AK50

PROCESS MOISTURE LOGGER FOR ON-LINE USE IN INDUSTRY

User's Manual for

Profibus DP Fieldbus

AK50 and Its Derivatives

Fully supported by V2.0S and later

PART #700184

Made in Finland
Manual printed in
Monninkylä, Finland

2016-46

Visilab Signal Technologies Oy
Address:
Sepäntie 4
FI-07230 Monninkylä
FINLAND

Tel.:
+358-45-635 4885

VAT FI 06312080

www.visilab.fi
Y 0631208-0 ALV Rek.
Mäntsälä Reg. 365.258

CONDITIONS OF GUARANTEE, COPYRIGHT NOTICE AND LIABILITIES OF THE MANUFACTURER

The manufacturer (Visilab Oy) grants a guarantee of two years for the buyer of AK50 moisture meter from the date of purchase. The guarantee covers all faults and misalignment which are in the equipment at the moment of purchase including those which appear during the guarantee period. The manufacturer is liable of repairing the instrument without cost to the buyer. The manufacturer can ship a new instrument of equivalent value and status if considered as a better solution than repairing. The buyer is liable of paying the freight costs to the factory of the faulty unit. The unit must not be sent to the manufacturer without a permission from the manufacturer. Units sent without a permission will be repaired at the cost of the buyer.

The guarantee does not cover wearing parts, like batteries, lamps or motors. The guarantee does not cover faults caused by errors or neglects of the user nor those faults which are caused by deliberate breaking. The guarantee does not cover faults caused by incorrectly installed cables or conductors. The guarantee does not cover any damages to the user or to any third party independently of the way how the instrument has been used. The guarantee does not cover faults caused by natural phenomena like lightnings or floods, nor user errors like dropping the unit. The guarantee is void if the unit is sold to any third party. All faults which are not covered will be repaired at the cost of the buyer.

If opening of the instrument has been attempted at those parts which are not intended for the user, the manufacturer can refuse to repair or service the instrument. Then the instrument will be shipped back to the buyer at the cost of the buyer. Such parts are the light source, the optical head and parts on the electronics board. The instrument can be opened only strictly according to the instructions in this manual and should not be disassembled unnecessarily.

Visilab Oy reserves all rights to changes and modifications in the looks, specifications, optical and electronic design, electronic and software interfaces and computer programs, and also the right to change the retail prices of the instrument or its parts without any notice to present or potential customers. All copyrights and design rights belong to Visilab Oy. The PC programs, which have been sold to the buyer, can be used and copied freely for his own use but can not be sold to any third party.

The manufacturer is not responsible for any casualties, damages or accidents which the user has caused directly or indirectly with this AK50 instrument, either to himself or to any third party.

NOTE: The Profibus DP interface configuration has been changed starting from internal SW V0.60DP and later. The old configuration was 4 bytes input/4 bytes output and the new configuration is 16 bytes input/16 bytes output. All the old commands are still valid as such and their parameters are placed in the IO array at the same positions. The extended command set in V0.60DP will use the other array positions 4...15 as well for passing information in/out. Starting from V0.71DP the last byte in the input array bi16 always returns the slave status value and the byte before that, bi15, always returns the optional command identifier which is sent by the master. If your meter's embedded software is of an older version, it does not support these new features nor any of the extended commands. Please, contact Visilab to upgrade your meter's software at a low cost. You will get all the other new features in the meter as well.

Contents

List of Commands Available via Profibus DP

General Commands (decimal and hexadecimal values)

	decimal	hex
Get the General System Status I7GSTATUS	76	0x4C
Get the Second System Status I7G2STATUS	86	0x56
Get the Third System Status I7G3STATUS	89	0x59
Read the Filter Characteristics I7GFILTER	50	0x32
Change the Filter Characteristics I7SFILTER	49	0x31
Get the Locking Status I7GETLOCK	53	0x35
Set the Locking I7GAINLOCK	51	0x33
Set the Autoranging I7GAINOPEN	52	0x34
Get the Lamp Status I7GLAMP	74	0x4A
Get the Chopper Speed I7GFREQ	60	0x3C
Get the Usage Counter Hours I7GETUSG	28	0x1C
Set the Terminal Mode ON (Keyboard Mode) I7STERM	47	0x2F
Set the Packet Protocol Mode ON I7SPACKET	75	0x4B
Get the Low Power Mode Status I7GETLPM	37	0x25
Set the Low Power mode ON/OFF, (ON = Low Power Mode, OFF = Normal Mode) I7SETLPM	38	0x26
Get the Voltage Output Source I7GVOUT	88	0x58
Set the Voltage Output Source I7SVOUT	87	0x57
Get the Cooler Enable Status I7GCOOLING	90	0x5A
Get the Cooler Temperature I7GCOOLTMP	93	0x5D
Get the Cooler On/off Status I7GCOOLON	94	0x5E
Get the Cooler Linking Status I7GCOOLINK	95	0x5F
Get the Cooler Status I7GCOOLSTA	97	0x61
Set the Cooling Enable I7SCOOLING	92	0x5C
Set the Cooling Linking I7SCOOLINK	96	0x60
Set the web temperature filter setting I7STLPF	99	0x63
Get the web temperature filter setting I7GTLPF	101	0x65
Set the web temperature offset I7SWEBB	102	0x66
Get the web temperature offset I7GWEBB	103	0x67
Get the head temperature alarm status I7GALM	104	0x68
Clear the head temperature alarm I7CALM	105	0x69

Calibration and Standardization Commands

	decimal	hex
Get the Current Material Entry I7GETMAT	14	0x0E
Switch to another Calibration Table in the Library I7SETMAT	15	0x0F
Get the Calibration Mode of the Current Material Entry I7GMODE	16	0x10
Set the Calibration Mode (MULTI/QUICK) I7SMODE	17	0x11
Read the Calibration Table Entry I7RXMAT	27	0x1B
Set the Calibration Table Entry I7TXMAT	26	0x1A
Set the Offset for Standardization I7SETTIM	39	0x27
Set the Standard Moisture Value for Standardization I7SSTD	72	0x48
Get the Material Entry Number Used in Standardization I7GSTDM	71	0x47
Set the Offset Value for Standardization I7SSHIFT	67	0x43
Get the Offset Value Resulting from Standardization I7GSHIFT	68	0x44

	decimal	*hex*
Get the Standard Value Set for Standardization I7GSTD	73	0x49
Standardize I7STDZE	69	0x45
Set the Standard Material Entry Number I7SSTDM	70	0x46

Data Acquisition Commands

	decimal	*hex*
Get the Optical Head Temperature I7GHEAD	79	0x4F
Get the Optional Extra Web Temperature I7GWEB2	100	0x64
Start Sending the Head Temperature instead of the Web Temperature I7GETTMP	46	0x2E
Start Sending the Web Temperature Instead of the Head Temperature I7GWEB	48	0x30
Get expansion module signal I7GXMOD	108	0x6C

Memory Bank Commands

	decimal	*hex*
Read the Number of Samples in the Current Bank I7GETDM	35	0x23
Read the Bank Number I7GBANK	55	0x37
Get the Autotimer Mode I7GAMODE	59	0x3B
Get the Autotimer Status I7GETAUTO	43	0x2B
Clear the Current Data Series (or Bank) I7CLRSER	21	0x15
Take a Sample into the Current Data Series (or Bank) I7SAMPLE	36	0x24
Set the Autotimer ON I7AUTOON	41	0x29
Set the Autotimer OFF I7AUTOOFF	42	0x2A
Select the Bank I7SBANK	54	0x36
Set the Autotimer Mode I7SAMODE	58	0x3A
Get the Autotimer Interval in 0.1ms Units I7GETTIM	40	0x28
Get the Current Batch Size I7GBATCH	57	0x39
Set the Current Batch Size I7SBATCH	56	0x38
Set the Autotimer Interval in Seconds I7SETTIM	39	0x27
Get Samples from the Current Memory Bank I7TXSER	20	0x14
Copy the Temperature Series to Bank4 I7COPYT	98	0x62
Set the Current Burst Size I7SBURST	112	0x70
Get the Current Burst Size I7GBURST	113	0x71
Set the Burst Mode I7SBUM	114	0x72
Get the Burst Mode I7GBUM	115	0x73
Get the Burst Mode Item Count I7GBUC	116	0x74
Clear the Burst Mode Item Count I7CBUC	117	0x75

Text String Commands

	decimal	*hex*
Get the Unit for Moisture I7GUNIT	13	0x0D
Get the Current Material Entry Name, part 1 I7GMATNM	31	0x1F
Get the Current Material Entry Name, part 2 I7GMATNM2	77	0x4D
Get the Current Library Name I7GLIBNM	29	0x1D
Get the Meter's Identifier String 1 I7TEST	10	0x0A
Get the Meter's Identifier String 2 I7TEST2	78	0x4E

	decimal	*hex*
Get the Meter's Identifier String 3 I7TEST3	80	0x50
Set the Current Library Name I7SLIBNM	30	0x1E
Set the Unit for Moisture I7SUNIT	12	0x0C
Set a Material Name, part 1 I7SMATNM	81	0x51
Set a Material Name, Part 2 I7SMATNM2	82	0x52
Get expansion module name I7GXNAME	110	0x6E

Special Commands

	decimal	*hex*
Get the LAN Addresses I7GLAN	84	0x54
Set the LAN Addresses I7SLAN	85	0x55
Initialize the Profibus DP slave I7DPINIT	65	0x41
Send a Short Pulse to the LED Indicator (if available on the connector panel) I7BEEP	34	0x22
Start Fast Fourier Transform in the Meter I7FFT	83	0x53
Get expansion module number I7GNXMOD	109	0x6D
Send a command to the expansion module I7SXCOM	111	0x6F

1. Introduction and Taking into Use

This document instructs you on how to use the standardized industrial fieldbus interface **Profibus DP** with your meter. Refer to your existing **Profibus DP** operating manuals delivered by its manufacturer. For other features of the **AK50** moisture meter refer to PC User's Manual and to the User's Manual of the instrument.

General

AK50 and its derivatives are very high-speed process surface moisture transmitters for requiring conditions in paper machines and other comparable applications. The moisture measurement speed is at least 400 Hz. That speed can be exploited in different ways. One can acquire the latest moisture value into some control system in digital form either via the RS232/485 serial port (typically a PC) or via **Profibus DP** fieldbus (max 12 Mbauds RS485). The moisture signal can be read also in analog voltage form (0...+10 V, +/-5 V or 0...+5V). The moisture signal can be sampled and collected to the unit's own battery backed data memory at regular intervals from 2.5 ms to 36000 s. The data can later be retrieved from the memory as a data series. One external trigger line is available for starting either continuous sampling or sampling a preset batch.

The **Profibus DP** (DP = Distributed Processing) interface complements your fixed or traversing control system in an excellent way. With it you can acquire moisture and temperature readings from the meter to be used elsewhere in your system via the fieldbus. You can also switch to another calibration table, while operating, with a simple command. Also the calibration mode can be changed between QUICK and MULTI. The Low Power or Normal modes can be set to save the meter during a production stoppage. There are about one hundred different commands available. Practically all operations can be done without a PC. **AK50** works as a standard DP slave in the fieldbus. **Profibus DP** allows the use of very long operating distances, up to 1200 meters at lower data transmission rates. It can also handle quite a large number of slaves simultaneously and you can have several masters as well. One advantage is its international standardization (EN 50170 and DIN 19245). Compatible modules can be supplied by several manufacturers. It is a plug and play game. You can detach a module from the network without causing any harm to other nodes unless they are dependent on the data sent by this slave. You can plug in any slave into a hot connector and the master immediately takes it as part of the fieldbus network. Refer to your existing **Profibus DP** documentation for more details. More information of the fieldbus can be supplied by your local representative for Profibus components and systems.

Installation

For installing and configuring the **Profibus DP** for **AK50** successfully the following items should be available (refer to the complete packing list in the user's manual):

1. **AK50** *instrument*
2. *Program diskette*
3. *User's manual (this)*
4. *Power source with cables*
5. *Connection cable with connectors and distributing box*
6. **Profibus DP** *cable into the distribution box*

You will need also a **Profibus DP** cable with a D9 male connector which is connected to the existing network.

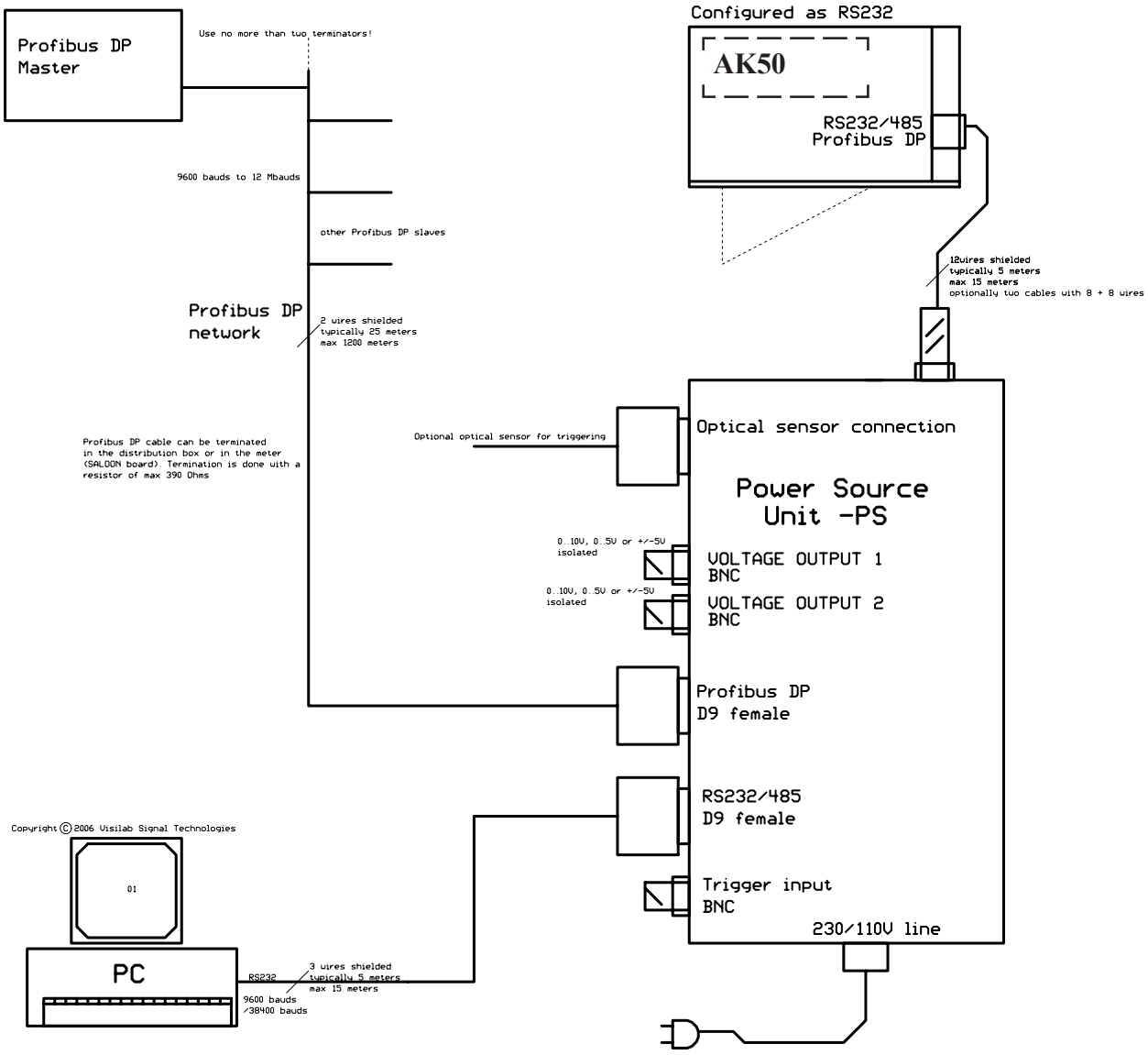

Figure 1. Assembly of electrical cables in AK50 for use via RS232 with a PC and Profibus DP simultaneously. The drawing is not to scale. The wire positioning may vary on the distribution box. The PC connection is optional in actual operation.

Connecting the Cables

After having checked the instrument operations as instructed in other manuals first, you can proceed as follows. Connect the cables as instructed in the **instrument user's manual** and test the PC interface operation as instructed in the PC User's Manual. Now we assume that the connections are all right and the PC part of the testing was successful.

The distribution box has a D9 connector for the PC serial cable and another similar for operating with the **Profibus DP** fieldbus (see Figure 1.). Connect the DP master to the **Profibus DP** connector. It does not matter what mode it is in, powered off, Stop, Clear or Operating.

The connector type used for the **Profibus DP** at the meter back panel is **Bulgin Mini Buccaneer** IP68 codes PX0800 (free body), 12735/1 (socket carrier) and SA3149/1 (socket). The pin numbering is the same as in the schematic (from 1 to 8) in Appendix 3.

Configuring

To actually configure your new DP slave you need to have some configuration tool for the fieldbus. The purpose of configuration is to tell the DP master what kind of input data can be retrieved from the slave and what kind output data can be sent to it. The most important specifications of **AK50** as a DP slave are the following:

- 16 bytes input:
  - 1. byte: moisture value whole
  - 2. byte: moisture value fraction
  - 3. byte: temperature value whole
  - 4. byte: temperature value fraction
  - 5. ..16. bytes parameters input
- 16 bytes output:
  - 1. byte: reserved (= 0)
  - 2. byte: command
  - 3. byte: data
  - 4. byte: reserved (=0)
  - 5. ..16. bytes parameters output
- vendor ID: 0 (=zero)
- time-out period: 20 ms
- slave name: IRMA7D-001
- Watchdog: ON
- slave address: 3 (initially, can be changed with the aid of PC interface)
- transmission rate: 1500 kbaud (default value for starting, the slave will adapt to any rate from 9600 baud to 12 Mbaud)
- freeze mode: OFF
- sync mode: OFF
- minimum slave interval: 5 ms

Refer to Appendices 1. and 2. for the contents of the configuration database text files for other vital specifications used in configuring the slave. Note that accepting the slave is not successful if most of the parameters are not set correctly. The master can communicate only with a slave which has been configured correctly.

Troubleshooting Hint

The **Profibus DP** cable has two wires only plus a protecting shield in the minimum configuration. If the slave refuses to operate at all, swap the two pins in the cable connector. That should do it. Refer to Appendix 3. for schematics of the distribution box.

2. Using the PC Program to Help Configure

General Notes

In addition to other features mentioned in the PC User's Manual, the PC program offers you a way of configuring the meter for operation in **Profibus DP**. That is done in the keyboard mode or in the graphical mode in menu F5. In the PC program, press F9 to enter the keyboard mode. You can always press the F9 again to get back to the graphical interface.

Menus and Settings

To change settings, supposing you are in the measuring state, press 'm' to go to the main menu which looks like this:

1=Series 2=Calibr.
3=Mater. 4=Profibus
5=Service 6=LPF:MEDM
7=Unit 8=Comm´s

Then press '4' to go to the **Profibus DP** menu:

0=Slave 8=ACTIVE
2=SetR 3=ReadR
6=IDLOW 7=IDHIGH
4=List 5=Init 9=Loop

Practically the only selection you may have to use in this menu is the option '0' for setting the slave address. **The default address is 3.** If your DP system already has a slave with that address you may have to change the address of **AK50**. Press '0' and a simple dialog asks you to edit the given address. Accept it after modifying it. To make it effective, you have to either power-off the meter or make a slave initialization by pressing '5'. Avoid using addresses reserved by other slaves or masters.

If by any chance, the **Profibus DP** support in this instrument has been deactivated, you can reactivate it here by pressing '8'. Also, if you wish to deactivate the fieldbus (not recommended) in case you have no use for it, press '8' to deactivate it. This menu always shows what will be the result of pressing the '8' key. Also a short text note will appear momentarily indicating the result of the operation. If the system has been inactive and you plan to take it into use, perform the initialization here as well ('5').

There is one situation which will automatically *deactivate* the fieldbus: ***When you perform the factory settings in the service menu. That, of course, should not be done without a very good reason. You will loose your calibration library in that process unless you have saved it.***

It is not recommended that you touch other alternatives in this menu in normal use. They are intended

for service personnel for diagnostic purposes. If by accident, you attempt to use them, please press '5' to initialize the fieldbus with correct settings. There is a possibility of changing the slave's ID. The ID's low byte is set in '6' and the high byte is set in '7'.

The meter has a DATAEX indicator lamp on the middle CPU board which is visible when the unit access cover is opened. When the lamp is off, the **Profibus DP** slave has been accepted as a correctly configured slave and data exchange is possible. If the lamp is ON, the slave is off-line and does not belong to the DP network. The master will always indicate successful slave parametrization and configuration. Refer to the DP master's manuals for operating the master. Refer to the **AK50** User's Manual for locating the indicator lamp.

3. Operating the Slave via Fieldbus

General

The **Profibus DP** fieldbus is based on slave polling and token passing between masters. Slave data is cyclically acquired and new output data is sent to those slaves that require it. Other features, like the PC interface, can be freely used simultaneously.

You can access the current moisture and temperature values of **AK50** with any **Profibus DP** master. The master can be a programmable logic controller (PLC), a more advanced industrial CPU or a PC-based master. The controllers have an embedded program which is usually downloaded from some workstation before the control system is set into operation. In a PC-based master there is a small add-on card having the master communications processor and other fieldbus hardware. The master always has software that can manage the fieldbus protocol and data exchange with other nodes in the fieldbus.

Each slave to be connected to the fieldbus has a number of special features which make it different from other slaves. These features have to be determined with some configuration tool. The data are saved into a text file and a binary database. The database is loaded to the master. Then it can recognize the new slave and communicate with it correctly.

Passing Commands

The procedure for sending any of these commands is the following. The default command that should normally be sent to a slave consists of zeros only. That will make it sure that no pending commands are processed. Note that it is required to send the command **only once** to a slave. Sending a command repeatedly will redo the same thing and possibly overload the slave. Do not send a command without a data byte if the command requires it. Refer to Appendix 4 for details.

The calibration table change is done immediately and the next moisture values will be linearized according to it. There is a short calculation period of less than 1 ms associated with the table change, during which time the acquired moisture value may be incorrect. That is however, usually not important due to the nature of the situation where this command is used.

While standardizing, note the following. The moisture standard should be placed under the meter in the proper position before starting this operation. If using a traverse, the meter head should be driven to that position where the standard is placed. In order to standardize correctly, the corresponding moisture reading of the standard should be set in the menu system as well as the calibration table which should be used for this purpose. The standardization menu can be found in the Calibration menu: **6=Std-ize**. There you will find the setting options. After having made this once you can later do the standardization automatically just by driving the head and sending the command. Nothing else is required.

Input Data

The input data information in **AK50** consists of the following:

o 16 bytes input (bi1...bi16):
- o 1. byte bi1: moisture value whole, **a continuously changing signal**
- o 2. byte bi2: moisture value fraction, **a continuously changing signal**
- o 3. byte bi3: temperature value whole, **a continuously changing signal**
- o 4. byte bi4: temperature value fraction , **a continuously changing signal**
- o 5..14 bytes bi5...bi14: parameter return values, these bytes **remain available until another command is sent which requires the use of any of these bytes.**
- o 15...16 bytes bi15..bi16 bi15 contains the returned freely usable command identifier (ref. to next page) and bi16 contains **always** the slave status **sta**, which is described in more detail with the command I7GSTATUS.

The actual values are calculated as

moisture = bi1 + (bi2 / 100.0)

and

temperature = bi3 + (bi4 / 100.0)

Note that the moisture value may become negative in some special circumstances. That is indicated by the whole value as a signed character (255 = -1, 254 = -2 etc). The input data are retrieved usually very often. The cyclic rate depends on the number of slaves and masters in the system. Also the data transmission rate will affect it. Typical values are 50 to 100 Hz in a system with a few slaves at 187.5 kbaud rate. Naturally, the output data are sent simultaneously in each transaction. Note that the temperature data is the **web** temperature if your meter has the standard IR thermometer assembled. Else the reading will be the **head** temperature. The selection has been made at the factory. If you wish to change it you can use the commands **I7GWEB** and **I7GETTMP**. The setting stays until set again, irrespective of powering down the meter.

Output Data

The output data should be handled in the following way if any commands are sent to the slave. Using them is not required. If output is not used, the output data should always be set to zero, all four bytes.

o 16 bytes output (bo1...bo16):
- o 1. byte bo1: command identifier **cid**, available for free use
- o 2. byte bo2 : command **com**
- o 3. byte bo3: data *
- o 4. byte bo4: data *
- o 5..16 bytes bo5...bo16: data *

The general structure of the arrays input data and output data use the following pattern:

Example Command Explanation Header:
I7COMMAND DD 0xHH

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The fields filled with some identifiers mean that they are already fixed in their use. Their use is not changed in any command in any way and they reserve these fields permanently. The various bytes left empty are filled in each command differently. The short names of various bytes used to describe the bytes are below.

cid	command identifier (optional) - user defined
com	command - user defined, e.g. I7GSTATUS as a byte
*	general data, not defined more precisely
s	string data, one character of it, may be an end marker also (zero)
c	a single character data, unsigned char
i	integer data, lower or upper byte
l	long integer data, any of the four bytes
f	floating point data formatted as two integers, the whole part and the fractional part. Typically the data is put into four or two bytes
sta	status byte, unsigned char
0	zero, set in command, returned in input data
(empty)	field not used, ignore any return values
mw	moisture whole
mf	moisture fraction
tw	temperature whole
tf	temperature fraction
xw	expansion module signal whole
xf	expansion module signal fraction

The first byte is has no immediate effects. The typical application is to use it as a sequential number which is **always returned with the next updated input data (bi15) which has the data required by the command**. One can easily separate input data belonging to each command sent and thus obtain the best possible polling speed. If one does not need it, one can forget it. The following commands are supported at this time (for details refer to the User's Manual). They are classified according to their purpose.

General Commands

Get the General System Status:
I7GSTATUS 76 0x4C

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 |sta|   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the status byte. The data will be valid until the next command requiring the use of same bytes in return data is sent out. The form of the input data is the following:

- o 5. byte bi5: 0
- o 6. byte bi6: status byte

The same data value is **always** returned in the last byte also, independent on the commands used:

- o 16. byte bi16: status byte

The 5th byte is kept as zero and is reserved for future use. The status data byte is bit-controlled and can have the following values.

bit	name	value 0	value 1
0:	Low power mode	**Normal oper.**	Low power mode
1:	Keyboard mode	**Packet mode**	Keyboard mode
2:	Calibration mode	QUICK	**MULTI**
3:	Autotimer mode	**Batch**	Auto (continuous)
4:	Autotimer:	**OFF**	ON
5:	T-autotimer:	**OFF**	ON
6:	Gain locking:	**OFF**	ON
7:	Lamp OK:	lamp fault	**lamp OK** (note 1.)

Flag settings marked with **bold** are default values for general use when applying mainly **Profibus DP**.

Note 1. If your moisture meter supports by hardware this feature, the flag is valid, else it will always be marked as OK, even in case of lamp failure. However, it is possible to identify from an operating meter's moisture signal that its light source is not working as the signal changes with large amplitudes (like between +/-100% moisture) without any correlation of the target. Refer also to the next command.

Get the Second System Status:
I7G2STATUS 86 0x56

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 |sta|   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the status byte. The data will be valid until the next command requiring the use of same bytes in return data is sent out. The form of the input data is the following:

- o 5. byte bi5: 0
- o 6. byte bi6: status byte

The 5th byte is kept as zero and is reserved for future use. The status data byte is bit-controlled and can have the following values.

bit	name	value 0	value 1
0:	burst mode	off, normal oper.	**on, BURST mode**
1:	analog output	**moisture**	web temperature
2:	quiet booting	**no**	yes
3:	linked autotimers	**no**	yes
4:	web OK, no break	FALSE	**TRUE (depends on web)**
5:	session start phase	**OFF**	ON
6:	reflective surface	**false**	true
7:	dark surface	**false**	true

Flag settings marked with **bold** are default values for general use when applying mainly **Profibus DP**. The burst mode is a special operating mode available for piecewise web measurements. The analog output can be configured in the menu system (Vscales). Quiet booting is required when using a LAN and RS485. The setting is done in menu 5 Services. The autotimers for moisture and web temperature can be linked with the temp as a slave in Unit menu - temp series. The web OK bit is an attempt to indicate web breaks (active low). They are interpreted as ultimate limits of darkness or reflection for at least three seconds duration. The bit will be automatically updated. The session start bit is set if the meter has just booted and this will take some 30 seconds. Reflective and dark surface indicators will be used momentarily in normal operation if conditions change. If the bit is set for a prolonged period, something may be wrong. Refer also to the command before this one.

Get the Third System Status:
I7G3STATUS 89 0x59

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 |sta|   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the status byte. The data will be valid until the next command requiring the use of the same bytes in return data is sent out. The form of the input data is the following:

- o 5. byte bi5: 0
- o 6. byte bi6: status byte

The 5th byte is kept as zero and is reserved for future use. The status data byte is bit-controlled and can have the following values.

bit	name	value = 0	value = 1
0:	cooling enable	off	**on, cooler enabled**
1:	**cooling status**	**FAILURE, more air is needed**	**OK**
2:	cooler linking	**no**	yes
3:	web break suspicion	**no break**	yes, a break is suspected
4:	web temperature filter	OFF	**ON**
5:	**overtemperature alarm**	**OFF, OK**	**ON, overheating of head**
6:	COMPOSER	inactive	active
7:	Expansion module	none	installed

Flag settings marked with **bold** are default values for general use when applying mainly **Profibus DP**. **The cooler settings are related to the meter hardware and apply only if a cooler exists.** The web break suspect bit is an attempt to indicate suspected web breaks when the gain locking is used. Refer also to the two commands before this one. The COMPOSER bit tells if the calibration expert system is installed into your moisture meter. If it is not supported, this bit reads zero.

Read the Filter Characteristics:
I7GFILTER 50 0x32

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the current filter setting. The data will be valid until the next command requiring the use of same bytes in return data is sent out. The form of the input data is the following and the setting may be one of the values in the table:

- o 5. byte bi5: 0
- o 6. byte bi6: filter setting

The setting may have any of the following values:

OFF	120	0x78
FAST	121	0x79
MEDIUM	122	0x7A
SLOW	123	0x7B
SPECIAL	124	0x7C
BOX	125	0x7D

Refer to User's Manual for more details and effects of the filter setting.

Change the Filter Characteristics:
I7SFILTER 49 0x31

```
output data bo:
-----------------------------------------------------------------
|cid|com| c |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

o 3. byte bo3: **filter** may have any of the following values:

symbol	dec. value	hex value
OFF	120	0x78
FAST	121	0x79
MEDIUM	122	0x7A
SLOW	123	0x7B
SPECIAL	124	0x7C
BOX	125	0x7D

The parameter **filter** may have any of these values causing also the corresponding filtering effect after initializing the filter system.

Get the Web Temperature Filter Status:
I7GTLPF 101 0x65

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the setting of the web thermometer low pass filter (LPF). The form of the input data is the following and the status may be one of the values in the table:

- o 5. byte bi5: 0
- o 6. byte bi6: filter status

The status may be one of the following:

symbol	dec. value	hex value
on, data is filtered	1	0x01
off, no filter	0	0x00

Refer to User's Manual for more details. Using the filter will slow down the response time but will lower the noise. This filter does not affect the EXTRA thermometer input signal in any way nor does it affect the head temperature signal.

Set the Web Temperature Filter:
I7STLPF 99 0x63

```
output data bo:
-----------------------------------------------------------------
|cid|com| c |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

o 3. byte bo3: **mode** may have any of the following values:

OFF	0	0x00	web temperature filter is off
ON	1	0x01	web temperature filter is on

Get the Locking Status:
I7GETLOCK 53 0x35

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the signal locking status of the signal amplifier autoranging system. The form of the input data is the following and the status may be one of the values in the table:

- o 5. byte bi5: 0
- o 6. byte bi6: locking status

The status may be one of the following:

symbol	dec. value	hex value
on, locked	1	0x01
off, autoranging	0	0x00

Refer to User's Manual for more details. Do not lock the gain if you do not know the resulting effects!

Set the Locking:
I7GAINLOCK 51 0x33

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The command will change the locking accordingly and it will stay until set again, independent on power cutoff. After this command, the gain is locked. Using this command will avoid signal transients in critical measurements which could be resulted by the autoranging system changing automatically the system gain. Signal clipping and resulting strong distortion may occur if light signal increases too much. Refer to User's Manual for more details. Do not lock the gain if you do not know the resulting effects!

Set the Autoranging:

I7GAINOPEN 52 0x34

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The command will change the locking accordingly and it will stay until set again, independent on power cutoff. After this command, the gain is unlocked and the autoranging system is able to adapt to varying web conditions.

Get the Lamp Status:
I7GLAMP 74 0x4A

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the light source status. Refer to I7GSTATUS for more details. The form of the input data is the following and the setting may be one of the values in the table:

- 5. byte bi5: 0
- 6. byte bi6: status

Status may be one of the following:

symbol	dec. value	hex value
OK	1	0x01
fault	0	0x00

Get the Chopper Speed:
I7GFREQ 60 0x3C

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | i | i |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the light source speed. The form of the input data is the following and the setting may be one of the values in the table:

- o 5. byte bi5: high byte of frequency (integer in Hz)
- o 6. byte bi6: low byte of frequency **(parts of 1/100)**

The actual frequency is calculated as follows:

f (Hz) = f whole + (f fractional / 100)

Get the Usage Counter Hours:
I7GETUSG 28 0x1C

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | l | l | l | l |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 to 8 are used for returning the usage counter reading in hours of active time. The form of the input data is the following:

- o 5. byte bi5: MSB of usage hours
- o 6. byte bi6: HMEDB byte of usage hours
- o 7. byte bi7: LMEDB byte of usage hours
- o 8. byte bi8: LSB of usage hours

The usage is calculated as **hours * (LSB + 256 * LMEDB + 65536 * HMEDB + 16777216 * MSB)**

Set the Terminal Mode ON (Keyboard Mode):
I7STERM 47 0x2F

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a result of this command, the Keyboard mode is turned on in the RS232 serial port interface.

Set the Packet Protocol Mode ON:
I7SPACKET 75 0x4B

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a result of this command, the Packet protocol mode is turned on in the RS232 serial port interface.

Get the Low Power Mode Status:
I7GETLPM 37 0x25

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the low power mode status. Refer to I7GSTATUS for more details. The form of the input data is the following and the setting may be one of the values in the table:

- o 5. byte bi5: 0
- o 6. byte bi6: status

Status may have one of the following values:

symbol	dec. value	hex value
Low Power	1	0x01
Normal	0	0x00

Set the Low Power mode ON/OFF, (ON = Low Power Mode, OFF = Normal Mode)
I7SETLPM 38 0x26

```
output data bo:
-----------------------------------------------------------------
|cid|com| c |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

o 3. byte bo3: **mode** may have any of the following values:

OFF	0	0x00
ON	1	0x01

Get the Voltage Output Source:
I7GVOUT 88 0x58

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the source of voltage ouput signal. The form of the input data is the following and the setting may be one of the values in the table:

- o 5. byte bi5: 0
- o 6. byte bi6: selection

The selection may be one of the following:

symbol	dec. value	hex value
Moisture	0	0x00
Web temperature	1	0x01
Head temperature	2	0x02
Extra temperature	3	0x03

Set the Voltage Output Source:
I7SVOUT 87 0x57

```
output data bo:
-----------------------------------------------------------------
|cid|com| c |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

o 3. byte bo3: **selection** having values:

symbol	dec. value	hex. value
Moisture	0	0x00
Web temperature	1	0x01
Head temperature	2	0x02
Extra temperature	3	0x03

The parameter will change the selection with a delay of 5 ms maximum to be seen at the analog output connector.

Get the Cooler Enable Status:
I7GCOOLING 90 0x5A

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the cooler enable status. The form of the input data is the following and the status may be one of the values in the table:

- o 5. byte bi5: 0
- o 6. byte bi6: cooler enable status

The status may be one of the following:

symbol	dec. value	hex value
on	1	0x01
off	0	0x00

Refer to User's Manual for more details. Do not enable the cooler unless you know the resulting effects! To use the cooler, it must be installed into your meter.

Get the Cooler Temperature:
I7GCOOLTMP 93 0x5D

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | i | i |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the cooler input temperature used in control. The form of the input data is the following and the setting may be one of the values in the table:

- o 5. byte bi5: high byte of temperature (integer in C)
- o 6. byte bi6: low byte of temperature **(parts of 1/100)**

The temperature retrieval is available only if your system **specifically** supports it. Refer to User's Manual for more details. Do not enable the cooler unless you know the resulting effects! To use the cooler, it must be installed in your meter.

The actual temperature is calculated as follows:

temperature (C) = temperature whole + (temperature fractional / 100)

Get the Cooler On/off Status:
I7GCOOLON 94 0x5E

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the cooler on/off status which may be different than the enable status due to the possible dependence on linking to Low Power mode. The form of the input data is the following and the status may be one of the values in the table:

- o 5. byte bi5: 0
- o 6. byte bi6: cooler status

The status may be one of the following:

symbol	dec. value	hex value
on	1	0x01
off	0	0x00

Refer to User's Manual for more details. Do not enable the cooler unless you know the resulting effects! To use the cooler, it must be installed into your meter.

Get the Cooler Linking Status:
I7GCOOLINK 95 0x5F

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the cooler linking status. The cooler may be linked to Low Power mode. It means, that the cooler will be turned to off when the Low Power mode is active. The form of the input data is the following and the status may be one of the values in the table:

- o 5. byte bi5: 0
- o 6. byte bi6: cooler linking

The status may be one of the following:

symbol	dec. value	hex value
on, linked to Low Power	1	0x01
off, not linked	0	0x00

Refer to User's Manual for more details. Do not enable the cooler unless you know the resulting effects! To use the cooler, it must be installed into your meter.

Get the Cooler Status:
I7GCOOLSTA 97 0x61

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the cooler operational status. The cooler may fail in operation due to insufficient purge air flow. The form of the input data is the following and the status may be one of the values in the table:

- o 5. byte bi5: 0
- o 6. byte bi6: cooler status

The status may be one of the following:

symbol	dec. value	hex value
OK	1	0x01
FAIL,need more purge air	0	0x00

Refer to User's Manual for more details. Do not enable the cooler unless you know the resulting effects! To use the cooler, it must be installed into your meter.

Set the Cooling Enable:
I7SCOOLING 92 0x5C

```
output data bo:
-----------------------------------------------------------------
|cid|com| c |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

o 3. byte bo3: **selection** having values:

symbol	dec. value	hex. value
cooler disabled	0	0x00
cooler enabled	1	0x01

Refer to User's Manual for more details. Do not enable the cooler unless you know the resulting effects! To use the cooler, it must be installed into your meter.

Set the Cooling Linking:
I7SCOOLINK 96 0x60

```
output data bo:
-----------------------------------------------------------------
|cid|com| c |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

o 3. byte bo3: **selection** having values:

symbol	dec. value	hex. value
cooler not linked	0	0x00
cooler linked to Low Power	1	0x01

Refer to User's Manual for more details. Do not enable the cooler unless you know the resulting effects! To use the cooler, it must be installed in your meter.

Get the Web Temperature Offset:
I7GWEBB 103 0x67

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |two|tfo|   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the offset in temperature as a Celsius reading. The form of the input data is the following:

- o 5. byte bi5: offset value whole
- o 6. byte bi6: offset value fractional **(parts of 1/100)**

The actual offset is calculated as follows:

offset value (C) = offset value whole + (offset value fractional / 100)

The offset is used for adjusting the thermometer in case it shows any long-term drift. The correct way to check it is to measure a temperature of an object of 21 - 26 C with a reliable meter and AK50's own web thermometer. The offset is then adjusted to match the reading of the reference meter. Do not do this at any other temperature as it may lead to increased nonlinearity of the signal! There may already be some reading for a unit leaving the factory. That is quite normal.

Set the Offset for Web Temperature:
I7SWEBB 102 0x66

```
output data bo:
-----------------------------------------------------------------
|cid|com| f | f | f | f |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

- o 3. byte bo3: high byte of temperature offset value whole as integer
- o 4. byte bo4: low byte of temperature offset value whole as integer
- o 5. byte bo5: high byte of temperature offset value fraction as integer * 10000
- o 6. byte bo6: low byte of temperature offset value fraction as integer * 10000

With this command one can force the result of the web temperature to change if there is any reason to expect the thermometer's reading to have some sort of long-term drift.

Get the Head Overtemp Status:
I7GALM 104 0x68

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the optical head overheating status. Refer to I7CALM for more details. The form of the input data is the following and the setting may be one of the values in the table:

- o 5. byte bi5: 0
- o 6. byte bi6: alarm status

Status may be one of the following:

symbol	dec. value	hex value
OK, no alarm	0	0x00
overheating of the head	1	0x01

Clear the Head Overheating Alarm:
I7CALM 105 0x69

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

This command will clear the alarm situation which will persist **until** it is specifically cleared. Manual clearing is done in the Keyboard mode by entering and leaving the Menu at least once. Via Profibus DP the only way to clear the alarm is to use this command. Pay attention to the fact that the meter is put into Low Power mode as well. To restart it you have to restore it into Normal operation with the proper command. One should always find out the reasons for overheating to protect your investment and avoid any damage to the instrument. Use air purge and the internal cooler to avoid overheating. If that is not sufficient as your conditions for measurement are so extremely difficult, use some kind of heat shield to protect the meter. Consult Visilab if you need advice.

Calibration and Standardization Commands

Get the Current Material Entry:
I7GETMAT 14 0x0E

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the current material entry number. The form of the input data is the following:

- 5. byte bi5: 0
- 6. byte bi6: entry (1...100) (0x00...0x64)

Switch to another Calibration Table in the Library:
I7SETMAT 15 0x0F

```
output data bo:
-----------------------------------------------------------------
|cid|com| c |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

- o 3. byte bo3: table entry (1...100) (0x01...0x64)

The effect of using this command is switching to using another table in the calibration library. The switching takes actually some time and the resulting moisture values are not reliable for about two seconds. However, due to the nature of the situation of the table switching, this should not be no problem.

Get the Calibration Mode of the Current Material Entry:
I7GMODE 16 0x10

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the mode which is used for the current material entry. The form of the input data is the following:

- o 5. byte bi5: 0
- o 6. byte bi6: mode

Mode may be one of the following:

symbol	dec. value	hex value
QUICK	78	0x4E
MULTI	79	0x4F

Remember that the QUICK mode is the global two-point calibration overriding the table chosen.

Set the Calibration Mode (MULTI/QUICK):
I7SMODE 17 0x11

```
output data bo:
-----------------------------------------------------------------
|cid|com| c |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

o 3. byte bo3: **mode** may have any of the following values:

symbol	dec. value	hex value
QUICK	78	0x4E
MULTI	79	0x4F

Read the Calibration Table Entry:
I7RXMAT 27 0x1B

```
output data bo:
-----------------------------------------------------------------
|cid|com| c | c | c | c |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | c | c | f | f | f | f | f | f | f | f |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

- o 3. byte bo3: entry number of table to be read
- o 4. byte bo4: number of step (1...10) of calibration table element to be returned
- o 5. byte bo5: signal/moisture table selector (0 = signal, 1 = moisture)
- o 6. byte bo6: 0

As a response to this request, the input fields, bytes 5 to 14 are used for returning two table elements of signal or moisture. The elements are either signal(step) and signal(step+1) or moisture(step) and moisture(step+1). The complete table consists always of elements:

step	signal	moisture
1	signal(1)	moisture(1)
2	signal(2)	moisture(2)
3	signal(3)	moisture(3)
4	signal(4)	moisture(4)
5	signal(5)	moisture(5)
6	signal(6)	moisture(6)
7	signal(7)	moisture(7)
8	signal(8)	moisture(8)
9	signal(9)	moisture(9)
10	signal(10)	moisture(10)

The actual number of steps in the table is from 2 to 10. Extra elements in the table have no effect. The signal value varies typically around 1.00. If invalid entry or step values are attached to a command, no actions are taken. The form of the input data is the following:

- o 5. byte bi5: total number of steps in this table
- o 6. byte bi6: calibration mode of this table (refer to command I7GMODE 16)
- o 7. byte bi7: high byte of signal/moisture(step) whole as integer
- o 8. byte bi8: low byte of signal/moisture(step) whole as integer
- o 9. byte bi9: high byte of signal/moisture(step) fraction as integer * 10000
- o 10. byte bi10: low byte of signal/moisture(step) fraction as integer * 10000
- o 11. byte bi11: high byte of signal/moisture(step) whole as integer
- o 12. byte bi12: low byte of signal/moisture(step) whole as integer
- o 13. byte bi13: high byte of signal/moisture(step) fraction as integer * 10000
- o 14. byte bi14: low byte of signal/moisture(step) fraction as integer * 10000

With this command one can read part of the contents of a calibration table entry. One can get an entire table information e.g. for saving by sending a few commands I7RXMAT from the DP Master. Note that the entry's material name can be read with another command I7GMATNM.

Set the Calibration Table Entry:
I7TXMAT 26 0x1A

```
output data bo:
-----------------------------------------------------------------
|cid|com| c | c | c | c | c | f | f | f | f | f | f | f | f |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

- o 3. byte bo3: entry number of table to be set
- o 4. byte bo4: the total number of steps in calibration table (1...10)
- o 5. byte bo5: calibration mode of this table (refer to command I7SMODE 17)
- o 6. byte bo6: number of step in calibration table (1...10) element to be returned
- o 7. byte bo7: signal/moisture table selector (0 = signal, 1 = moisture)
- o 8. byte bo8: high byte of signal/moisture(step) whole as integer
- o 9. byte bo9: low byte of signal/moisture(step) whole as integer
- o 10. byte bo10: high byte of signal/moisture(step) fraction as integer * 10000
- o 11. byte bo11: low byte of signal/moisture(step) fraction as integer * 10000
- o 12. byte bo12: high byte of signal/moisture(step) whole as integer
- o 13. byte bo13: low byte of signal/moisture(step) whole as integer
- o 14. byte bo14: high byte of signal/moisture(step) fraction as integer * 10000
- o 15. byte bo15: low byte of signal/moisture(step) fraction as integer * 10000

As a response to this request, the output fields, bytes 8 to 15 are used for setting two table elements of signal or moisture. The elements are either signal(step) and signal(step+1) or moisture(step) and moisture(step+1). The complete table consists always of elements:

step	signal	moisture
1	signal(1)	moisture(1)
2	signal(2)	moisture(2)
3	signal(3)	moisture(3)
4	signal(4)	moisture(4)
5	signal(5)	moisture(5)
6	signal(6)	moisture(6)
7	signal(7)	moisture(7)
8	signal(8)	moisture(8)
9	signal(9)	moisture(9)
10	signal(10)	moisture(10)

The actual number of steps in the table is from 2 to 10. It is not possible to add extra elements into the table. note that the element(step=11) would be placed on top of element(10). The signal value varies typically around 1.00. If invalid entry or step values are attached to a command, no actions are taken.

With this command one can set part of the contents of a calibration table entry. One can construct an entire table by sending a few commands I7TXMAT from the DP Master. Note that the entry's material name can be set with another command I7SMATNM.

Set the Offset for Standardization:
I7SSHIFT 67 0x43

```
output data bo:
-----------------------------------------------------------------
|cid|com| f | f | f | f |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

- o 3. byte bo3: high byte of moisture offset value whole as integer
- o 4. byte bo4: low byte of moisture offset value whole as integer
- o 5. byte bo5: high byte of moisture offset value fraction as integer * 10000
- o 6. byte bo6: low byte of moisture offset value fraction as integer * 10000

With this command one can force the result of the standardization to change (= "manual" standardization).

Set the Standard Moisture Value for Standardization:
I7SSTD 72 0x48

```
output data bo:
-----------------------------------------------------------------
|cid|com| f | f | f | f |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

- o 3. byte bo3: high byte of moisture standard value whole as integer
- o 4. byte bo4: low byte of moisture standard value whole as integer
- o 5. byte bo5: high byte of moisture standard value fraction as integer * 10000
- o 6. byte bo6: low byte of moisture standard value fraction as integer * 10000

With this command one can set the standard value of the constant moisture sample before performing the standardization. The value is kept in the meter's memory and this is not usually necessary to do more than once. If you change the moisture standard to another, you have to set this variable also.

Get the Material Entry Number Used in Standardization:
I7GSTDM 71 0x47

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the entry number. The form of the input data is the following:

- 5. byte bi5: 0
- 6. byte bi6: table entry (1...100)

Get the Offset Value Resulting from Standardization:
I7GSHIFT 68 0x44

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |mwo|mfo|   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the offset in moisture as a percent reading. The form of the input data is the following:

- o 5. byte bi5: offset value whole
- o 6. byte bi6: offset value fractional (*100)

The actual offset is calculated as follows:

shift value (%) = shift value whole + shift value fractional / 100

Get the Standard Value Set for Standardization:
I7GSTD 73 0x49

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |mws|mfs|   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the actual standard sample reading in moisture in percents. The form of the input data is the following:

- o 5. byte bi5: standard value whole
- o 6. byte bi6: standard value fractional (*100)

The actual offset is calculated as follows:

standard value (%) = standard value whole + standard value fractional / 100

Standardize:
I7STDZE 69 0x45

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

Note that the standardization operation may take about one minute. Refer to User's Manual for details.

Set the Standard Material Entry Number:
I7SSTDM 70 0x46

```
output data bo:
-----------------------------------------------------------------
|cid|com| c |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

- 3. byte bo3: **entry** (1...100) (0x01...0x64)

This entry number is used in the standardization operation.

Data Acquisition Commands

Get the Optical Head Temperature:
I7GHEAD 79 0x4F

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |twh|tfh|   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the head temperature. The form of the input data is the following:

- 5. byte bi5: whole part of temperature
- 6. byte bi6: decimal part of temperature (*100) **(parts of 1/100)**

The actual temperature is calculated as follows:

temperature (C) = temperature whole + temperature fractional / 100

Get the Optional Extra Web Temperature:
I7GWEB2 100 0x64

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |tw2|tf2|   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the optional web temperature if your meter's hardware support it. The form of the input data is the following:

- o 5. byte bi5: whole part of temperature
- o 6. byte bi6: decimal part of temperature (*100) **(parts of 1/100)**

The actual web temperature is calculated as follows:

web temperature (C) = temperature whole + temperature fractional / 100

This is an optional feature in some meters having two web thermometers.

Start Sending the Head Temperature instead of the Web Temperature:
I7GETTMP 46 0x2E

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

This command will turn on the head temperature as the data sent out via polling the temperature field.

Start Sending the Web Temperature Instead of the Head Temperature:
I7GWEB 48 0x30

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

This command will turn on the web temperature as the data sent out via polling the temperature field.

Get the Expansion Module Signal:

I7GXMOD 108 0x6C

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |xw|xf|   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the expansion module signal. The form of the input data is the following:

- o 5. byte bi5: whole part of expansion module signal, xh
- o 6. byte bi6: fractional part of expansion module signal (*100), xf, **(parts of 1/100)**

The actual expansion module signal is calculated as follows:

signal(G) = signal whole + signal fractional / 100

The unit is now marked as G as in generall, the unit is not known. The signal may have negative values as well. The optional module may be a special measuring unit or it may have some other purpose.

Memory Bank Commands

Read the Number of Samples in the Current Bank:
I7GETDM 35 0x23

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | i | i |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the number of samples in the current bank. The form of the input data is the following:

- o 5. byte bi5: high byte of count
- o 6. byte bi6: low byte of count

Read the Bank Number:

I7GBANK 55 0x37

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the identifier of the current bank. The form of the input data is the following and the setting may be one of the values in the table:

- o 5. byte bi5: 0
- o 6. byte bi6: **bank**

bank may be one of the following:

symbol	dec. value	hex. value
Series with 4096 points	0	0x00
Bank1 with 1024 points	1	0x01
Bank2 with 1024 points	2	0x02
Bank3 with 1024 points	3	0x03
Bank4 with 1024 points	4	0x04

Select the Bank:

I7SBANK 54 0x36

```
output data bo:
-----------------------------------------------------------------
|cid|com| c |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

o 3. byte bo3: **bank**

bank may be one of the following:

symbol	dec. value	hex. value
Series with 4096 points	0	0x00
Bank1 with 1024 points	1	0x01
Bank2 with 1024 points	2	0x02
Bank3 with 1024 points	3	0x03
Bank4 with 1024 points	4	0x04

The parameter **bank** causes immediate bank switching to the corresponding bank. Invalid bank references cause no action. Always one of the banks is selected as the current bank.

Get the Autotimer Status:

I7GETAUTO 43 0x2B

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the autotimer status. The form of the input data is the following and the status may be one of the values in the table:

- o 5. byte bi5: 0
- o 6. byte bi6: autotimer status

The status may be one of the following:

symbol	dec. value	hex value
ON	**1**	**0x01**
OFF	**0**	**0x00**

Clear the Current Data Series (or Bank):

I7CLRSER 21 0x15

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

This command will clear the data series in the current memory bank.

Take a Sample into the Current Data Series (or Bank):

I7SAMPLE 36 0x24

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

This command will add the latest moisture reading to the data series in the current bank.

Set the Autotimer ON:

I7AUTOON 41 0x29

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

This command will turn on the autotimer. It will stay on until it is turned off (continuous or Normal operation) or it will be turned off automatically (Batch mode) after fetching the predetermined number of samples. Even in Batch mode the autotimer can be turned off. This will affect the web temperature autotimer as well if they are linked together. Note that the web temperature data is collected to a special memory bank with a rate of 16 samples / s. In earlier SW versions, this rate was one sample / second. The Autotimer is used in the Burst mode too but its control is fully given to the mode as soon as the Autotimer is started. The preset number, Burst count, is acquired at preset time intervals of the Autotimer. Then, the Autotimer is turned off by the mode itself.

Set the Autotimer OFF:

I7AUTOOFF 42 0x2A

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

This command will turn off the autotimer, independent on the mode. This will affect the web temperature autotimer as well if they are linked together. The Autotimer is used in the Burst mode too but its control is fully given to the mode as soon as the Autotimer is started. The preset number, Burst count, is acquired at preset time intervals of the Autotimer. Then, the Autotimer is turned off by the mode itself.

Set the Autotimer Mode:

I7SAMODE 58 0x3A

```
output data bo:
-----------------------------------------------------------------
|cid|com| c |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

o 3. byte bo3: **mode** having values:

symbol	dec. value	hex. value
Normal	1	0x01
Batch	0	0x00

The parameter will change the autotimer mode accordingly. The Burst mode has no use for this mode as it has its own "Batch" count, the Burst count.

Get the Autotimer Mode:

I7GAMODE 59 0x3B

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the autotimer mode. The form of the input data is the following and the setting may be one of the values in the table:

- o 5. byte bi5: 0
- o 6. byte bi6: autotimer mode

The mode may be one of the following:

symbol	dec. value	hex value
Normal	1	0x01
Batch	0	0x00

Get the Autotimer Interval in 0.1ms Units:

I7GETTIM 40 0x28

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | l | l | l | l |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 to 8 are used for returning the autotimer sampling interval. The form of the input data is the following and the setting may be from 0.0025 to 32000 s.

- o 5. byte bi5: MSB of interval
- o 6. byte bi6: HMEDB byte of interval
- o 7. byte bi7: LMEDB byte of interval
- o 8. byte bi8: LSB of interval

The interval is calculated as **0.1ms * (LSB + 256 * LMEDB + 65536 * HMEDB + 16777216 * MSB)**

Set the Autotimer Interval in Seconds:

I7SETTIM 39 0x27

```
output data bo:
-----------------------------------------------------------------
|cid|com| i | i | i | i |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

- o 2. byte bo2: high byte of interval value whole as integer
- o 3. byte bo3: low byte of interval value whole as integer
- o 4. byte bo4: high byte of interval value fraction as integer * 10000
- o 5. byte bo5: low byte of interval value fraction as integer * 10000

The setting may be from 0.0025 to 32000 s in increments of 0.0025 s. Invalid settings are limited in value.

Get the Current Batch Size:

I7GBATCH 57 0x39

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | i | i |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the current bacth size. The form of the input data is the following:

- o 5. byte bi5: high byte of batch count
- o 6. byte bi6: low byte of batch count

Set the Current Batch Size:

I7SBATCH 56 0x38

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   | i | i |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

- o 5. byte bo5: high byte of batch count
- o 6. byte bo6: low byte of batch count

Get Samples from the Current Memory Bank:

I7TXSER 20 0x14

```
output data bo:
-----------------------------------------------------------------
|cid|com| i | i |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |mw1|mf1|mw2|mf2|mw3|mf3|mw4|mf4|   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 to 12 are used for returning four moisture samples. The output fields used are as follows:

- o 2. byte bo2: high byte sample index (1...4096 or 1...1024 depending on the current bank)
- o 3. byte bo3: low byte sample index

The form of the resulting input data is the following:

- o 5. byte bi5: moisture whole, sample index
- o 6. byte bi6: moisture fractional (*100), sample index
- o 7. byte bi7: moisture whole, sample index + 1
- o 8. byte bi8: moisture fractional (*100), sample index + 1
- o 9. byte bi9: moisture whole, sample index + 2
- o 10. byte bi10: moisture fractional (*100), sample index + 2
- o 11. byte bi11: moisture whole, sample index + 3
- o 12. byte bi12: moisture fractional (*100), sample index + 3

The actual moisture is calculated as follows for each of the four samples:

moisture (%) = moisture whole + moisture fractional / 100

Note that the index to be used depends on the bank you are using. Also the number of samples actually collected into it is important. If you have taken only 128 samples but are indexing at sample number 256 to start, the data will be totally improper. Also, for example, if you index to 255 and have taken only 256 samples into the array, ignore samples after #256. If bank boundaries are exceeded, the data may be zero or invalid. Negative indices will be ignored and the earlier data is returned.

Copy the Temperature Series to Bank4:

I7COPYT 98 0x62

```
output data bo:
-----------------------------------------------------------------
|cid|com| c |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

This command copies the data series collected with the temperature autotimer to an independent bank to the globally visible Bank4. The original special memory bank is cleared.

Get the Current Burst Size:

I7GBURST 113 0x71

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | i | i |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the current burst size. The form of the input data is the following:

- o 5. byte bi5: high byte of burst count
- o 6. byte bi6: low byte of burst count

Set the Current Burst Size:

I7SBURST 112 0x70

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   | i | i |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

- o 5. byte bo5: high byte of burst count
- o 6. byte bo6: low byte of burst count

Set the Burst Mode:

I7SBUM 114 0x72

```
output data bo:
-----------------------------------------------------------------
|cid|com| c |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

o 3. byte bo3: **selection** having values:

symbol	dec. value	hex. value
burst mode off	0	0x00
burst mode on	1	0x01

Refer to User's Manual for more details of the burst mode. It overrides the normal Autotimer operation in some ways.

Get the Burst Mode:

I7GBUM 115 0x73

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | 0 | c |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the burst mode setting. The form of the input data is the following and the status may be one of the values in the table:

- o 5. byte bi5: 0
- o 6. byte bi6: burst mode

The status may be one of the following:

symbol	dec. value	hex value
burst mode on	1	0x01
burst mode off	0	0x00

Refer to User's Manual for more details.

Get the Current Burst Mode Item Count:

I7GBUC 116 0x74

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | i | i | i | i |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 to 8 are used for returning the current item count. The form of the input data is the following (long int):

- o 5. byte bi5: MSB byte of item count
- o 6. byte bi6: 3. byte of item count
- o 7. byte bi7: 2. byte of item count
- o 8. byte bi8: LSB byte of item count

Clear the Current Burst Mode Item Count:

I7CBUC 117 0x75

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

This command will clear the item counter used with the burst mode.

Text String Commands

Get the Unit for Moisture:
I7GUNIT 13 0x0D

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | s | s | s | s | s | s | s | s | s | s |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 to 14 are used for returning the unit used in displaying the moisture reading in the meter itself. The form of the input data (a string of characters) is the following:

- o 5. byte bi5: s(0) 1. character of string
- o 6. byte bi6: s(1) 2. character of string
- o 7. byte bi7: s(2) 3. character of string
- o 8. byte bi8: s(3) 4. character of string
- o 9. byte bi9: s(4) 5. character of string
- o 10. byte bi10: s(5) 6. character of string
- o 11. byte bi11: s(6) 7. character of string
- o 12. byte bi12: s(7) 8. character of string
- o 13. byte bi13: s(8) 9. character of string
- o 14. byte bi14: s(9) 10. character of string

- o xx. byte bixx: s(x) end marker character of string. ASCII 0

The end marker may be at any position indicating the actual length of the string. In that case, the rest of the data must be ignored. If the total length of the string is 10 characters, there is no end marker. The unit is constructed easily as a string from this data. Usually, the unit is maximum 6 characters long.

Get the Current Material Entry Name, part 1:

I7GMATNM 31 0x1F

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | s | s | s | s | s | s | s | s | s | s |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 to 14 are used for returning the material name used in the meter for linearization. The name is divided into two parts to fit the space of 10 characters available for transfer. If all byte fields are in use, there is no end marker. The first part consists of the first 10 characters maximum of the name and the second part consists of the rest if any. The form of the input data (a string of characters) is the following:

- o 5. byte bi5: s(0) 1. character of string
- o 6. byte bi6: s(1) 2. character of string
- o 7. byte bi7: s(2) 3. character of string
- o 8. byte bi8: s(3) 4. character of string
- o 9. byte bi9: s(4) 5. character of string
- o 10. byte bi10: s(5) 6. character of string
- o 11. byte bi11: s(6) 7. character of string
- o 12. byte bi12: s(7) 8. character of string
- o 13. byte bi13: s(8) 9. character of string
- o 14. byte bi14: s(9) 10. character of string

- o xx. byte bixx: s(x) end marker character of string. ASCII 0

The end marker may be at any position indicating the actual length of the string. In that case, the rest of the data must be ignored. If the total length of the string transferred is 10 characters, there is no end marker. The name is constructed easily as a string from this data and possibly also from the second part. Usually, the material name is maximum 20 characters long in total.

Get the Current Material Entry Name, part 2:

I7GMATNM2 77 0x4D

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | s | s | s | s | s | s | s | s | s | s |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 to 14 are used for returning the second part of the material name used in the meter for linearization. The name is divided into two parts to fit the space of 10 characters available for transfer. The first part consists of the first 10 characters maximum of the name and the second part consists of the rest if any. If all byte fields are in use there is no end marker. The form of the input data (a string of characters) is the following:

- o 5. byte bi5: s(0) 1. character of string
- o 6. byte bi6: s(1) 2. character of string
- o 7. byte bi7: s(2) 3. character of string
- o 8. byte bi8: s(3) 4. character of string
- o 9. byte bi9: s(4) 5. character of string
- o 10. byte bi10: s(5) 6. character of string
- o 11. byte bi11: s(6) 7. character of string
- o 12. byte bi12: s(7) 8. character of string
- o 13. byte bi13: s(8) 9. character of string
- o 14. byte bi14: s(9) 10. character of string

- o xx. byte bixx: s(x) end marker character of string. ASCII 0

The end marker may be at any position indicating the actual length of the string. In that case, the rest of the data must be ignored. If the total length of the string transferred is 10 characters, there is no end marker. The name is constructed easily as a string from this data and possibly also from the first part. Usually, the material name is maximum 20 characters long in total.

Get the Current Library Name:

I7GLIBNM 29 0x1D

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | s | s | s | s | s | s | s | s |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 to 12 are used for returning the library name stored in the meter. The form of the input data (a string of characters) is the following:

- o 5. byte bi5: s(0) 1. character of string
- o 6. byte bi6: s(1) 2. character of string
- o 7. byte bi7: s(2) 3. character of string
- o 8. byte bi8: s(3) 4. character of string
- o 9. byte bi9: s(4) 5. character of string
- o 10. byte bi10: s(5) 6. character of string
- o 11. byte bi11: s(6) 7. character of string
- o 12. byte bi12: s(7) 8. character of string

- o xx. byte bixx: s(x) end marker character of string. ASCII 0

The end marker may be at any position indicating the actual length of the string. In that case, the rest of the data must be ignored. If the total length of the string transferred is 8 characters, there is no end marker. The library name is constructed easily as a string from this data. Usually, the library name is maximum 8 characters long.

Get the Meter's Identifier String 1:

I7TEST 10 0x0A

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | s | s | s | s | s | s | s | s | s | s |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 to 14 are used for returning the first part of the internal id string stored in the meter. The form of the input data (a string of characters) is the following:

- o 5. byte bi5: s(0) 1. character of string
- o 6. byte bi6: s(1) 2. character of string
- o 7. byte bi7: s(2) 3. character of string
- o 8. byte bi8: s(3) 4. character of string
- o 9. byte bi9: s(4) 5. character of string
- o 10. byte bi10: s(5) 6. character of string
- o 11. byte bi11: s(6) 7. character of string
- o 12. byte bi12: s(7) 8. character of string
- o 13. byte bi13: s(8) 9. character of string
- o 14. byte bi14: s(9) 10. character of string

- o xx. byte bixx: s(x) end marker character of string. ASCII 0

The end marker may be at any position indicating the actual length of the string. In that case, the rest of the data must be ignored. If the total length of the string transferred is 10 characters, there is no end marker. The id string is constructed easily as a string from this data. Usually, the first id string describes the meter model.

Get the Meter's Identifier String 2:

I7TEST2 78 0x4E

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | s | s | s | s | s | s | s | s | s | s |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 to 14 are used for returning the second part of the internal id string stored in the meter. The form of the input data (a string of characters) is the following:

- o 5. byte bi5: s(0) 1. character of string
- o 6. byte bi6: s(1) 2. character of string
- o 7. byte bi7: s(2) 3. character of string
- o 8. byte bi8: s(3) 4. character of string
- o 9. byte bi9: s(4) 5. character of string
- o 10. byte bi10: s(5) 6. character of string
- o 11. byte bi11: s(6) 7. character of string
- o 12. byte bi12: s(7) 8. character of string
- o 13. byte bi13: s(8) 9. character of string
- o 14. byte bi14: s(9) 10. character of string

- o xx. byte bixx: s(x) end marker character of string. ASCII 0

The end marker may be at any position indicating the actual length of the string. In that case, the rest of the data must be ignored. If the total length of the string transferred is 10 characters, there is no end marker. The id string is constructed easily as a string from this data. Usually, the second id string describes the meter's embedded software version.

Get the Meter's Identifier String 3:

I7TEST3 80 0x50

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | s | s | s | s | s | s | s | s | s | s |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 to 14 are used for returning the third part of the internal id string stored in the meter. The form of the input data (a string of characters) is the following:

- o 5. byte bi5: s(0) 1. character of string
- o 6. byte bi6: s(1) 2. character of string
- o 7. byte bi7: s(2) 3. character of string
- o 8. byte bi8: s(3) 4. character of string
- o 9. byte bi9: s(4) 5. character of string
- o 10. byte bi10: s(5) 6. character of string
- o 11. byte bi11: s(6) 7. character of string
- o 12. byte bi12: s(7) 8. character of string
- o 13. byte bi13: s(8) 9. character of string
- o 14. byte bi14: s(9) 10. character of string

- o xx. byte bixx: s(x) end marker character of string. ASCII 0

The end marker may be at any position indicating the actual length of the string. In that case, the rest of the data must be ignored. If the total length of the string transferred is 10 characters, there is no end marker. The id string is constructed easily as a string from this data. Usually, the third id string describes the meter's serial number.

Get the Expansion Module Name:

I7GXNAME 110 0x6E

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | s | s | s | s | s | s | s | s |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 to 12 are used for returning the expansion module name stored in the meter. The form of the input data (a string of characters) is the following:

- o 5. byte bi5: s(0) 1. character of string
- o 6. byte bi6: s(1) 2. character of string
- o 7. byte bi7: s(2) 3. character of string
- o 8. byte bi8: s(3) 4. character of string
- o 9. byte bi9: s(4) 5. character of string
- o 10. byte bi10: s(5) 6. character of string
- o 11. byte bi11: s(6) 7. character of string
- o 12. byte bi12: s(7) 8. character of string

- o xx. byte bixx: s(x) end marker character of string. ASCII 0

The end marker may be at any position indicating the actual length of the string. In that case, the rest of the data must be ignored. If the total length of the string transferred is 8 characters, there is no end marker. The name is constructed easily as a string from this data. The module name is maximum 8 characters long. The name is factory set when the module is installed to the meter. It's existence is shown in status3 also. The name is a short description of what the module does. It may be a special measuring unit or it may have some other purpose.

Set the Current Library Name:

I7SLIBNM 30 0x1E

```
output data bo:
-----------------------------------------------------------------
|cid|com| 0 | 0 | s | s | s | s | s | s | s | s |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The meter's library name is changed with this command. The output fields used are as follows:

- o 3. byte bo3: 0
- o 4. byte bo4: 0
- o 5. byte bo5: s(0) 1. character of string
- o 6. byte bo6: s(1) 2. character of string
- o 7. byte bo7: s(2) 3. character of string
- o 8. byte bo8: s(3) 4. character of string
- o 9. byte bo9: s(4) 5. character of string
- o 10. byte bo10: s(5) 6. character of string
- o 11. byte bo11: s(6) 7. character of string
- o 12. byte bo12: s(7) 8. character of string
- o 13. byte bo13: s(8) end marker character of string. ASCII 0

- o xx. byte boxx: s(x) end marker character of string. ASCII 0

The end marker may be at any position indicating the actual length of the string. In that case, the rest of the data will be ignored. You can fill in the rest of the output bytes with zero. The library name is maximum 8 characters long.

Set the Unit for Moisture:

I7SUNIT 12 0x0C

```
output data bo:
-----------------------------------------------------------------
|cid|com| 0 | 0 | s | s | s | s | s | s | s |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The measuring unit is changed with this command. The output fields used are as follows:

- o 3. byte bo3: 0
- o 4. byte bo4: 0
- o 5. byte bo5: s(0) 1. character of string
- o 6. byte bo6: s(1) 2. character of string
- o 7. byte bo7: s(2) 3. character of string
- o 8. byte bo8: s(3) 4. character of string
- o 9. byte bo9: s(4) 5. character of string
- o 10. byte bo10: s(5) 6. character of string
- o 11. byte bo11: s(6) end marker character of string. ASCII 0

- o xx. byte boxx: s(x) end marker character of string. ASCII 0

The end marker may be at any position indicating the actual length of the string. In that case, the rest of the data will be ignored. The unit is maximum 6 characters long. Note that the unit does not affect the resulting moisture value in any way. It is displayed in Keyboard mode only in the PC program's screen. It works as a reminder in special cases.

Set a Material Name, part 1:

I7SMATNM 81 0x51

```
output data bo:
-----------------------------------------------------------------
|cid|com| 0 | 0 | s | s | s | s | s | s | s | s | s |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The name of a material entry is changed with this command. Only the first 10 characters of the name are transferred. The output fields used are as follows:

- o 3. byte bo3: entry # (1...100)
- o 4. byte bo4: 0
- o 5. byte bo5: s(0) 1. character of string
- o 6. byte bo6: s(1) 2. character of string
- o 7. byte bo7: s(2) 3. character of string
- o 8. byte bo8: s(3) 4. character of string
- o 9. byte bo9: s(4) 5. character of string
- o 10. byte bo10: s(5) 6. character of string
- o 11. byte bo11: s(6) 7. character of string
- o 12. byte bo12: s(7) 8. character of string
- o 13. byte bo13: s(8) 9. character of string
- o 14. byte bo14: s(9) 10. character of string

- o xx. byte boxx: s(x) end marker character of string. ASCII 0

The end marker may be at any position indicating the actual length of the string. In that case, the rest of the data will be ignored. The material name (part 1) is maximum 10 characters long. If the 10 characters are all used, no end marker is required. The second part is sent with another command and the strings are patched automatically. Note that this command goes as the first command, else an incorrect patching will be done.

Set a Material Name, Part 2:

I7SMATNM2 82 0x52

```
output data bo:
-----------------------------------------------------------------
|cid|com| 0 | 0 | s | s | s | s | s | s | s | s | s |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The second part of a name of a material is changed with this command. The characters after the first 10 characters in the name are transferred, if any (max. 20 characters total length). If the 10 characters places are all used, no end marker is required. The output fields used are as follows:

- o 3. byte bo3: entry # (1...100)
- o 4. byte bo4: 0
- o 5. byte bo5: s(0) 1. character of string
- o 6. byte bo6: s(1) 2. character of string
- o 7. byte bo7: s(2) 3. character of string
- o 8. byte bo8: s(3) 4. character of string
- o 9. byte bo9: s(4) 5. character of string
- o 10. byte bo10: s(5) 6. character of string
- o 11. byte bo11: s(6) 7. character of string
- o 12. byte bo12: s(7) 8. character of string
- o 13. byte bo13: s(8) 9. character of string
- o 14. byte bo14: s(9) 10. character of string

- o xx. byte boxx: s(x) end marker character of string. ASCII 0

The end marker may be at any position indicating the actual length of the string. In that case, the rest of the data will be ignored. The material name (part 2) is maximum 10 characters long. The first part is sent with another command and the strings are patched automatically. Note that this command goes only after the first command, else an incorrect patching will be done.

Special Commands

Get the LAN Addresses:
I7GLAN 84 0x54

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | i | i | i | i |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 to 8 are used for returning the master and slave addresses used in the RS485-based Local Area Network. The form of the input data is the following:

- o 5. byte bi5: 0
- o 6. byte bi6: master address (0...255)
- o 7. byte bi7: 0
- o 8. byte bi8: slave address (0...255)

With this command you can check the addresses if troubleshooting is required for your second network.

Set the LAN Addresses:

I7SLAN 85 0x55

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   | i | i | i | i |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

- o 5. byte bo5: 0
- o 6. byte bo6: master address (0...255)
- o 7. byte bo7: 0
- o 8. byte bo8: slave address (0...255)

With this command you can set the addresses for your second network. The default addresses are 0 for Master and 1 for Slave.

Initialize the Profibus DP slave:

I7DPINIT 65 0x41

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

This will initialize the slave. Note that it will also be incapable of responding to DP master while initializing all SW and HW subsystems (< 1 second).

Send a Short Pulse to the LED Indicator (if available on the connector panel):

I7BEEP 34 0x22

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

This command will flash once the indicator lamp on the back panel of the meter.

Start Fast Fourier Transform in the Meter:

I7FFT **83** **0x53**

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The meter's own FFT routine is started with this command. It is a fixed 1024 samples' transform and it uses the contents of Bank1. While performing it, it needs the space of the other banks, Bank2, Bank3, and Bank4. The long Series is not touched during that process. The conversion will take usually less than ten seconds and the resulting power spectrum will be placed into Bank1. The spectrum is not flattened with logarithm, it is linear. One can use it if the Advanced PC program is not available or there are no frequency domain algorithms available in your **Profibus DP** development tool. The data from Bank1 can then be downloaded to your DP Master for further analysis and display.

Get the Expansion Module Number:

I7GNXMOD 109 0x6D

```
output data bo:
-----------------------------------------------------------------
|cid|com|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf | i | i |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

As a response to this request, the input fields, bytes 5 and 6 are used for returning the number of the expansion module connected to this meter. The form of the input data is the following:

- o 5. byte bi5: high byte of count = 0
- o 6. byte bi6: low byte of count = unsigned character 0...255 module number

The number is factory set when the module is installed to the meter. It's existence is shown in status3 also. The number is a condensed description of the type of the module and gives the acquisition system instructions on how to handle the associated data.The module may be a special measuring unit or it may have some other purpose.

Send a Command to the Expansion Module:

I7SXCOM 111 0x6F

```
output data bo:
-----------------------------------------------------------------
|cid|com| c |   |   |   |   |   |   |   |   |   |   |   |   |   |
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
input data bi:
-----------------------------------------------------------------
|mw |mf |tw |tf |   |   |   |   |   |   |   |   |   |   |cid|sta|
-----------------------------------------------------------------
  1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16    byte
```

The output fields used are as follows:

o 3. byte bo3: having values:

symbol	dec. value	hex. value
command	0...255	0x00...0xFF

The **command** will be directly transferred to the expansion module without any filtering, interpretation or waiting for a reply. There is only a simple protocol associated with this, no actual error checking. The module may be a special measuring unit or it may have some other purpose. If error checking or command accepting response is required by the expansion module, it will respond by the signal it transmits. That is interpreted at the highest level to make sure the command was accepted without errors. Refer to the expansion module's own User's Guide for details about its commands, if any are available. If no expansion module is installed to **AK50**, the commands have no effect.

Appendix 1. A Sample Database Text File for DP Slave Configuration

This data file is included on your program diskette as IRMA7416.TXT and as a binary loadable database IRMA7416.LDB.

```
@DB_TYP  CP5412A2  V1.0
@LANGUAGE  ENGLISH
#NCMCV_CONVERTER  V1.0  ASCII-OUTPUT
##DB_Ident_ID        = 0                                              # constant
##versionStr         = "@@DB_TYP CP5412A2 V1.0 DEUTSCH DOS"           # constant
# creator            = "UNKNOWN"                                      # default
##SystemSDBBlockId   = 121                                            # constant

  SystemSDB
  {

        SDBParameterblock
        {
            ##PblkNo1            = 1
# constant
            ##PblkVersion1       = 0
# constant
            # LocalEdit          = "TRUE"
# default
              MaxMaster          = 1
              BusParCalculate    = "dpmono"
            # ProfileType        = 0
# default
              MaxSlave           = 2
            ##PblkNo2            = 2
# constant
            ##PblkVersion2       = 0
# constant
              Hsa                = 126
              Ts                 = 2
            # StationType        = "ACTIVE"
# default
              BaudRate           = "1500.0"
            ##MediumRed          = 0
# constant
              RetryCtr           = 1
              DefaultSap         = 52
              NetwkConSap        = 53
              Tsl                = 300
              Tqui               = 0
              Tset               = 1
              MinTsdr            = 11
              MaxTsdr            = 150
              Ttr                = 1000000
              G                  = 10
            ##InRingDesired      = 1
```

```
# constant
            ##PhysLayer          = 0
# constant
            ##SblkNo5            = 5
# constant
            ##SblkVersion5       = 0
# constant
        }
  }
##DpSdbBlockId       = 128                                          # constant

  DpSDB
  {

        SDBParameterblock
        {
            ##PblkDpMgt          = 1
# constant
            ##PblkVersion        = 0
# constant
            ##DpManagementLEN    = 37
# constant
            # CtrPollToResetStatLst = 1
# default
              DataControlTime    = 60
              MasterPollTimeout  = 2
            # Master2PollTimeout = 200
# default
              MasterGsdFile      = “CP3425DP.GSD”
              reservedWdTimeout  = 10
              ClearDP            = “FALSE”
              zyklGlobalCtrlCommand = “NONE”
              bWdDefault         = “TRUE”
            ##ReservedDp          = 0b0000
# constant
            # zyklGlobalCtrlGrpIdent = 0
# default
              maxMinSlvIntervall = 5
              DpDelayTime        = 5

              NormSlave
              {
                  ##PblkDpMgt          = 2
# constant
                  ##PblkVersion        = 0
# constant
                    SlvAdr             = 3
                  ##NormSlaveKey       = 0
# constant
                    SlvName            = “AK50 Moisture meter”
                    GsdFile            = “IRMA7D04.GSD”
```

```
                  MinSlvIntervall    = 50
                # SlvActivate        = “Activated”
# default
                # ActivatedSlvMode   = “SlvDataExchange”
# default
                # DeActivatedSlvMode = “SlvDoNothing”
# default
                # Passiv             = “Active”
# default
                ##Reserved          = 0b00
# constant
                ##Reserved1         = 0b000
# constant
                  Watchdog          = “On”
                  FreezeMode        = “Off”
                  SyncMode          = “Off”
                ##MstAccess         = 0b01
# constant
                  WdFact1           = 1
                  WdFact2           = 10
                  MinTsdr           = 11
                  IdentNumber       = 0x08
                  GrpIdent          = 0x00
                  OutLenAll         = 16
                  InLenAll          = 16

                  ModEntry
                  {
                        ModType           = “Moisture/temp./data”
                        OutBegin          = 0
                        InBegin           = 0
                  }

                  ModEntry
                  {
                        ModType           = “Moisture/temp./data”
                        OutBegin          = 0
                        InBegin           = 0
                  }

                  Module
                  {

                        SimpleModDescr
                        {
                              IoLen             = 0b1111
                              IoType            = “Input”
                              Alignment         = “Byte”
                              Consistency       = “Value”
                        }
                  }
```

```
                    Module
                    {

                          SimpleModDescr
                          {
                                IoLen              = 0b1111
                                IoType             = “Output”
                                Alignment          = “Byte”
                                Consistency        = “Value”
                          }
                    }
              }
        }
  }
```

Appendix 2. A Sample GSD Data File for DP Slave Configuration

This GSD data file is included on your program diskette as IRMA7D04.GSD. Note that the earlier GSD file is for slaves with ID = 0. Now the ID = 8 until a final ID is received. To adapt to the new number requires only changing one field in the file. The latest internal software allows also the changing of the unit's own ID.

```
;
;  Description of a DP slave. Lines beginning with a semicolon are
;  comments only.
;=============================================================
; GSD-Data for AK50                    Visilab Signal Technologies Oy
;                                  Vantaa, Finland
; Vers. : V1.02  -        Mr. H. Stenlund Tel. +358-45-6354885
; AK50 is a fast on-line surface moisture meter for
; measuring paper, board and other substances.
; It supports all rates from 9600 bauds to 12 Mbauds.
; The RTS signal is available at TTL level. It also has an
; isolated +5V pin on the D9 connector supplying some 50mA max.
; The address can be changed by the master.
; Rev. History:  V1.01 The Ident-Number was changed to 0x00008
;                V1.02 the IO structure was changed to 16 bytes in/out with
;                 full compatibility with old systems. Use this
;                 GSD file for the 16/16 bytes and the old irma7d03.gsd for the
;                 4/4 bytes IO. All old commands are supported with no changes.
;=============================================================
; General parameters
#Profibus_DP
Vendor_Name = "Visilab Signal Technologies"
Model_Name = "AK50 Moisture meter - Rev5K"
Revision = "Version 2"
Ident_Number = 0x0008
Protocol_Ident = 0
; 0=Slave, 1=Master (Class 1)
Station_Type = 0
; 0=DP only, 1=DP and FMS
FMS_supp = 0
Hardware_Release = "A01"
Software_Release = "Z02"
9.6_supp = 1
```

```
19.2_supp = 1
93.75_supp = 1
187.5_supp = 1
500_supp = 1
1.5M_supp = 1
3M_supp = 1
6M_supp = 1
12M_supp = 1
MaxTsdr_9.6 = 60
MaxTsdr_19.2 = 60
MaxTsdr_93.75 = 60
MaxTsdr_187.5 = 60
MaxTsdr_500 = 100
MaxTsdr_1.5M = 150
MaxTsdr_3M = 250
MaxTsdr_6M = 450
MaxTsdr_12M = 800
Redundancy = 0
; RTS-Signal (CNTR-P) Pin 4  9-pin SUB-D
; 0-not available,  1-RS485, 2-TTL
Repeater_Ctrl_Sig = 2
;  24V Pin in 9-pin. SUB-D
; 0-none, 1-RS485, 2-TTL
24V_Pins = 0
Implementation_Type="SPC3"
Bitmap_Device="DP_NORM"
orderNumber="AK50"
;
;--Slave specific values-----
;
; Freeze Mode
Freeze_Mode_supp = 0
; Sync Mode
Sync_Mode_supp = 0
; Autom. Baud rate search
Auto_Baud_supp = 1
; Address is changeable
Set_Slave_Add_supp = 1
;unsigned 8
User_Prm_Data_Len = 0x0
User_Prm_Data =
; Slave  unsigned 16  (Basis 100us),  5 ms
Min_Slave_Intervall = 0x0032
```

Modular_Station = 1
Max_Module = 0x01
Max_Input_Len = 0x20
Max_Output_Len = 0x20
Max_Data_Len = 0x0040
; Module, 16 bytes in, 16 bytes out
Module= "Moisture/temperature/data in, command/data out" 0x1F,0x2F
EndModule

Appendix 3. Schematic of Electrical Connections for Model D in the connection box. **If you open the distribution boxes cover, please, disconnect the power lead first to avoid a shock!** Find below a schematic for wiring of the circuit board inside the power supply -PS. On the next page, find a corresponding component placement drawing. You can find more information from the model -PS Operating guide.

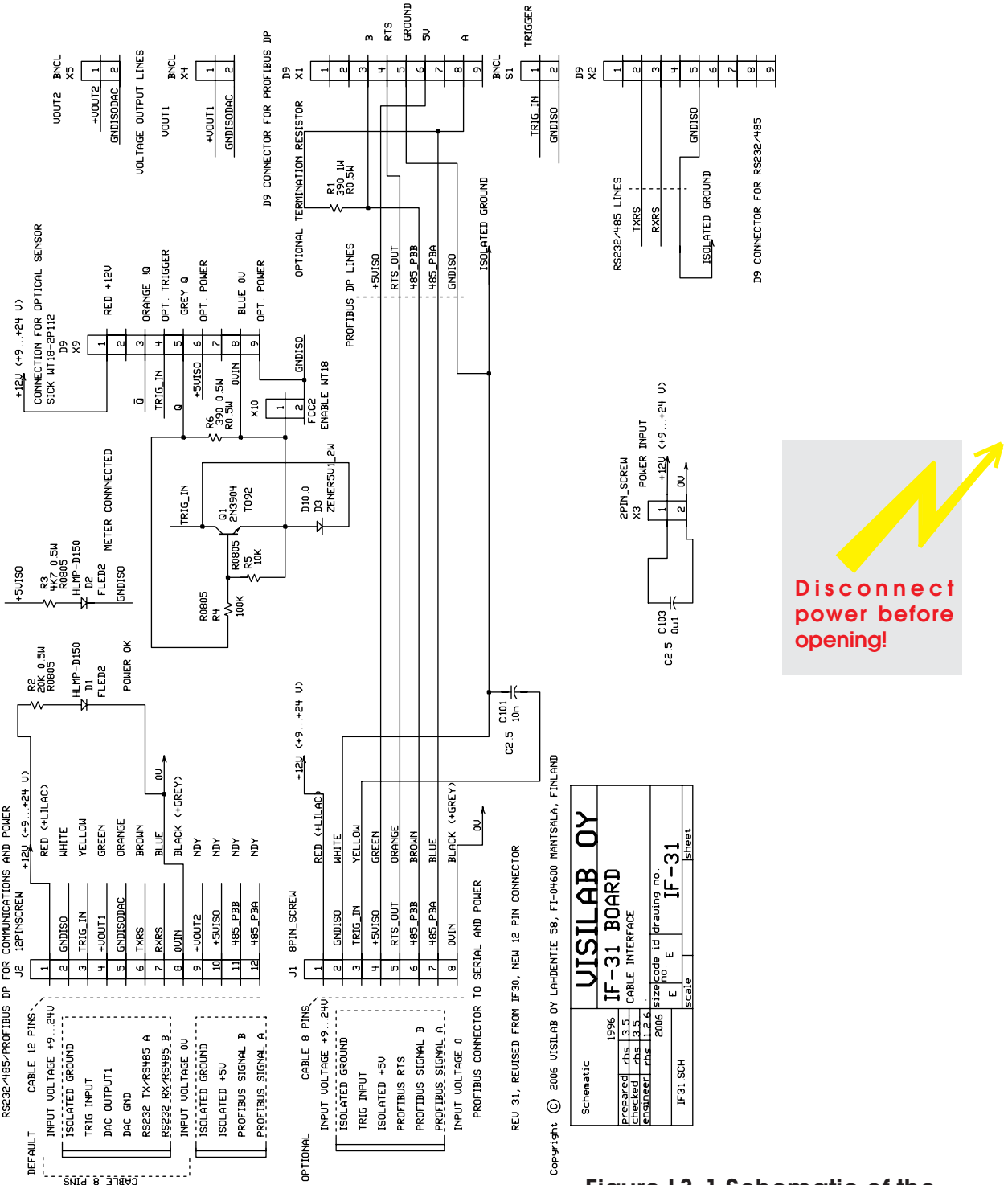

Figure L3-1 Schematic of the circuit board in the model -PS power supply unit

Appendix 4. Procedure for Passing Commands to the Slave

The procedure for sending any of these commands is the following. The default command that should normally be sent to a slave, consists of zeros only. That will make it sure that no pending commands are processed. Note that it is required to send the command **only once** to a slave. Sending a command repeatedly will redo the same thing and possibly overload the slave. For simplicity, in the following the command identifier **cid** is left as zero.

For that reason you have to go along the following lines to communicate with the slave, independent of the type of the master and the user interface attached to it.

step	command bytes bo1	bo2	bo3	bo4 and rest of bytes	action
1.	0	0	0	0	default, no action taken
2.	0	0	data	data	no action taken
3.	**0**	**command**	**data**	**data**	**command performed**
4.	0	0	data	data	no action taken
5.	0	0	0	0	default, no action taken

In this way you will prevent sending incorrectly interpreted or duplicated commands. Zero commands are simply ignored. The problematic commands are:

	bo1	bo2	bo3 and rest of bytes	
n.	0	command	undefined data	
and:				
n+1.	0	command	real data	
n+2.	0	command	real data	
n+3.	0	command	real data	
n+4.	0	command	real data	(an accidentally repeated command)

In the first case (step n.) the data part is not yet defined and this may lead to unexpected results, like to changing to work with an empty calibration table. The next command (step n+1.) is repeated infinitely until replaced with another. That will cause possibly overloading of the slave and decreases the response times.

If you are sure that your master is able to send the command output data as a single operation, like step 3. above, then you can apply the commands as such without the intermediate steps 2. and 4.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W