The variable is a record of the following type:

```
Record
    Enablebit: Array[0..7] Of Boolean;   (* 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 WriteEnable variable may optionally be write protected by means of a hardware locking. The hardware locking is enabled by setting ModuleConfig.Enablebit[1] to TRUE and setting a TRUE signal on digital input 5. This results in a complete write protection of ALL EEPROM variables in the Weight Transmitter. When the signal at input 5 is FALSE, normal writing is possible to the WriteEnable variable.

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 service channel.

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 ("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 8 seconds. 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: Array[0..15] Of Boolean;     (* Offset = 2 *)
    Functions: Array[0..15] Of Boolean; (* Offset = 4 *)
End
```

For the service channel, ChType has the following value:

ChannelType = 1

Exist =

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	1	1	1	1	0	0	1	1	0	1	1

Functions =

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

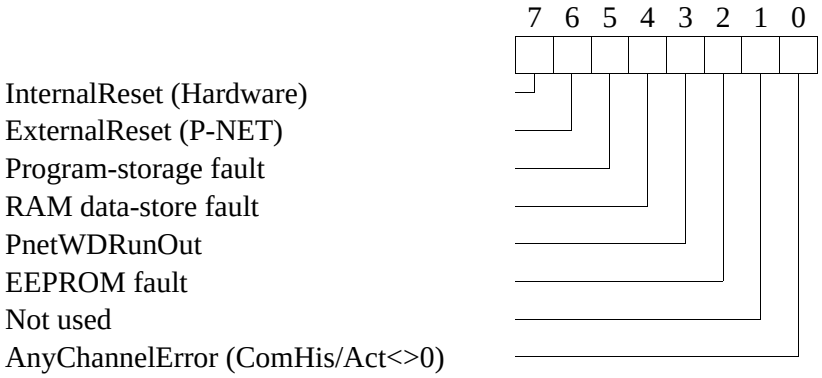
Not used
Not used
Not used
Not used
Not used
Not used
Not used
Not used
Not used
Not used
Not used
Not used
Not used
Not used
No Watchdog
Watchdog

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 (the error will disappear after approx. 16 sec.).

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

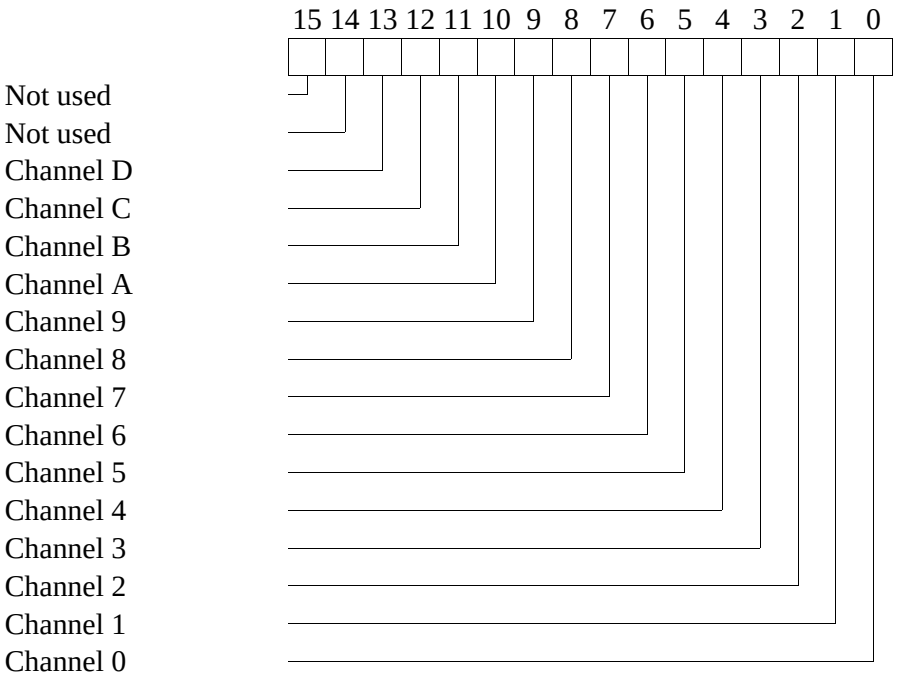
The following function of ChError.His and ChError.Act is analogous in all Channels:

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.



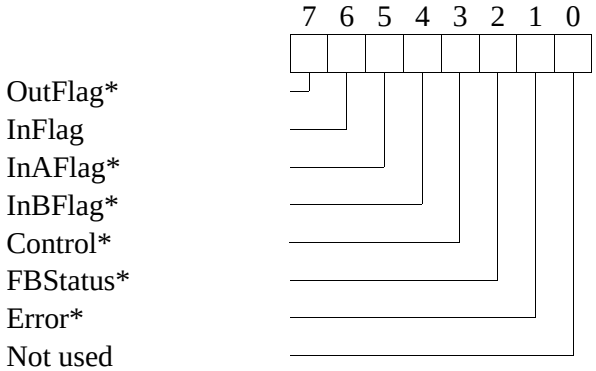
3 Digital I/O channel (channel 1 - 6).

Variables on 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						
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						
xD	Maintenance	*	EEPROM RPW	- - - - - -	Record	- - -
xE	ChType		PROM Read Only	- - - - - -	Record	- - -
xF	ChError		RAM Read Only	Binary	Record	- - -

* Not available for Ch5 and Ch6.

SWNo x0: FlagReg



* Not available for Ch5 and Ch6

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 1/8 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.

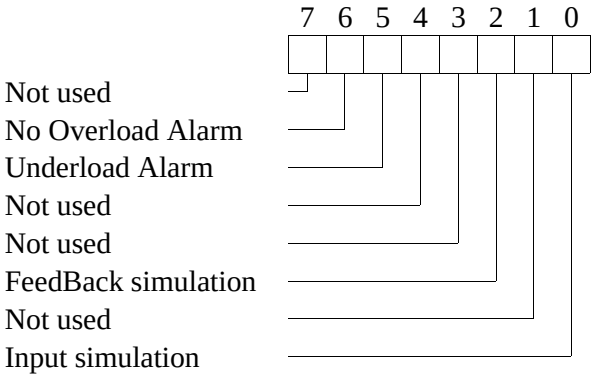
Feedback control: The correct feedback state is Input A = Output and Input B = NOT Output. The feedback inputs can be disabled by writing a 0 as the channel number, in ChConfig.Ref_A and/or ChConfig.Ref_B fields, if only one or no input channels are required.

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 = $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 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 and category.

The 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 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:

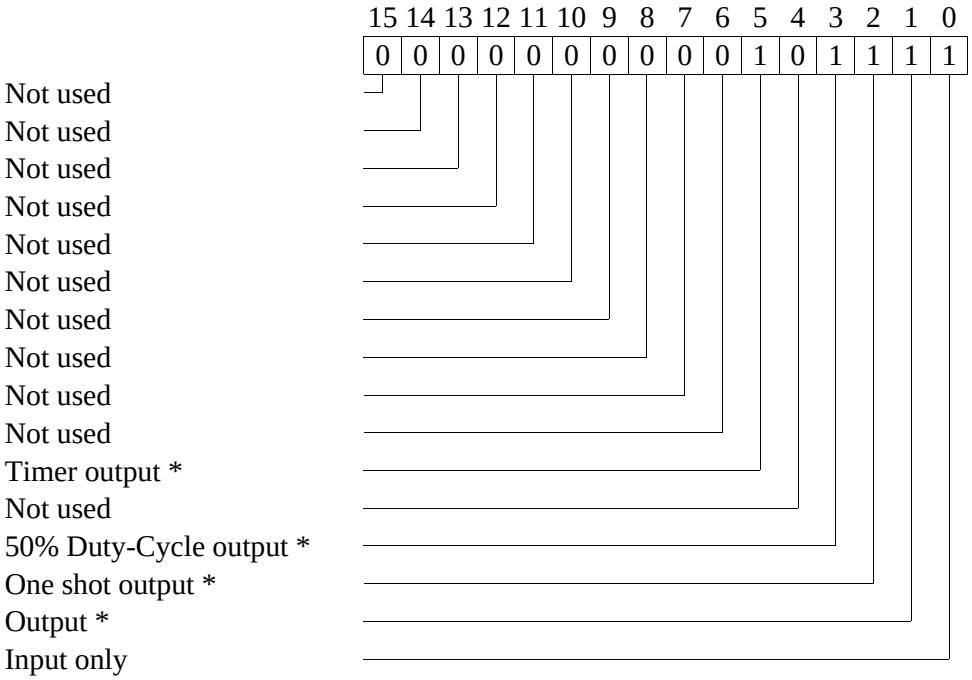
ChannelType = 2

Exist =

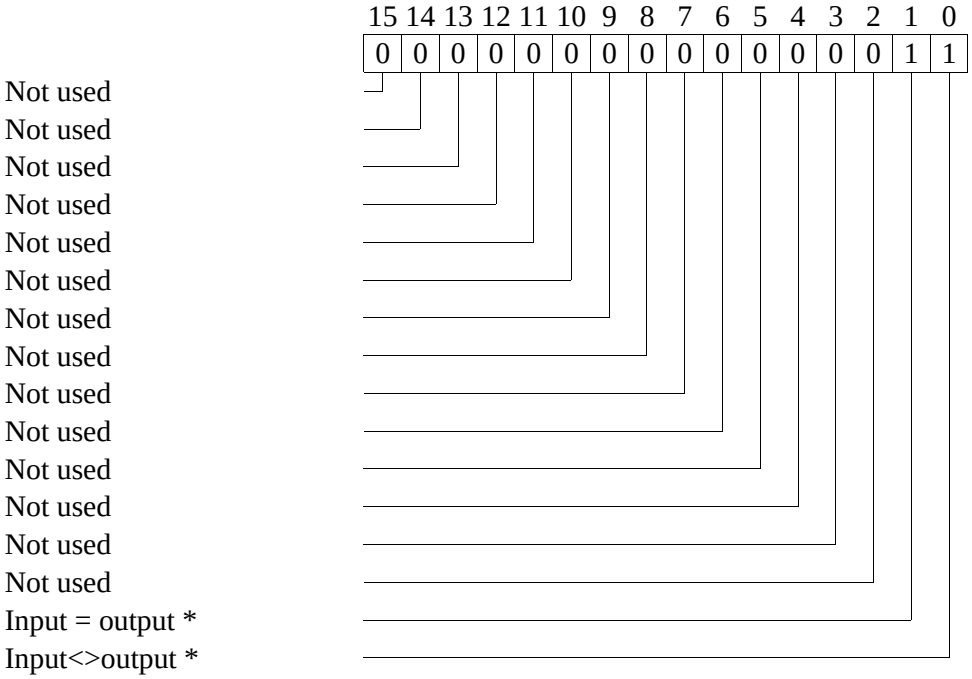
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1
		*		*	*		*	*	*			*		*	

* Not available for Ch5 and Ch6.

Functions =



Feedback =

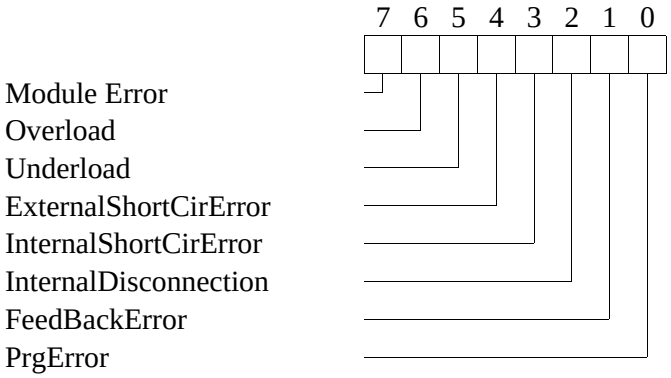


* These bits are not set for Ch5 and Ch6.

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. 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) or if the channel is configured as an output and no power supply is connected. 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.

3.1 Connections to DI/DO.

If the digital inputs or outputs are to be used a power supply must be connected to the 24V+ and 24V- terminals.

When the connected load cell is isolated from ground, the 24VIN+ and 24VIN- terminals can be connected. Otherwise an external isolated power supply must be used to avoid ground loops (see also Hardware diagram, Principle, on page 4).

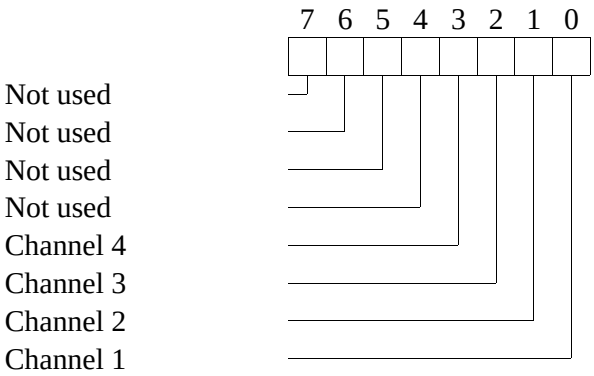
If a digital channel is configured as an output and no power supply is connected, External-ShortCirError is set (refer to the Hardware diagram, Principle, on page 4).

4 Common I/O channel (channel 7).

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

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

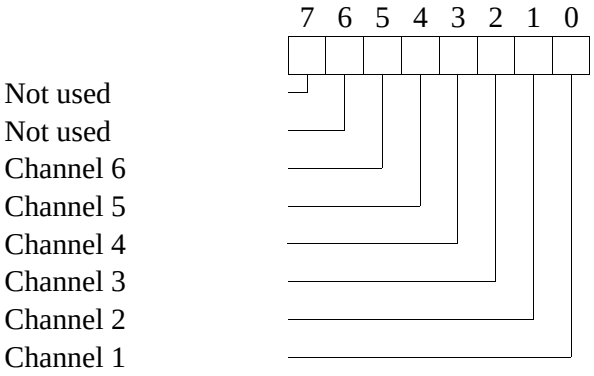
SWNo 70: OutFlags



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.

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 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: Array[0..15] Of Boolean;         (* Offset = 2 *)
    ExistingChannels: Array[0..7] Of Boolean;   (* Offset = 4 *)
End
```

ChType has the following value:

ChannelType = 3

Exist =

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1

ExistingChannels =

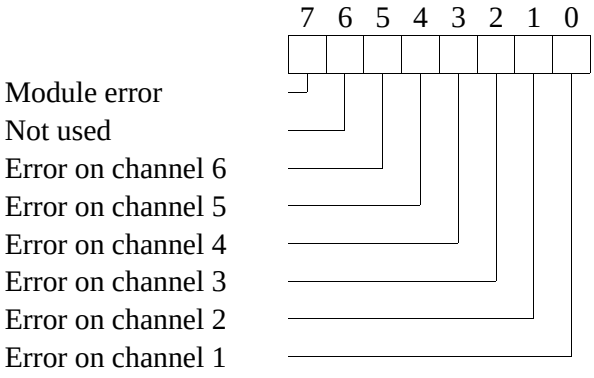
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

SWNo 7F: IOChError

IOChError has the following type:

```
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.

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").

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 Weight channel (channel 8).

The weight channel has an input for weight measurement and possesses high resolution conversion of load measurements into engineering units with single and double taring of the measured value. The flow (weight change per time) is continuously calculated. A beltweight providing either an active signal (belt running) or a pulse frequency proportional to the belt velocity can be connected directly to the module.

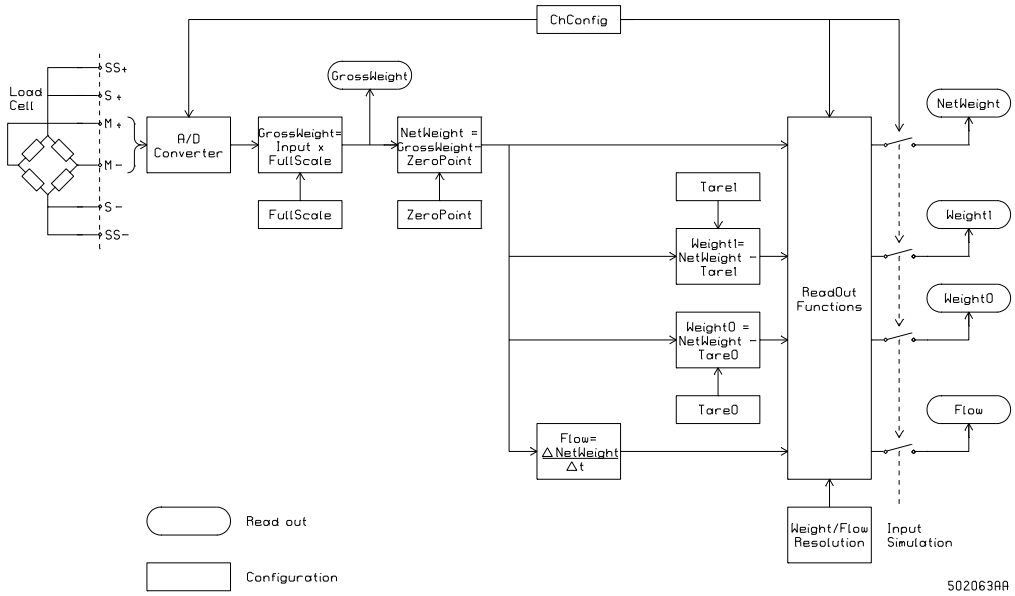
The module uses a non sampling, full time averaging principle. This principle always suppress 50 and 60 Hz line interference, and is user adjustable to damp the influence from vibrations and filter the measured signal.

The input uses a six point measurement technique to eliminate cable losses. The PD 3230 module can supply paralleled load cells down to a load resistance of 60 Ohms and the PD 3235 module down to a load resistance of 30 Ohms. The input sensitivity is 2 mV/V.

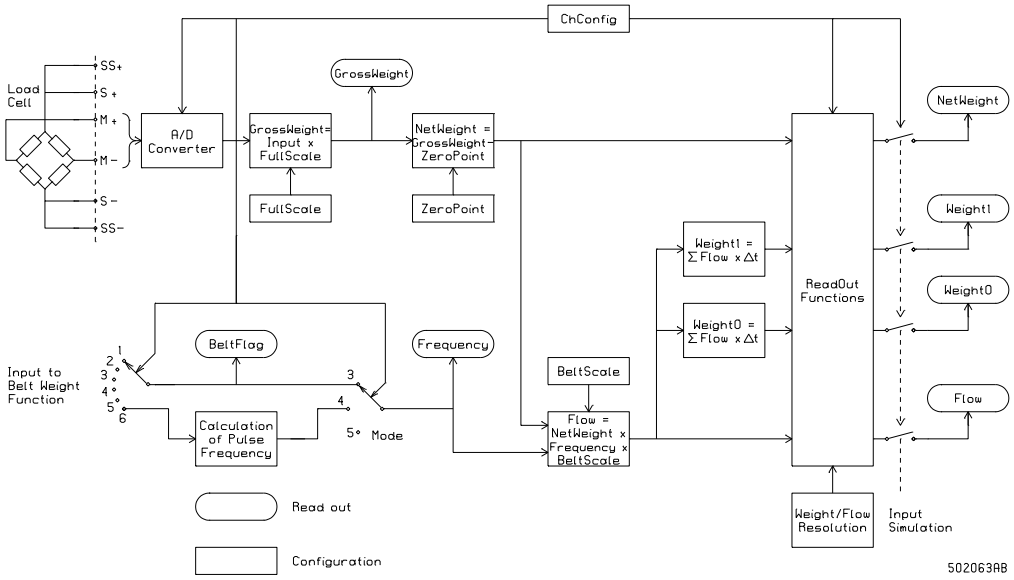
Averaging and/or rounding (readout resolution) can be chosen for the readout of each of the main variables Weight0, Weight1, NetWeight and Flow.

Various automatic functions can be selected, such as automatic calculation of fullscale and zeropoint under adjustment, automatic calculation of tare and presettable limit switches.

Block diagram of weight function:



Block diagram of belt weight function:



Variables on Weight channel (channel 8). Channel identifier: **Weight**

SWNo	Identifier	Memory type	Read out	Type	*SI Unit
80	Weight0	RAM Read Write	Decimal	Real	kg
81	Weight1	RAM Read Write	Decimal	Real	kg
82	NetWeight	RAM Read Write	Decimal	Real	kg
83	Flow	RAM Read Write	Decimal	Real	kg/s
84	Frequency	RAM Read Write	Decimal	Real	- - -
85	FlagReg	RAM Read Write	Binary	Bit8	- - -
86	Tare	RAM Read Write	Decimal	Record	kg
87	HighLevel	RAM Init EEPROM	Decimal	Real	kg
88	LowLevel	RAM Init EEPROM	Decimal	Real	kg
89	ChConfig	EEPROM RPW	- - - - - -	Record	- - -
8A	Factors	EEPROM RPW	Decimal	Record	- - -
8B	FullScale	EEPROM RPW	Decimal	Real	kg
8C	ZeroPoint	EEPROM RPW	Decimal	Real	kg
8D	Maintenance	EEPROM RPW	- - - - - -	Record	- - -
8E	ChType	PROM Read Only	- - - - - -	Record	- - -
8F	ChError	RAM Read Only	Binary	Record	- - -

* Any weight unit can be used instead of kg. The unit is chosen by the value inserted in the **FullScale** variable.

SWNo 80: Weight0.
When the Weight Channel is in weight mode, this variable holds the Zeropoint adjusted and tared weight.

Calculation (Weight mode):
Weight0 = NetWeight - Tare0

If a value is written in the variable, it is preset at that value and the resulting tare value is calculated and placed in the Tare0 variable (see SWNo 86 Tare).

When the channel is in belt weight mode, the variable holds the summed flow.

Calculation (Belt weight mode):
Weight0:= Weight0 + Flow * SampleTime

If a value is written in the variable, it is preset at that value.

SWNo 81: Weight1.
This is an independent variable with features similar to Weight0.

Calculation (Weight mode):
Weight1 = NetWeight - Tare1

Calculation (Belt weight mode):
Weight1:= Weight1 + Flow * SampleTime

SWNo 82: NetWeight.
NetWeight holds the zeropoint and fullscale adjusted weight.

Calculation: NetWeight = Input * FullScale - ZeroPoint

ZeroPoint will be calculated automatically if 0 is written in NetWeight, and the Service channel WriteEnable bit is set.

FullScale will be calculated automatically if the known weight of a calibrated load at the load cell is written in NetWeight, and the Service channel WriteEnable bit is set. To ensure accuracy, the calibration load must be a reasonable fraction of the maximum load.

Since the automatic calculation of FullScale and ZeroPoint is based on the averaged weight input, the functions should not be repeated with short time intervals. The averaging function needs at least SampleTime * NoOfSamples [s] (see description of ChConfig) to stabilize with a new FullScale or ZeroPoint value.

SWNo 83: Flow.
This variable holds the actual flow (weight change per second) in kg/s.

Calculation (Weight mode):
Flow = (NetWeight - previous NetWeight) / SampleTime

Calculation (Belt weight mode):
Flow = NetWeight * Frequency * BeltScale

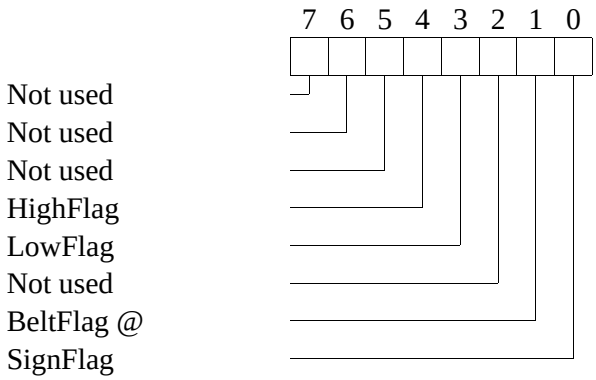
SWNo 84: Frequency.
Frequency is only used when the channel is in Belt Weight modes.

In Belt Weight Active mode, Frequency indicates the input status for the channel selected in BeltRef (see also SWNo 89 ChConfig). Frequency acts like a boolean, indicating with "0" that the belt is stopped and "1" that the belt is running.

In Belt Weight Pulse mode, Frequency holds the input pulse frequency [Hz]. The terminals of the digital input channel DI6 is automatically used for Pulse frequency input.

In Belt Weight Frequency mode, the input frequency can be written directly in Frequency.

SWNo 85: FlagReg.



When SignFlag is set Weight0, Weight1 and Flow is multiplied by -1. This can be useful when dosing to/from a vessel.

HighFlag and LowFlag is set by the HighLevel and LowLevel (see description) limit switches for NetWeight. Operation is not affected by ChConfig.EnableBit.

BeltFlag is only used in Belt Weight Active mode (ChConfig.Functions = $3X). It is an indirect variable showing FlagReg[InFlag] at the digital input channel selected with BeltRef. If BeltRef = "0", it is possible to write directly in BeltFlag. BeltFlag = 1 indicates that the belt is running at the speed inserted in BeltScale, and BeltFlag = 0 indicates that the belt is stopped.

SWNo 86: Tare.

```
Record
    Tare0: Real          (Offset = 0);
    Tare1: Real          (Offset = 4);
    GrossWeight: Real    (Offset = 8);
End.
```

The tare variables Tare0 and Tare1 is only used when the channel is in weight mode and holds the tare for Weight0 and Weight1 respectively. The known weight of a container etc. can be inserted in the tare variables.

If zero or another value is written in the corresponding weight variable, the resulting tare value is calculated and inserted automatically.

Calculation: Tare0 = NetWeight - Weight0
Tare1 = NetWeight - Weight1

GrossWeight holds the load of the load cell without any taring or zeropoint adjustment.

Calculation: GrossWeight = Input * FullScale

SWNo 87: HighLevel.
HighLevel is a limit switch for NetWeight with the following function:

```
IF (NetWeight > HighLevel) AND ChConfig.Enablebit[4]
THEN HighAlarm := True ELSE HighAlarm := False;

IF (NetWeight > HighLevel)
THEN HighFlag := True ELSE HighFlag := False;
```

SWNo 88: LowLevel.
LowLevel is a limit switch for NetWeight with the following function:

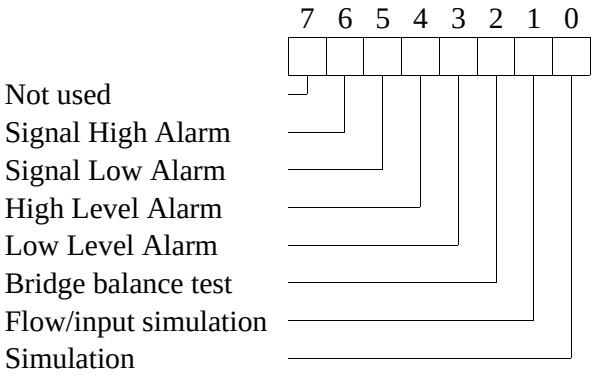
```
IF (NetWeight < LowLevel) AND ChConfig.Enablebit[3]
THEN LowAlarm := True ELSE LowAlarm := False;

IF (NetWeight < LowLevel)
THEN LowFlag := True ELSE LowFlag := False;
```

SWNo 89: ChConfig.
The channel configuration for the weight channel is stored in a record with the following type:

```
Record
    Enablebit: Array[0..7] Of Boolean;   (* Offset = 0 *)
    Functions: BYTE;                     (* Offset = 1 *)
    ReadOut: Array[0..7] Of Boolean;     (* Offset = 2 *)
    NoOfSamples: BYTE;                   (* Offset = 3 *)
    BeltRef: BYTE;                       (* Offset = 4 *)
End;
```

The fields in ChConfig have the following interpretation:

Enablebit:



It is recommended to enable Signal High Alarm and Signal Low Alarm. These alarms combined with Overload Error, External Error and Internal Error which can not be disabled, detects most signal, connection and hardware errors. High Level Alarm and Low Level Alarm are user programmable limit switches.

The bridge balance test is one of the conditions for External Error. This test checks for disconnected or defective load cells by measuring the voltage at the M+ and M- terminals. If the voltage at one of the terminals is more than 8% from midpoint of the load cell supply voltage, an External Error is indicated. It is recommended to enable this test, unless poor tolerance load cells are causing External Error alarms.

Simulation.

Simulation modes are tools for testing application programmes or simulation of an active module. Simulation is enabled by setting ChConfig.Enablebit[0] to TRUE. The type of simulation is selected with ChConfig.Enablebit[1].

Automatic calculation of Tare0, Tare1, ZeroPoint, FullScale and the readout functions Round and Average are disabled during simulation. Weight measuring and calculation of GrossWeight continues during simulation. Alarm and error functions maintain the normal function.

Flow simulation mode:
The weight variables Weight0, Weight1 and NetWeight changes linearly with time according to the value of Flow, making it possible to test dosing programmes etc. The constant flow value must be inserted in the Flow variable.
Flow simulation is enabled by setting ChConfig.EnableBit[0] to TRUE and ChConfig.Enable-Bit[1] to TRUE.

Input simulation mode:
Weight0, Weight1, NetWeight, Flow, Frequency and BeltFlag will not be updated. Values written in these variables will not be updated or overwritten by the module. This makes it possible to simulate any value in the variable for test purposes.
Input simulation is enabled by setting ChConfig.EnableBit[0] to TRUE and ChConfig.Enable-Bit[1] to FALSE.

Functions:
The functions field is used in a hexadecimal format, where the most significant digit is used to specify the channel functions and the least significant digit is used to specify the time between readouts (SampleTime).

Mode:

Functions = $00 => Channel Disabled
Functions = $1X => Weight Precision
Functions = $2X => Weight Industrial
Functions = $3X => Belt Weight Active
Functions = $4X => Belt Weight Pulse Input
Functions = $5X => Belt Weight Frequency

SampleTime:

Functions = $X7 => SampleTime = 0.1 s
Functions = $X8 => SampleTime = 0.2 s
Functions = $X9 => SampleTime = 0.5 s
Functions = $XA => SampleTime = 1 s
Functions = $XB => SampleTime = 2 s
Functions = $XC => SampleTime = 5 s

If the channel is not used, write "00" in ChConfig.Functions. Otherwise errors may occur.

The Weight channel can be configured in either Weight Mode: Weight Precision, Weight Industrial or in Belt Weight Mode: Belt Weight Active, Belt Weight Pulse Input or Belt Weight Frequency. The modes are described in the following.

Weight mode:

In Weight Mode, ordinary weight function is performed.

In most applications the Weight Industrial mode (Functions = $2X) is recommended.

In the Weight Precision mode (Functions = $1X), accuracy is improved at the cost of a slightly reduced noise immunity.

Belt Weight mode:

In Belt Weight Mode, an additional input signal is provided. If a constant belt speed is used, this signal can be an active input signal from one of the Digital channels, indicating with TRUE that the belt is running with the speed/m corresponding to the value inserted in Frequency, and indicating with FALSE that the belt is stopped. If a variable speed belt is used, the signal must be a pulse input, where the pulse frequency indicates the belt speed. The scale factor BeltScale, which gives information about the weighing system, must be calculated and inserted.

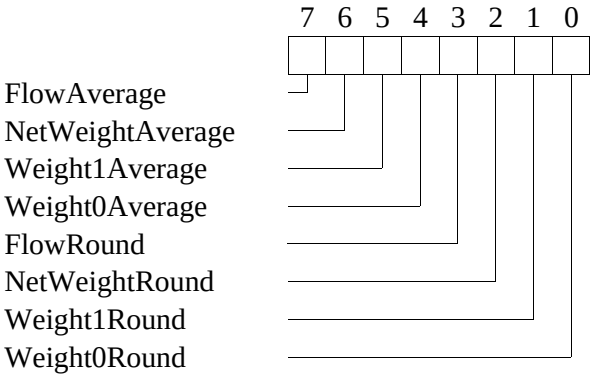
In the Belt Weight Active mode (Functions = $3X), the digital input channel for the active signal is selected by setting BeltRef equal to the channel no. If BeltRef = 0, the state for the Active input must be written directly in the FlagReg[BeltFlag] variable (SWNo. 85). The Frequency variable holds the status at the digital input.

In the Belt Weight Pulse mode (Functions = $4X), the digital input on channel no. 6 is automatically used for Pulse frequency input. The Frequency variable holds the actual input frequency.

In the Belt Weight Frequency mode (Functions = $5X), the input frequency must be written directly in the Frequency variable.

SampleTime is the time between readouts. A sample represents the averaged measured input since last readout. The module uses a non sampling, full time averaging principle to obtain high resolution and high suppression of mechanical interference in the weight system.

ReadOut:



ReadOut bits enables averaging and/or rounding on the main variables. Averaging or rounding of a variable does not affect the calculation of other derived variables.

If averaging is desired on one or more variables, the number of samples to be averaged must be inserted in ChConfig.NoOfSamples.

If rounding of a weight variable is desired, the resolution in kg must be inserted in the variable Factors.WeightResolution. For the Flow variable the resolution in kg/s must be inserted in Factors.FlowResolution (see also SWNo. 8A).

NoOfSamples:
The number of samples to be averaged must be inserted in NoOfSamples. The value is used for averaged read out of variables and for stable automatic calculation of FullScale and ZeroPoint. The range for NoOfSamples is 1..32.

The total average time is calculated as SampleTime * NoOfSamples. To damp the influence from vibrations/periodic noise, the total average time should be a multiplum of the vibration/noise period. Note that 50 and 60 Hz is always suppressed by any value of SampleTime. The best setting is a compromise between speed and noise and can be found by calculation and/or experiments.

BeltRef:
In the Belt Weight Active mode (Functions = $3X), the Input Channel No for the active signal must be inserted in BeltRef. If BeltRef = "0" the active signal must be written directly in the BeltFlag variable.

SWNo 8A: Factors.
The Factors variable holds different values for selecting flow- and weight resolution, and a scale factor for the belt speed.

Factors is a record of the following type:

Record

```
        WeightResolution: Real;      (* Offset = 0 *)
        FlowResolution: Real;        (* Offset = 4 *)
        BeltScale: Real;             (* Offset = 8 *)
    End
```

WeightResolution:
This variable offers the facility of rounding the variables Weight0, Weight1 and NetWeight to a valid resolution. If rounding of one or more of the variables is desired, the resolution in kg must be inserted in WeightResolution. The rounding of the separate variables, are enabled by setting the corresponding bits in ChConfig.ReadOut to TRUE.

FlowResolution:
If rounding of the Flow variable is desired, the resolution in kg/s must be inserted in Flow-Resolution. The rounding of Flow is enabled by setting ChConfig.ReadOut[3] to TRUE.

Belt weight mode:
The FlowResolution variable in the Factors register is also used as a minimum value for Flow. This means that if the absolute value for the calculated Flow is less than FlowResolution, resulting in the readout value for Flow to be zero, Flow will also be set to zero and consequently Weight will not be summarized.
If rounding of Flow is enabled, this zero suppression is also enabled for Flow.
If a low limit for flow is not used when summarizing weight, this may lead to miscalculated weight results due to an empty running belt with a small zeropoint error.

BeltScale:
If the belt weight functions are used, the value of BeltScale must be calculated and inserted by the user. BeltScale is calculated from the following information about the weighing system:

X: Active length of belt over load cell [m].

Y: Number of pulses per meter [pulses/m]. (Belt Weight Pulse or Frequency modes)

Z: Belt velocity [m/s]. (Belt Weight Active mode)

The calculation of BeltScale depends on the selected mode:

Belt Weight Pulse or Frequency modes:
BeltScale = 1 / (X * Y) [ ]

Belt Weight Active mode:
BeltScale = Z / X [Hz]

SWNo 8B: FullScale.
This variable holds the FullScale value for GrossWeight. The value expected at maximum signal from the load cell(s) must be calculated and inserted in FullScale.

FullScale can be calculated from the following information about the weighing system:

A: Number of load cells [ ]
B: Maximum load per load cell [kg]
C: Weight transmitter sensitivity: 2 [mV/V]
D: Load cell sensitivity [mV/V]

Calculation of FullScale:
FullScale = A * B * (C / D)

Example: FullScale = 6 cells * 100kg/cell * 2mV/V / 2mV/V = 600kg

FullScale can also be calculated automatically if the known weight of a calibrated load at the load cell(s) is written in NetWeight. To ensure accuracy the calibration load must be a reasonable fraction of maximum load (Note: The load at any load cell should always be within the specified maximum safe load range).

The Service Channel WriteEnable bit must be set TRUE to change the value of FullScale.

Automatic calculation of FullScale is disabled when the channel is in Input Simulation or Flow Simulation mode.

SWNo 8C: ZeroPoint.
This variable holds the ZeroPoint value for NetWeight. The value expected at minimum signal from the load cell must be inserted, or calculated automatically by writing "0" in NetWeight. The Service Channel WriteEnable bit must be set TRUE to change ZeroPoint.

Automatic calculation of ZeroPoint is disabled when the channel is in Input simulation mode.

SWNo 8D: 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 8E: ChType.
For the Weight channel, ChType has the following type:

```
Record
    ChannelType: Word;                  (* Offset = 0 *)
    Exist: Array[0..15] Of Boolean;     (* Offset = 2 *)
    Functions: Array[0..15] Of Boolean; (* Offset = 4 *)
    SampleTime: Array[0..15] Of Boolean; (* Offset = 6 *)
End;
```

ChannelType = 10

Exist =

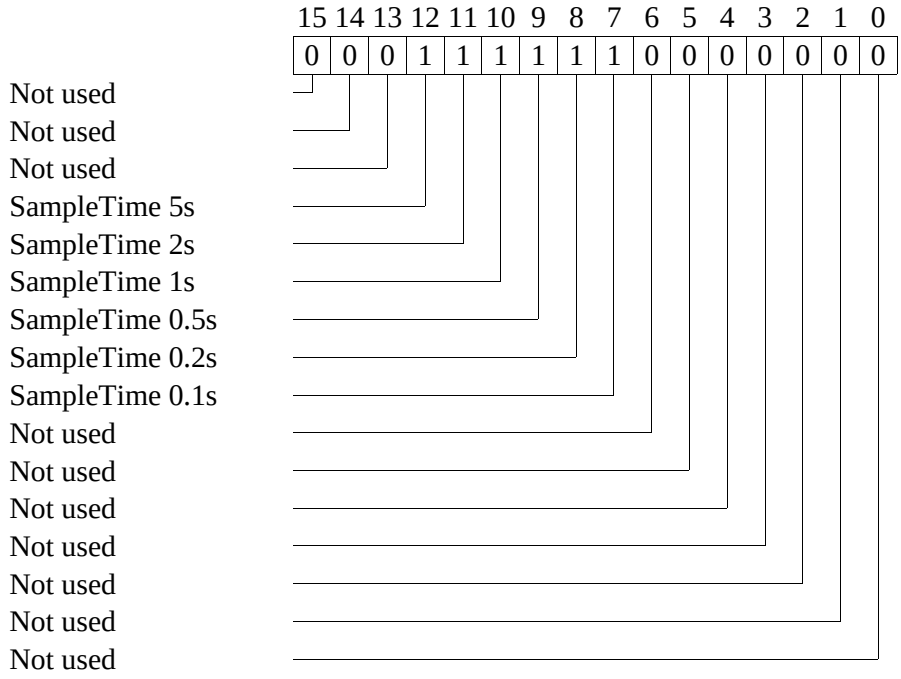
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Functions =

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

- Not used
- Not used
- Not used
- Not used
- Not used
- Not used
- Not used
- Not used
- Not used
- Not used
- Belt Weight Frequency
- Belt Weight Pulse
- Belt Weight Active
- Weight Industrial
- Weight Precision
- Channel disabled

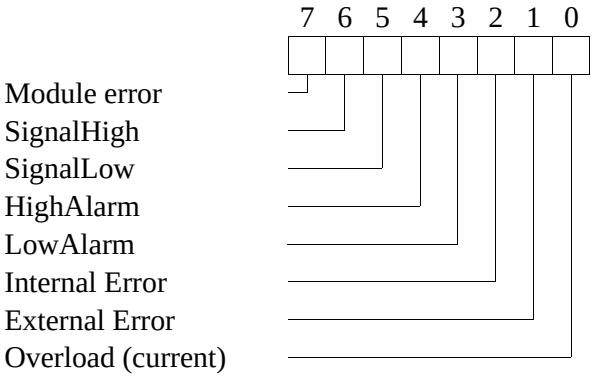
SampleTime:



SWNo 8F: CHError.

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

The 8 bit i ChError.His and ChError.Act has 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 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 (10mV) with more than 10 % and ChConfig.Enablebit[6] = TRUE.

Bit 5 SignalLow is set if the input signal falls below the minimum value (0mV) with more than 5 % of the maximum value and ChConfig.Enablebit[5] = TRUE.
The M+/M- or SS+/SS- wires may be exchanged. This connection error inverts the input signal and limits it to 5% of fullscale.

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

Bit 3 LowAlarm is set if NetWeight < 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 An external error is indicated. Please check connections and load cells.
A load cell wire may be wrong connected or a load cell may be defective.
The voltage at M+ or M- terminal may be more than 8% from midpoint of supply voltage due to load cell tolerances (unloaded load cell). This External Error condition can be disabled by setting ChConfig.Enablebit[2] to FALSE.

Bit 0 An overload error is indicated. Please check connections and load cells. Current from S+ or S- terminals is too large or external voltage is connected. A short circuit may exist, or load cell resistance is too low.

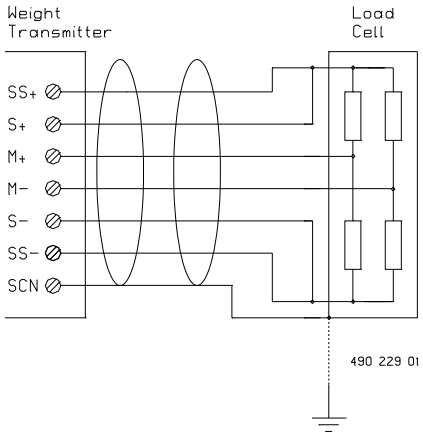
Note: When an error disappears, data can be invalid for a period, depending on the setting of SampleTime and NoOfSamples in ChConfig.

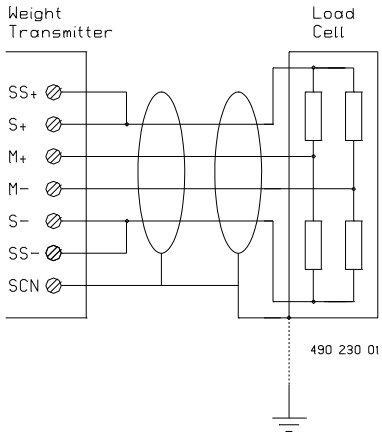
5.1 Connections to weight input.

The Weight Transmitter uses a 6-point measurement technique to compensate for cable resistance and thermoelectric voltages generated in junctions.
The notation of the 7 terminals associated with the weight input are:

SS+, SS-:	Positive and negative Supply Sense
S+, S-:	Positive and negative load cell Supply voltage
M+, M-:	Positive and negative Measuring signal
SCN:	Termination of cable Screen

Connection of load cell with shielded 6-core cable.
Load cells with shielded 6-core cable uses a 6-point measurement technique. The connection is straight forward with one wire in each screw terminal as shown in the figure. The cable can be shortened to the desired length.





Connection of load cell with fixed length shielded 4-core cable.
Some load cells are delivered calibrated with a fixed length shielded 4-core cable. This cable should not be shortened since it is an integral part of the calibrated load cell. Jumpers must be mounted between the S+ and SS+ terminals and the S- and SS- terminals to establish the necessary Supply Sense feedback to the Weight Transmitter.

Grounding of load cells.
The module is isolated from the power supply to allow grounding of the load cells. When connecting external equipment to the digital inputs or outputs an external isolated power supply must be used to avoid ground loops. If the connected load cells are isolated from ground, the digital inputs/outputs can be supplied from the 24VIN+ and 24VIN-terminals.

5.2 Parallel connection of load cells.

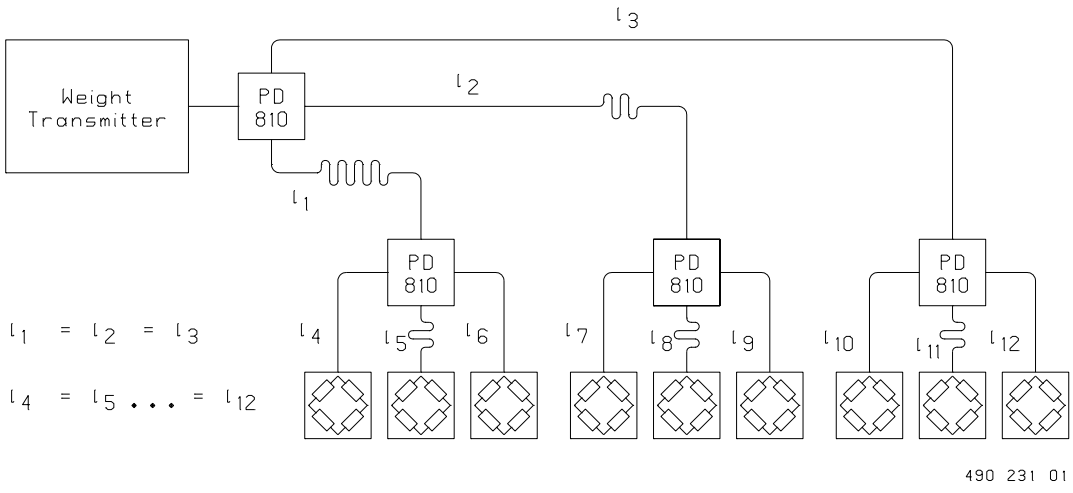
Parallelled load cells must be identical (same type and sensitivity) and connected with equal length and type cable, to obtain correct summing of signals.
The PD 810 Cable Junction Box should be used if more than 2-3 wires are to be connected to the same screw terminal.

Maximum number of load cells connected to a Weight Transmitter.
The number of load cells that can be parallelled at a Weight Transmitter depends of the specified input resistance of the load cells. The PD 3230 Weight Transmitter can supply parallelled load cells down to a load resistence of 60 Ohms and the PD 3235 can supply parallelled load cells down to a load resistance of 30 ohms.

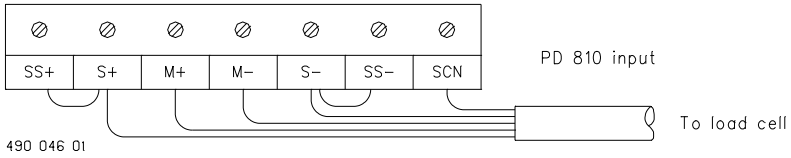
Calculation:	Max no. of load cells = Input resistance / Min load resistance
Example (PD3230):	600 Ohms per load cell / 60 Ohms = 10 load cells
Example (PD3235):	300 Ohms per load cell / 30 Ohms = 10 load cells

Note that dummy load cells also counts in the calculation.

Example 1: Parallel connection of 9 load cells.
In this example 9 load cells are connected to a Weight Transmitter via four PD 810 Cable Junction boxes. Cables at the same level in the diagram must be loaded with the same number of load cells and have equal length to obtain correct summing of signals. All connections should be made with shielded 6-core cable.



If load cells with fixed length shielded 4-core cable are used jumpers must be mounted from S+ to SS+ and from S- to SS- at each load cell input at the PD 810 cable Junction Boxes.

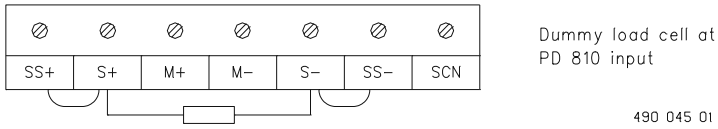


Example 2: Parallel connection of 8 load cells.
If 8 load cells are to be connected to a Weight Transmitter a dummy load cell must be connected between the S+ and S- terminal at the unused PD 810 input to obtain equal load of the cables.

A dummy load cell is simply a resistor with the following specifications:

Resistance [Ohms] = Specified input resistance of load cells +/- 1%.
Power [W] >= 0.25 W.

Typical values: 604/348 Ohms 1%, 0.25 W.



When connecting 4 load cells, no dummies are needed since equal load can be obtained by leaving one input open at each PD 810.

Combinations with 5 or 7 load cells always need dummy load cells.

6 Calculator channel (channel 9).

The Weight Transmitter 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.

The speed of a calculator programme in a module depends on the setup of the individual channels and the amount of P-NET communication. In the weight transmitter it is not recommended to use the Calculator Channel when the Weight Channel Functions = $X7.

Variables on Calculator channel. Channel identifier: **Calculator**

SWNo	Identifier	Memory type	Read out	Type
90	UniversalA	RAM Auto Save	Decimal	Real
91	UniversalB	RAM Auto Save	Decimal	Real
92	UniversalC	RAM Init EEPROM	Decimal	Real
93	UniversalD	RAM Init EEPROM	Decimal	Real
94	UniversalE	RAM Init EEPROM	Decimal	Real
95	UniversalF	RAM Init EEPROM	Decimal	Real
96	UniversalG	RAM Init EEPROM	Decimal	Real
97	Universal	RAM Read Write	- - - - - -	Array20Real
98	UserTimer	RAM Read Write	Decimal	Real
99	RunEnable	RAM Init EEPROM	Binary	Boolean
9A	LookUp1	EEPROM RPW	- - - - - -	Array20Real
9B	LookUp2	EEPROM RPW	- - - - - -	Array10Real
9C	LookUp3	EEPROM RPW	- - - - - -	Array10Real
9D	ProgramStep	EEPROM RPW	- - - - - -	Array200Word
9E	ChType	PROM Read Only	- - - - - -	Record
9F	IOChError	RAM Read Only	Binary	Record

SWNo 90-96: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 97: 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.

SWNo 98: 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 99: RunEnable
This variable is used to start and stop program execution in the calculator channel. When RunEnable is set to TRUE, the program always restarts from the first instruction line. RunEnable should be set to FALSE before a program is downloaded.

SWNo 9A: 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 9B-9C: 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 9D: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 programme is 200.

The execution time for each instruction is approximately 1 ms. By disabling unused automatic functions or channels in the module, the operating speed can be maximized. Because of speed requirements it is not recommended to run calculator programmes if the Weight Channel SampleTime = 0.1 s (Functions = $X7).

Some typical instruction execution times are shown below. The measuring conditions are: All digital outputs are configured as 50 % duty cycle outputs and the Weight Channel is configured to SampleTime = 0.2 s (Functions = $28).

Instruction	**Typ. time**
Move CR2:6,Acc	0.4 ms
Move Acc,CR2:4	0.4 ms
Move 9,CR2	0.4 ms
Sub 123.4	0.8 ms
Add 123.4	0.8 ms
Div 123.4	1.0 ms
Mul 123.4	1.0 ms
LookUp CR2:#A	12.0 ms

Refer to the PD Calculator Assembler Manual (PD no. 502 061) for a list of the available instructions and information on how to programme the calculator.

SWNo 9E: ChType

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

```
Record
    ChannelType: WORD;          (* Offset = 0 *)
    Exist: Array[0..15] Of Boolean;   (* Offset = 2 *)
    NoOfProgramStep: WORD;      (* Offset = 4 *)
end
```

ChType has the following value:

ChannelType = 7

Exist =

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

NoOfProgramStep = 200

SWNo 9F: 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;
```

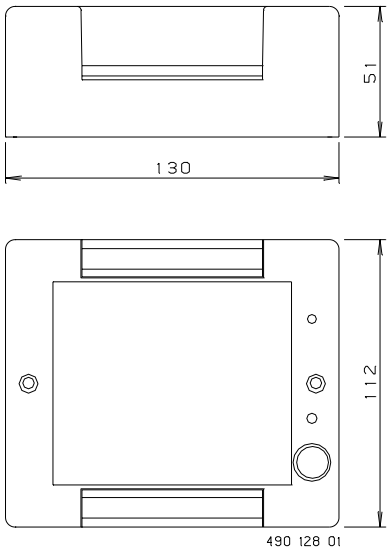
7 Construction, Mechanical.

The Weight Transmitter 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):

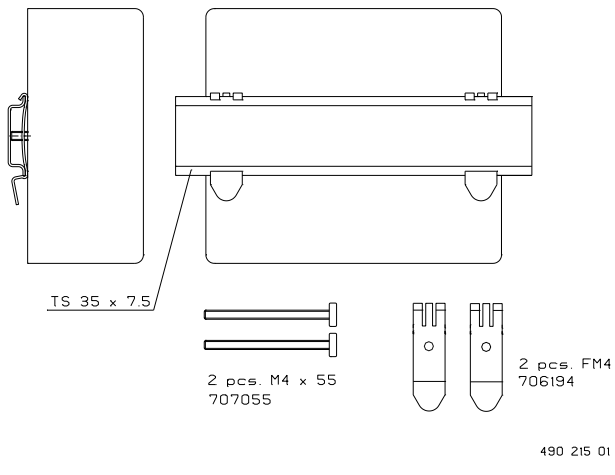


Materials:

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

Weight: 400 gram.

Rail mounting:



8 Specifications.

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

All specifications are respected in the approved EMI conditions. EMC test specifications for the Weight Transmitter are available in a separate document, PD no. **506 020**.

8.1 Power supply.

Power supply DC:	nom.	24.0 V
	min.	20.0 V
	max.	28.0 V
Ripple:	max.	5%
Power consumption (load cell resistance 30 Ohm):	typ.	2.3 W
Power consumption (load cell resistance 30 Ohm)	max.	5.0 W
Current at power up:	max.	250 mA
Fuse: 1 A time delay		

8.2 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
Pulse duration:	min.	10 ms
Counter frequency:	max.	50 Hz
Belt weight frequency input via DI6:	max.	100 Hz

8.3 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 current > 2 A):	max.	10 ms
One-shot and duty-cycle resolution:		125 ms
FeedBack update time:		500 ms
Load current measurement:		
Accuracy:		± 19 mA
Resolution:		12.5 mA
Repeatability:		± 12.5 mA
Current measurement update time:		125 ms

8.4 Weight input.

Load cell supply, alternating DC:	max.	5 V
(Short circuit proof)		
Load resistance (PD 3230):	min.	60 Ohm
Load resistance (PD 3235):	min.	30 Ohm
Sensitivity:		2 mV/V ± 0.2 %
Linearity error:	max.	0.02 %

Measuring time (fully integrating) settings:		
SampleTime = 0.1 / 0.2 / 0.5 / 1 / 2 / 5 s		
Averaging function settings:		
NoOfSamples = 1 / 2 / ... / 32		
Averaging time (SampleTime * NoOfSamples):		0.1 - 160 s
Resolution (averaging time = 0.1 s):	typ.	10 000 d
Resolution in % of fullscale:	typ.	0.01 %
Resolution (averaging time = 1.0 s):	typ.	100 000 d
Resolution in % of fullscale:	typ.	0.001 %

Belt weight frequency input via DI6:	max.	100 Hz

8.5 Ambient Temperature.

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

8.6 Humidity.

Relative humidity:	max.	95 %

8.7 Approvals.

Compliance with EMC-directive no.:	89/336/EEC
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

PD 3230 / PD 4000 pattern approved non-automatic weighing system: EN 45501
Refer to PD Manual 502 073 01 for further information.

9 Survey of variables in the Weight Transmitter.

SWNo	Service channel 0	Digital I/O 1–4	Digital input 5–6	Common I/O 1–6 7	Weight channel 8	Calculator 9
x0	NumberOfSWNo	FlagReg	FlagReg	OutFlags	Weight0	UniversalA
x1	DeviceID	OutTimer		InFlags	Weight1	UniversalB
x2		Counter	Counter		NetWeight	UniversalC
x3	Reset	OutCurrent			Flow	UniversalD
x4	PnetSerialNo	OperatingTime	OperatingTime		Frequency	UniversalE
x5					FlagReg	UniversalF
x6		FBTimer			Tare	UniversalG
x7	FreeRunTimer	FBPreset			HighLevel	Universal
x8	WDTimer	OutPreset			LowLevel	UserTimer
x9	ModuleConfig	ChConfig	ChConfig		ChConfig	RunEnable
xA	WDPreset	MinCurrent			Factors	LookUp1
xB		MaxCurrent			FullScale	LookUp2
xC					ZeroPoint	LookUp3
xD	WriteEnable	Maintenance			Maintenance	ProgramStep
xE	ChType	ChType	ChType	ChType	ChType	ChType
xF	CommonError	ChError	ChError	IOChError	ChError	ChError

502063AD

Contents