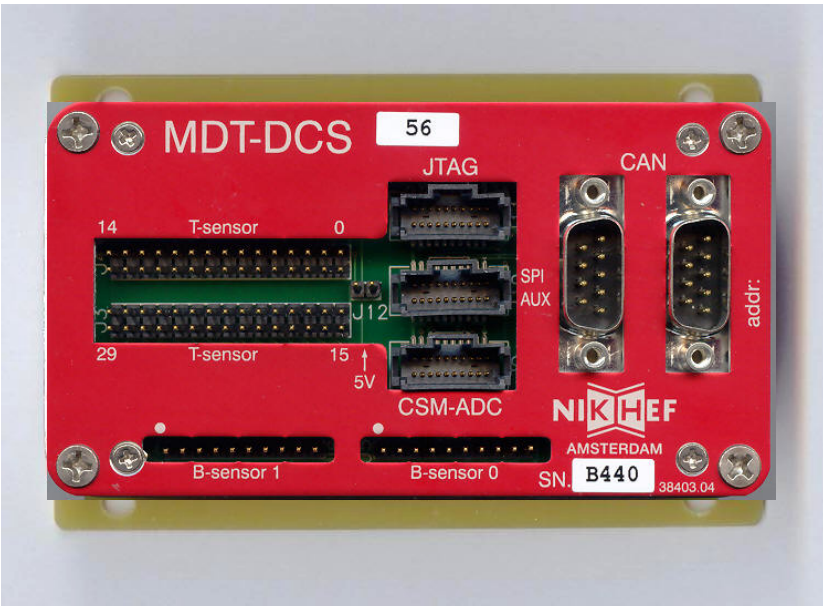
MDT-DCS
CANopen Module



user manual & reference
v2.6, 8 March 2008

Henk Boterenbrood
NIKHEF, Amsterdam, NL

ABSTRACT

Each ATLAS MDT muon chamber is equipped with an MDT-DCS module. The module has been designed to monitor the chamber's environmental parameters, i.e. temperature (NTC sensors), magnetic field (B-field sensors) and front-end electronics parameters (CSM and Mezzanine Board voltages and temperatures), as well as to initialize and configure the MDT chamber's front-end electronics on the CSM and Mezzanines Boards. The heart of the module is the general-purpose ELMB plug-on microcontroller board with CAN interface for communication. This document gives a detailed description of the MDT-DCS module and its ELMB application firmware including the CANopen communication protocol and CANopen Object Dictionary.

Table of Contents

Version History		
Version	Date	Comments
2.6	8 Mar 2008	• Some minor modifications to the text.
2.5	21 Aug 2007	• Describes firmware version "MD24", minor version "0001". • PDOs for JTAG can have any number of bytes between 2 and 8 (instead of previously exactly 5). • Added Objects 4831h and 4832h, for setting the JTAG state after a SHIFT_IR or SHIFT_DR operation.
2.4	22 Jun 2006	• Describes firmware version "MD24", minor version "0000". • Added info on CSM+Mezz temperature sensors. • Created separate sections for T and B data conversion. • MDT-DCS module schematics added. • Various minor changes, corrections and additions. • Added Objects 480Ch, 480Dh, 49XCh, 49XDh and 49XEh.
2.3	3 Aug 2005	• Describes firmware version "MD24", minor version "0000". • Added description and objects for JTAG string uploading and downloading by means of the Segmented SDO protocol. • Added description of 'autoconfigure' capability. • Added pictures of production acceptance test setup. • Corrected wrong values for the reference T-sensors 30 and 31. • Various minor changes to text.
2.2	27 May 2004	• Describes firmware version "MD23". • JTAG action storage, actions and sequences implemented. • 'Readout-on-delta-change' feature added for T-sensors and CSM analog inputs. • Support for remotely configurable Node-ID added. • Presence of B-sensor modules now controlled by a mask. • PDO event timers now in seconds (instead of ms) and active for all transmission types. • Digital Out power-up setting now on individual bit basis. • Another update of the section on board testing. • Up-to-date pictures of frontpanels ('Barrel' as well as 'EndCap').
2.1	16 Jan 2004	• Describes firmware version "MD22". • Support for up to 4 B-sensor modules, instead of 2, as before. • Support for 'raw' T-sensor data readout. • JTAG TDI and TDO signals swapped. • Does *not* yet support JTAG-action strings storage. • Update of the section on board testing.
2.0	17 Nov 2003	• Describes firmware version "MD21". • Does *not* yet support JTAG-action strings storage. • Added this document change record.
1.0	30 May 2002	Describes firmware version "MD14" (and older), for the MDT-DCS module prototype equipped with ELMB103 modules.

Table 1. Document change record.

1 Introduction and Overview

The ***MDT-DCS module*** is the local monitor-and-control platform for the ATLAS **MDT** muon chambers. It is based on the **ELMB**[1] module, which is a general-purpose plug-on board which was developed by the ATLAS collaboration to serve various detector control tasks in and around the ATLAS detector. The ELMB is also used in several applications outside the ATLAS environment, in other LHC experiments and at CERN. The ELMB features an in-system-programmable microcontroller, a **CAN**-bus controller and interface for communication with a host system and/or the central SCADA system, and a number of analog inputs and digital in- and outputs. It is in-system-programmable, including remotely, via the CAN-bus. The latter combined with the ELMB's low cost and the availability of a low cost development environment for programming in C, have all contributed to its success. For its application in ATLAS the fact that its radiation tolerance/sensitivity has been extensively tested and quantified is very important.

In the MDT muon subdetector of ATLAS the MDT-DCS module monitors MDT chamber environmental parameters, i.e. temperature and magnetic field in and around the chamber, and MDT front-end electronics voltages and temperatures. The MDT front-end electronics consist of the so-called **CSM** (*Chamber Service Module*) plus connected **Mezzanine Boards**.
The MDT-DCS module has a JTAG interface, that connects to the CSM, for configuration of the MDT front-end electronics. In addition there are a number (7) of Digital I/Os for control output to the CSM and error status input from the CSM.

The ***CAN*** bus is the chosen fieldbus by the ATLAS Detector Control System (**DCS**) for interconnecting distributed I/O within the detector. The ***CANopen*** protocol [2] [3] has been adopted as the communication protocol standard to be used on the CAN-bus.

The application firmware running on the ELMB inside the MDT-DCS module complies where possible with the *CANopen* DS-401 Device Profile for I/O-modules [4], but it has a range of additional 'manufacturer-specific' *Object Dictionary* entries and configuration options. The complete *Object Dictionary* (*OD*) of the MDT-DCS *CANopen* node can be found in section 9.
The MDT-DCS firmware development is based on a framework provided by the so-called ***ELMBio*** application firmware described in [1].

A simplified block diagram of an MDT-DCS module mounted on an MDT muon chamber, and its connections to sensors and front-end electronics is shown in Figure 1. There are connections to the on-chamber T- (temperature) and B-sensors (magnetic field), and (multiple) connections to the MDT front-end electronics.

The T-sensors are *NTC* resistors (for Barrel MDT chambers) integrated in special cables and mounted on various locations on the MDT chamber.

Each B-sensor module measures the magnetic field along 3 orthogonal axes (B_x, B_y and B_z) and the temperature *(T)* of the environment in the immediate vicinity of the 3 Hall-effect transducers, which are mounted on the B-sensor module PCB. By using special cables the number of connected B-sensor modules may be increased from 2 to 4 per MDT-DCS module.

[1] see http://elmb.web.cern.ch/

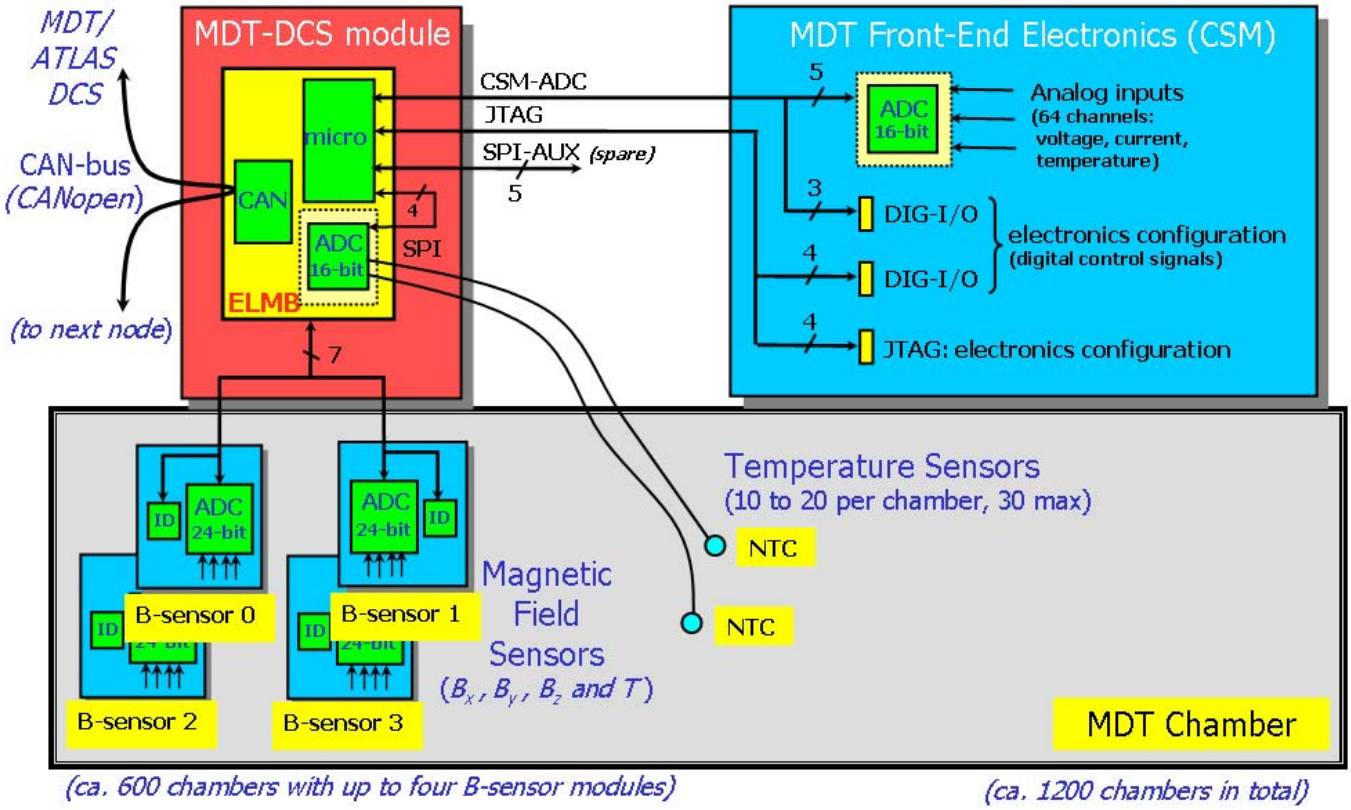
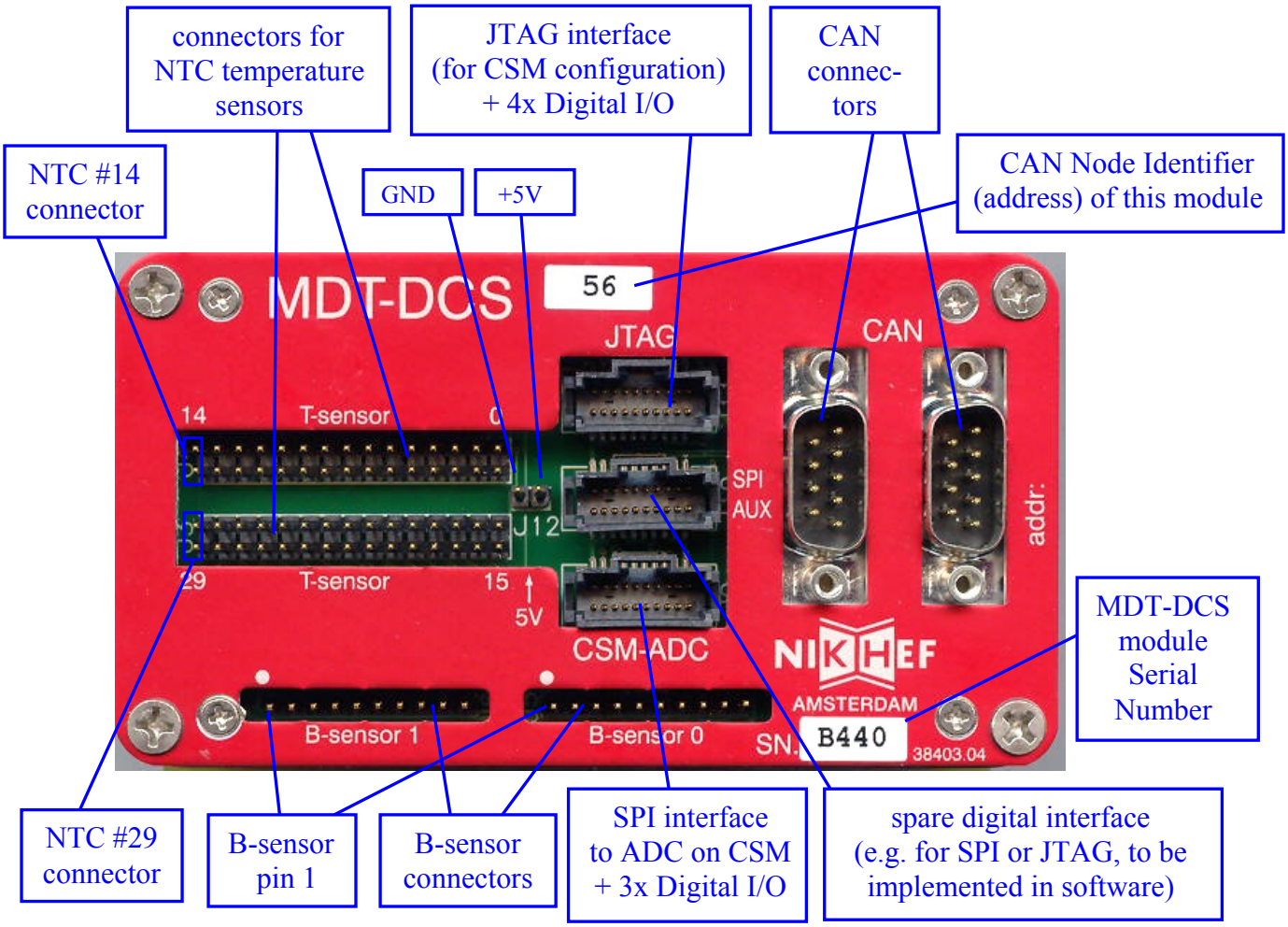


Figure 1. Block diagram of the MDT-DCS module with ELMB, its connections to the MDT front-end electronics and the external sensors (for temperature and B-field), mounted on an MDT chamber. Although originally intended to support two Magnetic Field Sensor modules, this number may be increased to four per MDT-DCS module using special cables (see section 5.3).

2 Hardware

2.1 Connectors and Interfaces

Figure 2 shows the front panel of the MDT-DCS module with its external interfaces. There are 2 types of MDT-DCS modules, a 'Barrel' type and an 'Endcap' type, which visibly only differ by their labels as shown in Figure 2. The 'Endcap' type is equipped for voltage-based T-sensors (such as PT1000), and the 'Barrel' type for resistance-type T-sensors (NTC); the firmware has been preconfigured accordingly.



Label with ATLAS number/barcode on the side of the MDT-DCS box

MDT-DCS module 'EndCap' type indicated by:

- a yellow label
- serial number is followed by an 'e' (on label only)

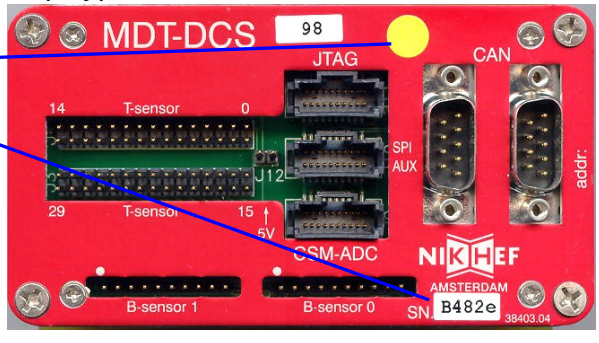
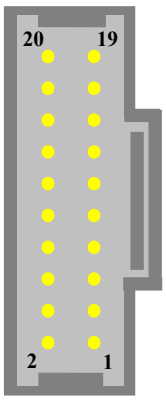


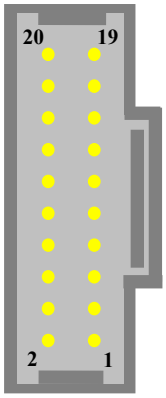
Figure 2. MDT-DCS module front panel connectors and labels.

Table 2 to Table 6 detail the pin layout of the MDT-DCS module's *JTAG*, *SPI-AUX*, *CSM-ADC* and *CAN* frontpanel connectors.



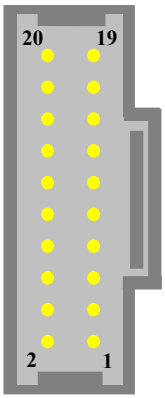
function	pin	pin	function	comment
GND	**20**	**19**	**GND**	
GND	**18**	**17**	Dig I/O 4 (PA7)	*Reprogram_FPGA**
GND	**16**	**15**	Dig I/O 3 (PA6)	*Reset_FPGA*
+3.3V	**14**	**13**	Dig I/O 2 (PA5)	*Sel_SW_TDO**
+3.3V	**12**	**11**	Dig I/O 1 (PA4)	*Sel_HW_TDO**
+3.3V	**10**	**9**	TDI (PA3, in)	JTAG interface
+3.3V	**8**	**7**	TMS (PA2, out)	JTAG interface
GND	**6**	**5**	TCK (PA1, out)	JTAG interface
GND	**4**	**3**	TDO (PA0, out)	JTAG interface
GND	**2**	**1**	**GND**	

Table 2. Layout of the *JTAG* connector pins: 8 general-purpose digital in- and outputs. In brackets the ELMB microcontroller pin name is shown, in italics the CSM's name for the signal function.



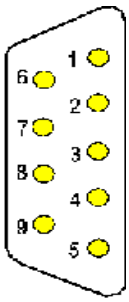
function	pin	pin	function
GND	**20**	**19**	**GND**
GND	**18**	**17**	*not connected*
GND	**16**	**15**	*not connected*
+3.3V	**14**	**13**	*not connected*
+3.3V	**12**	**11**	Aux I/O (PC5)
+3.3V	**10**	**9**	Aux I/O (PC4)
+3.3V	**8**	**7**	Aux I/O (PF6)
GND	**6**	**5**	Aux I/O (PC6)
GND	**4**	**3**	Aux I/O (PC7)
GND	**2**	**1**	**GND**

Table 3. Layout of the *SPI-AUX* connector pins: 5 general-purpose Digital I/Os, sufficient and suitable for implementing a serial interface like SPI, I^2C or JTAG for instance (to be implemented in the MDT-DCS/ELMB firmware). In brackets the ELMB microcontroller pin name is shown.



function	pin	pin	function	comment
GND	**20**	**19**	**GND**	
GND	**18**	**17**	Dig I/O 7 (PF4)	*GOL | TTC Not Ready*
GND	**16**	**15**	Dig I/O 6 (PF3)	*I2C Error*
+3.3V	**14**	**13**	Dig I/O 5 (PF2)	*CSM Error*
+3.3V	**12**	**11**	MUX (PE7, out)	for ADC
+3.3V	**10**	**9**	CS (PC3, out)	for ADC
+3.3V	**8**	**7**	SDO (PE6, in)	for ADC
GND	**6**	**5**	SDI (PE5, out)	for ADC
GND	**4**	**3**	SCLK (PE4, out)	for ADC
GND	**2**	**1**	**GND**	

Table 4. Layout of the *CSM-ADC* connector pins: SPI serial interface (SCLK, SDI and SDO) with Chip-Select (CS) and ADC-multiplexer latch signal (MUX) go to the ADC on the CSM (which has a copy of the ELMB's on-board ADC circuitry). In addition there are 3 general-purpose Digital I/Os. In brackets the ELMB/microcontroller pin name is shown, in italics the CSM's description for the signal function.

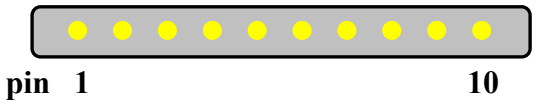


function	pin	pin	function
		1	*not connected*
CAN-GND	**6**	**2**	**CAN-L**
CAN-H	**7**	**3**	**CAN-GND**
+VAP (6-12V)	**8**	**4**	**AGND**
CAN-POWER (8-12V)	**9**	**5**	**CAN-SHIELD**

Table 5. Layout of the *CAN* connectors pins; there are 2 connectors on each MDT-DCS module for easy daisy-chaining multiple modules on one CAN-bus. All 9 pins of both connectors are 1-to-1 connected. *CAN-POWER* powers the CAN-driver part of the ELMB. *+VAP* powers both digital and analog parts of the ELMB. *CAN-SHIELD* is not connected to the MDT-DCS module internally. Pins 3 and 5 (*CAN-GND*) are connected internally; if only one pin is connected externally in the cable it must be pin 3 (CANopen cable definition).

The MDT-DCS module's serial number can be read out remotely (actually it is the serial number of the ELMB module inside; this means the ELMB inside should *not* be exchanged!).

The module's CAN node identifier is stored in ELMB EEPROM (so not set by means of the ELMB's dip-switches) and can be changed remotely, if necessary.



Pin	Function	Comment
1	***SCLK***	SPI Serial Clock (to ADC)
2	GND	
3	***SDI***	SPI Serial Data In (to ADC)
4	GND	
5	***SDO***	SPI Serial Data Out (from ADC)
6	GND	
7	***CS***	Chip Select (to ADC)
8	***ID***	1-Wire interface (to ID-chip)
9	–	
10	V+	from CAN-connector (pin 8)

Table 6. Layout of the B-sensor module connector pins.

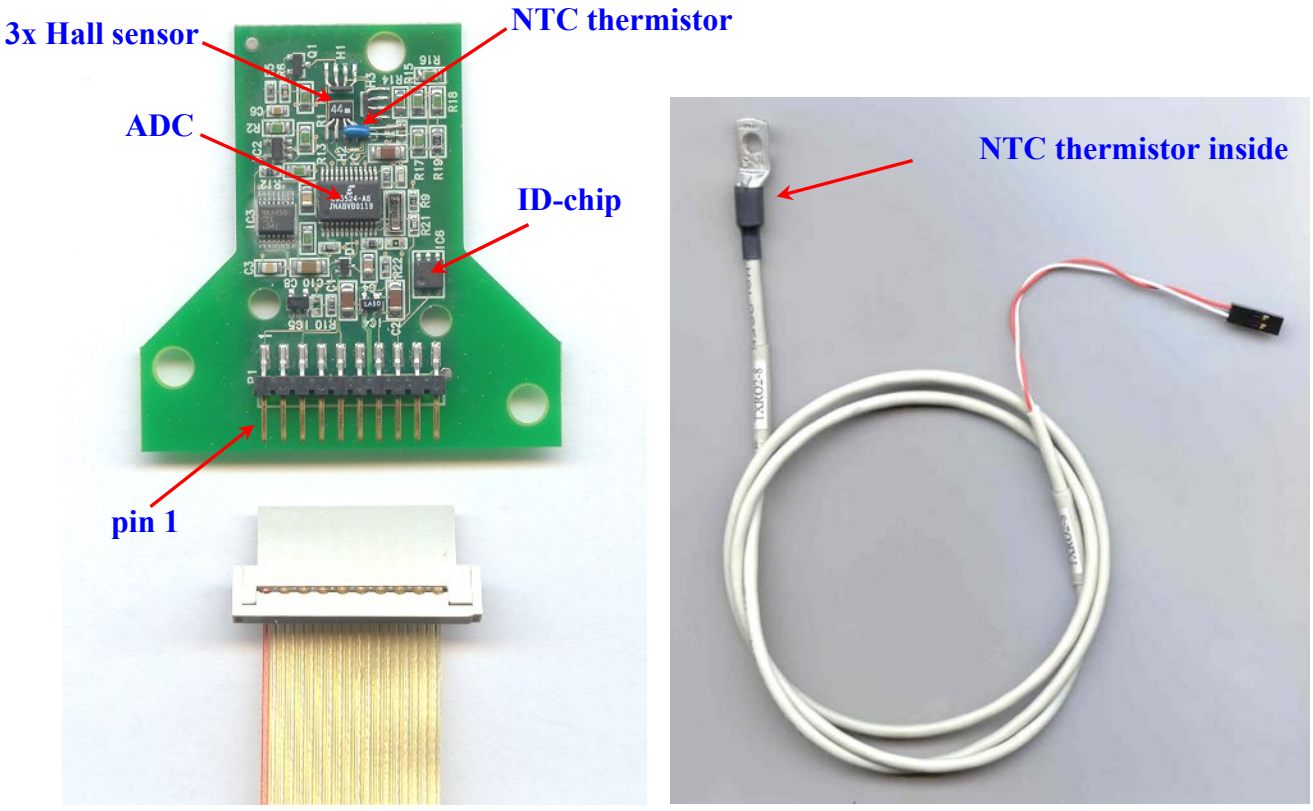


Figure 3. Left: MDT-DCS B-sensor module and cable.
Right: T-sensor cable with integrated NTC thermistor.

Figure 3 shows some images with details of the B-sensor module and temperature sensor.

The module has 2 CAN-bus connectors to enable easy daisy chaining of multiple modules on one CAN-bus. The last module on the bus must be equipped with a termination resistor (120 Ω, or 180 Ω in the Y-shaped bus layout used for MDT CAN-buses), using a special cable-less connector with the terminator installed inside the connector housing, which is then plugged into the empty CAN-connector of the last module on the bus.

2.2 ELMB / MDT-DCS Interface and ELMB Jumpers

This section describes how the ELMB board inside the MDT-DCS module interfaces hardware-wise to the rest of the system, and explains the function of the jumpers and switches present on the ELMB. It is given here for reference only.

Table 7 shows the mapping of I/O-pin-to-function of the ATmega128 microcontroller on the ELMB inside the MDT-DCS module:

- **ADC_*xxx*** is the SPI interface for the ELMB on-board ADC with monitors up to 64 channels of MDT-chamber T-sensors (NTCs).
- **AUX_IO** is the spare interface with 5 digital I/O lines (function to be defined; not yet under control of the firmware); present on module connector labelled ***SPI-AUX***.
- **B_*xxx*** is the SPI interface including two chip-select lines (**B_CS*x***) and 2 lines carrying the 1-Wire protocol for the Identification-chips (**B_ID*x***), for up to 2 B-sensor modules on the MDT-chamber; present on module connectors labelled ***B-sensor 0*** and ***B-sensor 1***.
- **CSM_*xxx*** is the SPI interface to the CSM front-end electronics ELMB-ADC which monitors up to 64 parameters; present on module connector labelled ***CSM-ADC.***
- **DIGIO*x*** are digital in- and outputs from/to the CSM front-end electronics (exact function still to be defined; the firmware assumes a default configuration of inputs and outputs, but this can be changed; see *Object Dictionary*); present on module connectors labelled ***JTAG*** and ***CSM-ADC***.

I/O PORT: **Function:**	**A** **In/Out**	**B** **In/Out**	**C** **In/Out**	**D** **In/Out**	**E** **In/Out**	**F** **I/O/ADC**
pin 0	*TDI*	**x**	*B_CS0*	**x**	**x**	*B_ID0*
pin 1	*TCK*	*SCLK*	*B_CS1*	**x**	**x**	*B_ID1*
pin 2	*TMS*	*SDI*		**x**	**x**	*DIGIO5*
pin 3	*TDO*	*SDO*	*CSM_CS*	*B_SDI*	*B_SCLK*	*DIGIO6*
pin 4	*DIGIO1*	**x**	*AUX_IO1*	*ADC_SCLK*	*CSM_SCLK*	*DIGIO7*
pin 5	*DIGIO2*	**x**	*AUX_IO2*	*ADC_SDI*	*CSM_SDI*	
pin 6	*DIGIO3*	**x**	*AUX_IO3*	*ADC_SDO*	*CSM_SDO*	*AUX_IO5*
pin 7	*DIGIO4*	**x**	*AUX_IO4*	*ADC_MUX*	*CSM_MUX*	*B_SDO*

Table 7. I/O-pin functions of the ELMB microcontroller (ATmega128) on the MDT-DCS module:
x = *NOT* available externally (used internally by ELMB).
SCLK/SDI/SDO = lines carrying **SPI**-protocol for the on-board CAN-controller.
(see text above for explanation of other signals).
Greyed out pin identifiers are not implemented in the MDT-DCS ELMB firmware (yet).

Using the ELMB's onboard DIP-switches a node identifier can be set between 1 and 63 (has to be unique on the CAN-bus the board is connected to), using 6 of the 8 switches, and a CAN-bus baud rate of 50, 125, 250 or 500 kbit/s, using the 2 remaining switches. See Figure 4 below for details. A label on the front panel shows the node identifier of the MDT-DCS module. Default the baud rate is set to 125 kbit/s.

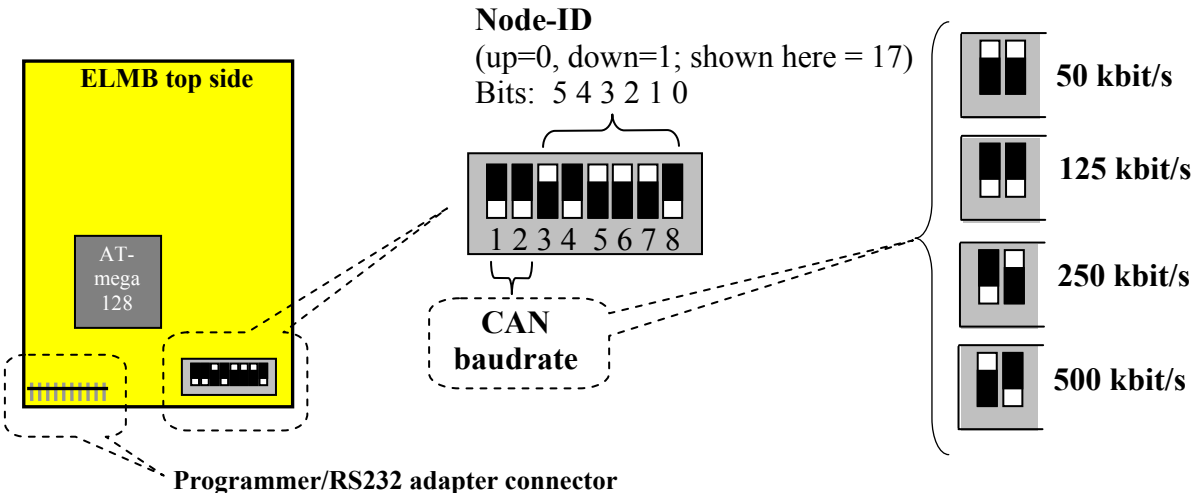


Figure 4. Location and function of ELMB DIP-switches and jumpers.

Note that, starting with MDT-DCS firmware version 2.3, it is possible to configure the node identifier remotely, i.e. using standard CANopen messages; see objects 3300h and 3301h in the MDT-DCS Object Dictionary for more details.

Once the Node-ID has been changed through CAN, the DIP-switch setting for the Node-ID is ignored, and the Node-ID is read from a fixed location in the ELMB's EEPROM. The baud rate setting is not affected.

***NB**: this feature should only be used if the ELMB's Bootloader firmware is version 1.3 or later!*

3 Initialisation

When the MDT-DCS ELMB firmware initialises, all hardware devices are reset and configured (CAN-controller, ADC for the NTCs, ADC on the CSM, the ADCs on the B-sensor modules, JTAG interface, etc.) and error counters and registers are reset. Digital outputs are initialised on the occurrence of a *hard* reset, but not on a *soft* reset (see below).

After power-up, watchdog reset, manual reset or a *CANopen* initiated reset action (i.e. by an NMT *Reset-Node* message, see below) a CANopen node sends a so-called ***Boot-up*** message (as defined by the *CANopen* standard) as soon as it has finished initializing (hardware, software); this is a CAN-message with the following syntax:

MDT-DCS module (NMT-Slave) → Host (NMT-Master)

COB-ID	Data Byte 0
700h + *NodeID*	0

NodeID is the CAN node identifier (initially) set by means of the ELMB onboard DIP-switches to a value between 1 and 63, as shown earlier in Figure 4. *NodeID* must be in the range between 1 and 127.

To *start* the MDT-DCS application in the CANopen sense of the word, the following CANopen NMT message must be sent:

Host (NMT-Master) → MDT-DCS module (NMT-Slave)

COB-ID	Data Byte 0	Data Byte 1
000h	01h (*Start_Remote_Node*)	*NodeID* or 0 (0: all nodes on the bus)

There is no reply to this message.

Now the MDT-DCS module is *Operational*, meaning that it monitors I/O channels (depending on configuration) and can send and receive (and processes) CANopen ***PDO*** messages, which carry the application data (see next sections).

Optionally a feature called *auto-start* may be enabled, so that the MDT-DCS module automatically goes to *Operational* state after power-up or reset. The *auto-start* feature can be configured in *OD* index 3200h, subindex 2.

To generate a *soft* reset the following *CANopen* NMT message must be sent:

Host (NMT-Master) → MDT-DCS module (NMT-Slave)

COB-ID	Data Byte 0	Data Byte 1
000h	81h (*Reset_Node*)	*NodeID* or 0 (0: all nodes on the bus)

Again, there is no reply to this message.

Note that at power-up it is the *Bootloader* application firmware that becomes active first and is in control of the MDT-DCS module; the Bootloader reports its presence by sending the following Emergency message (see also section 8):

Bootloader → Host

COB-ID	Byte 0-1	Byte 2	Byte 3-7
080h + *NodeID*	Emergency Error Code (00h 50h)	Error Register (Object 1001h) (80h)	Manufacturer specific error field (FEh 01h 28h ZZh 00h) (ZZh = MCUCSR)

(*MCUCSR* = MCU Control and Status Register; for details see section 10 or the ATmega128 datasheet).

Having the Bootloader activate at power-up guarantees that it is always possible to upload new application software to the ELMB, even when the application currently programmed in the ELMB is faulty or corrupted.

After about 4 s the Bootloader automatically jumps to the application. Alternatively, the Bootloader starts the application immediately, if it receives an NMT *Reset-Node* message –as shown above- within this period.

4 Node Guarding and Life Guarding

Node Guarding in CANopen is a mechanism whereby an *NMT-master* checks the state of other nodes on the bus, at regular intervals. It can do this in one of two different ways:

1. The master sends a Remote Transmission Request (RTR) for the Node Guard message, to each node on the bus, in turn; a node that receives the RTR, sends the Node Guard message, which contains one data byte indicating the (CANopen) state of the node, as well as a toggle bit. If a node does not reply the master should signal this to the higher-level software and/or take appropriate action.
   The RTR for the Node Guard message looks like this (a Remote Frame, so the CAN-message has no data bytes):

   Host (NMT-Master) → MDT-DCS module (NMT-Slave)

COB-ID
700h + *NodeID*

   The reply Node Guard message from a node looks like this:

   MDT-DCS module (NMT-Slave) → Host (NMT-Master)

COB-ID	DataByte 0
700h + *NodeID*	bit 7: *toggle bit*, bit 6-0: *state*

2. Each node on the bus sends a Heartbeat message at regular intervals; typically, the NMT-master monitors these messages and keeps a time-out period for each node. The master detects nodes that stop sending their Heartbeat messages and should signal this to the higher-level software and/or take appropriate action.

A Heartbeat message looks like this:

MDT-DCS module (Heartbeat producer) → Consumer(s) (e.g. NMT-Master)

COB-ID	DataByte 0
700h + *NodeID*	*state*

State is one of these CANopen states: 0 (*Initializing*), 4 (*Stopped*), 5 (*Operational*) or 127 (*Pre-operational*). Note that this makes the *Boot-up* message the first Heartbeat message after a node reset (see previous section).

According to the CANopen standard, a node is not allowed to support both Node Guarding and Heartbeat protocols at the same time. The MDT-DCS module supports both methods of Node Guarding (but indeed not at the same time), i.e. it can send the Node Guard message or it can send the Heartbeat message with an interval, which is configurable in *OD* index 1017h.

Life Guarding in CANopen is a mechanism whereby a node checks the aliveness of the host or master, by applying a time-out on messages received. CANopen defines that the message to time-out is the RTR for the Node Guard message, sent by the NMT-master; however, the MDT-DCS module resets its Life Guarding timer at each properly received message addressed to it.

Life Guarding is controlled through *OD* objects 100Ch and 100Dh. In the MDT-DCS module the Life Guarding time-out can be set between 1 and 255 seconds, by setting *OD* index 100Dh to the corresponding value, or can be switched off, by setting *OD* index 100Dh to zero.

If a Life Guarding time-out occurs, the node should take whatever appropriate action. The MDT-DCS module resets and reinitializes the CAN-controller, and (tries to) resume(s) normal operation, after sending an Emergency message (see section 10).

5 MDT On-Chamber Sensors Monitoring

5.1 Data Read-out

Each data object in the MDT-DCS module can be accessed through the CANopen *Object Dictionary* (*OD*). The CANopen ***SDO*** (Service Data Object) confirmed message mechanism is used to read from and write to data objects in the *OD*.

A complete overview of the Object Dictionary of the MDT-DCS module can be found in section 9.

A more efficient method of read-out of data from the MDT-DCS module is offered by the CANopen mechanism of ***PDO*** (Process Data Object) messages. This is an unconfirmed message mechanism without protocol overhead, and thus much more suitable for regular monitoring of the *process* data of the MDT-DCS module, such as the T- and B-sensor data. The sending of this type of messages may be triggered by a host system or autonomously by the MDT-DCS module firmware.

From the point of view of the MDT-DCS module data are transmitted by a ***PDO*** message, called a ***Transmit-PDO*** (or ***TPDO***), and data are received in a PDO message, called a ***Receive-PDO*** (or ***RPDO***). In CANopen the CAN-identifier, message content and *transmission type* of PDO messages may be configurable (configure by writing to the appropriate objects in the Object Dictionary using the SDO mechanism).

However, the CANopen standard defines a predefined set of CAN-identifiers (the so-called ***Predefined Connection Set***), defining which CAN-identifier to use for which kind of CANopen message, without the need for the node to support configuration. The MDT-DCS module uses this set of identifiers. Also the PDO message content is fixed in the MDT-DCS module and cannot be changed. The content of PDO messages can be found and read from the *OD* from objects called *PDO mapping objects* (stored at fixed entries in the *OD*).

A feature that is configurable on the MDT-DCS module is the so-called *transmission type* of the TPDOs, which controls what triggers it to send its 'process' data, e.g. periodically, on request or on-change. For each of the monitored subsystems (T, B, front-end) this is described in the sections following.

Serious problems occurring during read-out, e.g. with the ADC hardware, are reported in so-called CANopen ***Emergency*** messages. A list of *Emergency* messages the MDT-DCS module can generate can be found in section 10, including a description of the problem.

5.2 T-sensor Read-out

5.2.1 T-sensor Data

T-sensor data is produced by the MDT-DCS module in the form of temperature readings in millidegrees centigrade of the NTC sensors (optionally as resistance values in Ohms). Even-numbered ADC channels measure the voltage across an NTC and odd-numbered channels the voltage resulting from the corresponding current through a precision resistor. A division results in the NTC resistance value, which is then converted to a temperature (see end of this section for the conversion formula used) and sent in a CAN-message by the module.

The MDT-DCS module sends one PDO message containing 4 bytes for every T-sensor. The CAN-identifier used for this PDO is the so-called *2nd-transmit-PDO* (**TPDO2**) of the CANopen *Predefined Connection Set*.

The number of T-sensors read out can be set by configuring the number of analog channels to any value up to 64 by writing to *OD* index 2100h, subindex 1. The number of T-sensors read out is this number divided by 2 (due to the two-analog-inputs measurement per sensor).

"T-sensor" number 30 and 31 are in fact onboard reference resistors (of 16369 Ω and 341.6 Ω, representing temperatures of 0.0°C and 100.0°C, respectively) whose values may be read out to check the proper functioning of module and ADC.

The setting of *OD* index 4400h determines whether the readings in the PDO messages are in Ohms or in millidegrees centigrade. The default setting is degrees.

The MDT-DCS module produces a 4-databyte TPDO2 per T-sensor formatted either (when *OD* index 4400h is set to 1) as:

MDT-DCS module → Host

COB-ID	Data Byte 0	Data Byte 1-3
280h + *NodeID*	NTC number	Temperature [m°C]

with:

Temperature: 24-bits temperature reading in millidegrees centigrade, LSB in byte 1, MSB in byte 3; invalid readings and ADC errors result in a temperature value of FFFFFFh (16777215).

NTC number: Number between 0 and 29.

or formatted (when *OD* index 4400h is set to 0) as:

MDT-DCS module → Host

COB-ID	Data Byte 0	Data Byte 1	Data Byte 2-3
280h + *NodeID*	NTC number	Status + ADC-config	Resistance [Ω]

with:

Resistance: 16-bits NTC resistance value in Ω, LSB in byte 2, MSB in byte 3.

NTC number: Number between 0 and 29.

Status+ADC-config: **bit 7**: Conversion status: 1=ERROR (overflow or oscillation occurred during at least one of the two ADC conversions), 0=OKAY.
bits 6-0: ADC configuration: conversion word rate (bits W0, W1 and

W2), gain range (bits G0, G1 and G2) and unipolar or bipolar (bit U/B); see below. For definitions see *OD* index 2100h, sub 2, 3 and 4.

BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Meaning	*Error*	W2	W1	W0	G2	G1	G0	U/B

The method by which all 30 (or less) T-sensors is read out depends on the *transmission-type* of TPDO2, which can be set by the user to the required value by writing to *OD* index 1801h, subindex 2 of the MDT-DCS module. The value may be stored in onboard EEPROM permanently, so that it will be the default transmission type after every subsequent reset or power-up. The default value before configuration can be found in the *OD* listing in section 9.

The following modes of TPDO2 transmission are supported (see *OD* index 1801h, subindex 2 and 5:

- **PDO transmission type 1**:
  after every so-called **SYNC** message issued on the CAN-bus the MDT-DCS module starts an analog input channel scan and sends (up to) 32 TPDO2 messages, one message for every T-sensor. Two A/D conversions have to be done for every T-sensor so it can take up to about 30 seconds before all TPDO2s have been sent, depending on how the ADC has been configured (the ADC conversion rate can be as low as 1.88 Hz).
  The SYNC message is a CAN-message with a fixed COB-ID and no data bytes:

  Host → all (SYNC-)slave nodes

COB-ID
080h

  Note that all nodes that have PDOs configured to respond to a SYNC message will respond to the SYNC, which is a broadcast message.

- **PDO transmission type 255**:
  after every so-called Remote Transmission Request (**RTR**) for TPDO2 the MDT-DCS module starts an analog input channel scan and sends (up to) 32 TPDO2 messages, one message per T-sensor. The *Remote Frame* CAN-message that constitutes this RTR has no data bytes and looks like this:

  Host → MDT-DCS module

COB-ID
280h+*NodeID*

  Note that an RTR is sent to and received/processed by only one particular node.

- **Event Timer > 0**:
  If TPDO2's *event timer* (*OD* index 1801h, sub 5) is set to a value unequal to zero (*event timer* is expressed in units of 1 s and must be <=255) the MDT-DCS module automatically starts an analog input channel scan (resulting in up to 32 TPDO2 messages, one message per T-sensor) periodically, triggered by a timer (in this mode an RTR or SYNC message also triggers an input scan, depending on the transmission mode as shown above). Also see section 5.2.4.

Optionally a reset and calibration sequence can be done before each ADC channel scan. This feature can be enabled via *OD* index 2300h (useful perhaps for increasing radiation tolerance).

Individual T-sensors resistance values can be read out using CANopen **SDO** messages by reading from *OD* index 4000h.

Individual T-sensors temperature values can be read out using CANopen **SDO** messages by reading from *OD* index 4010h.

Individual analog inputs (as used in the T-sensor readout) can be read out using CANopen **SDO** messages by reading from *OD* index 6404h (in ADC counts) or from OD index 4300h (in microvolts). Note that the data in objects 6404h and 4300h contain a 'flags' byte (generated by the ADC), which is formatted as follows:

BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Value	1	1	1	0	CI1	CI0	OD	OF

with *CIn* = Channel Indicator bits, indicating which CS5523 ADC physical channel (1 to 4, coded as 00, 01, 10 and 11, respectively) is used, *OD* = Oscillation Detect Flag bit and *OF* = Over-range Flag bit.

5.2.2 ADC Data Conversion

The MDT-DCS T-sensor is an NTC, *Thermometrics* type number DC95F502W, with a nominal resistance of 5 kΩ. See Appendix B for datasheet and temperature data of the NTC.

Voltage $\mathbf{U_V}$ across the NTC on ADC-channel 2**n* is measured, then the current through the NTC is measured by measuring voltage $\mathbf{U_I}$ which the current generates across a 10 kΩ resistor (± 1%) on ADC-channel 2**n*+1. With the ADC set to 2.5V unipolar range, the conversion from raw ADC counts $\mathbf{A_{2n}}$ and $\mathbf{A_{2n+1}}$ to resistance value $\mathbf{R_{NTC}}$ of T-sensor *n* is done by:

$$\mathbf{R_{NTC} = U_V/(U_I/10^4) = 10^4((2.5*A_{2n})/0xFFFF)/((2.5*A_{2n+1})/0xFFFF)) = 10^4 A_{2n} / A_{2n+1}}$$

To calculate temperature **T** (in ºC, in the range from 0 to 100 ºC) of the NTC from NTC resistance value $\mathbf{R_{NTC}}$ (in Ω), the following approximation equation (see Appendix B) is used:

$$\mathbf{T = (1.0 / (a + b \ln(r) + c (\ln(r))^2 + d (\ln(r))^3)) - 273.15}$$

with **r** = $\mathbf{R_{NTC}}$/5000,
and **a** = 3.3540154E-03
b = 2.5627725E-04
c = 2.0829210E-06
d = 7.3003206E-08
when 3.274 >= **r** > 0.36036 (i.e. when 0º C <= **T** < 50º C),
or **a** = 3.3539264E-03
b = 2.5609446E-04
c = 1.9621987E-06
d = 4.6045930E-08
when 0.36036 >= **r** >= 0.06831 (i.e. when 50º C <= **T** <= 100º C).

The conversion functions above are applied by the MDT-DCS firmware to the ADC readings when temperature read-out is set to millidegrees centigrade, which is the default.

5.2.3 ADC Raw Data

Starting with MDT-DCS firmware version 2.2, it is possible to configure the TPDO2 containing the T-sensor ADC data, as described in the previous section, such that each PDO message contains an individual analog input conversion value (in ADC counts), i.e. the PDO message contains an object from *OD* index 6404h.

Note:

- this mode is the default mode of read-out of the socalled *EndCap*-type of MDT-DCS modules, where the NTCs have been replaced by voltage-based T-sensors (plus additional circuitry); the host system does the conversion to temperature units.
- read-out in this mode results in calibrated values, because the ELMBs have calibration constants stored onboard for every possible voltage-range; the constants are applied by the firmware, so the conversion from ADC-counts to voltage –by the host system– is straightforward (for example: in 2.5V unipolar mode, an ADC conversion count of 65535 corresponds indeed to 2.5V).

When *OD* index 4401h is set to 1, the MDT-DCS module produces a 4-databyte TPDO2 formatted as follows:

MDT-DCS module → Host

COB-ID	Data Byte 0	Data Byte 1	Data Byte 2-3
280h + *NodeID*	Channel number	Status + ADC-config	ADC count

with:

ADC count: 16-bits ADC count, LSB in byte 2, MSB in byte 3.

Channel number: Number between 0 and 63 (ADC input channel number).

Status+ADC-config: **bit 7**: Conversion status: 1=ERROR (overflow or oscillation), 0=OKAY.
bits 6-0: ADC configuration: conversion word rate (bits W0, W1 and W2), gain range (bits G0, G1 and G2) and unipolar or bipolar (bit U/B); see above. For definitions see *OD* index 2100h, sub 2, 3 and 4.

5.2.4 Readout-on-Change

Starting with MDT-DCS firmware version 2.3.1 a so-called 'readout-on-change' feature was added. It means that the MDT-DCS module automatically and periodically scans the T-sensor channels and sends a message (a TPDO2) only for a T-sensor that changed its value with a preset minimum value (the *'delta'*). This *delta* value is one of the ADC's configuration parameters, and can be set to any value. There is one delta that applies to all T-sensor channels.

To enable this feature for the T-sensors:

- set the TPDO2 event timer (Object 1801h, sub 5) to a value > 0: this will be the period (in seconds) between two consecutive T-sensor channel scans,
- set the T-sensor ADC delta value (Object 2100h, sub 22) to a value > 0,
- set the MDT-DCS module to *Operational*.

The first scan cycle does not produce any output, but the T-sensors are read out, and the readings are used as reference values to detect a 'delta' change in any of the values in subsequent channel scans. As soon as this occurs the value is sent and taken as the new reference for the channel that changed.

At any time a host system may request a read-out of all T-sensors by sending a SYNC or RTR message to the MDT-DCS module; it does not influence the 'scan-for-change' feature, although an ongoing T-sensor channel scan is aborted; current T-sensor reference values are not changed by this action. A next channel scan is automatically started when the timer expires again.

If the TPDO2 event timer is set to a value > 0, but delta is set to 0, the 'normal' procedure of read-out takes place: every *n* seconds all T-sensors are read out and their values sent in messages, as described in section 5.2.1.

Note that TPDO event timer triggered readout takes place only when the node is in *Operational* state.

Note that the delta value is always taken to be in the units in which read-out currently takes place, i.e. millidegrees, Ohms or raw ADC-counts. So if you change your unit of read-out, the delta value itself does not change, but its unit does !

5.3 B-sensor Read-out

By writing to *OD* index 2800h, none, or up to four B-sensor modules can be selected, in the form of a bit mask, i.e. if *OD* index 2800h has value Fh, all four B-sensor modules are present). The default is: no B-sensor module present, *OD* index 2800h has value 0 (zero).

The MDT-DCS module has originally been designed to read out up to two B-sensor modules only. In case three or four B-sensor modules are connected to one MDT-DCS module, they must be connected to a cable in pairs, as illustrated in Figure 5. The module numbering is fixed and is as shown in Figure 5.

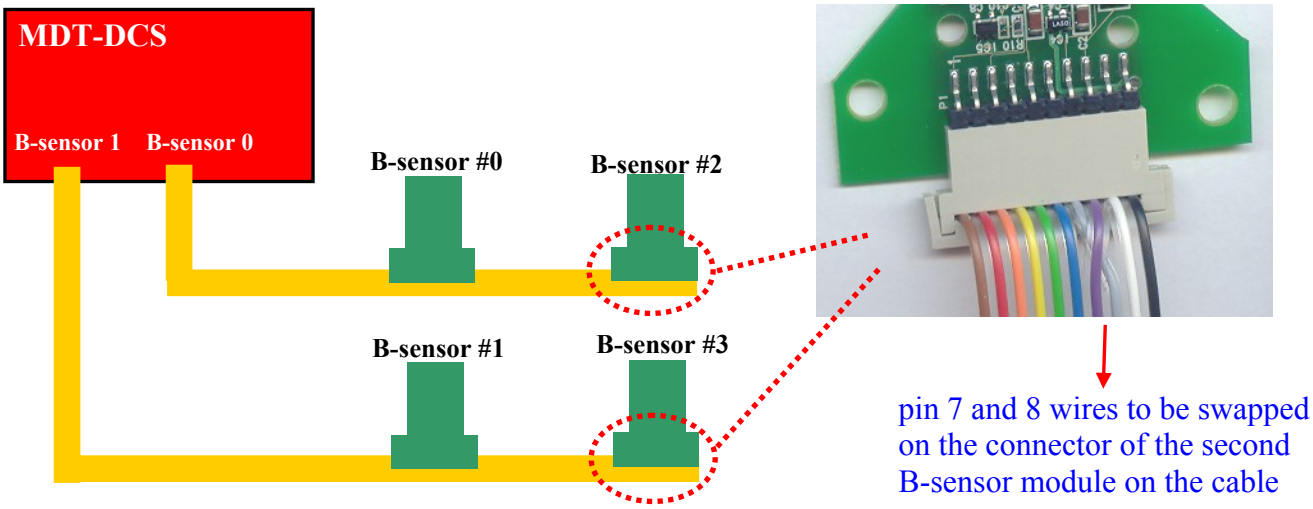


Figure 5. Connecting more than two B-sensor modules to one MDT-DCS module. (Note: the cable shown in the picture is of a type not approved for ATLAS !)

5.3.1 B-sensor Data

The MDT-DCS module sends one PDO message containing 5 bytes for each B-sensor input and per B-sensor module 4 inputs are read: Hall sensors H1, H2 and H3 and the temperature sensor. The CAN-identifier used for this PDO is the so-called *4th-transmit-PDO* (**TPDO4**) of the CANopen *Predefined Connection Set*.

The MDT-DCS module produces the following 5-databyte TPDO4:

MDT-DCS module → Host

COB-ID	Data Byte 0	Data Byte 1	Data Byte 2-4
480h + *NodeID*	Channel number	ADC-config	24-bit ADC value

with:

ADC value: Signed/unsigned 24-bits ADC value, LSB in byte 2, MSB in byte 4.
Note: Hall sensors: 24-bit signed value; T-sensor: 24-bit unsigned value (either an ADC count or a temperature in millidegrees centigrade depending on the setting of *OD* index 4400h),

Channel number: Number between 0 and 15.
Chan 0-3: Hall sensor H1, H2, H3 and T-sensor resp. of B-sensor #0,

Chan 4-7: Hall sensor H1, H2, H3 and T-sensor resp. of B-sensor #1.
Chan 8-11: Hall sensor H1, H2, H3 and T-sensor resp. of B-sensor #2.
Chan 12-15: Hall sensor H1, H2, H3 and T-sensor resp. of B-sensor #3.

ADC-config: **bit 7**: not used.
bits 6-0: ADC configuration: conversion word rate (bits W0, W1 and W2), gain range (bits G0, G1 and G2) and unipolar or bipolar (bit U/B); see below. For definitions see *OD* index 2500h/2501h, sub 2,3,4,5,6 and 7.

BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Meaning	-	W2	W1	W0	G2	G1	G0	U/B

The method by which the B-sensor module inputs are read out depends on the *transmission-type* of TPDO4, which can be set in *OD* index 1803h, subindex 2 of the MDT-DCS module. The method options are identical to what has been described for the read-out of the T-sensors in section 5.2.

Optionally a reset and calibration sequence can be done before each B-sensor ADC channel scan. This feature can be enabled via *OD* index 2700h (useful perhaps for increasing radiation tolerance).

Individual B-sensor module channels (there are actually 7 per module) can be read out using CANopen **SDO** messages by reading from *OD* index 4200h to 4203h (see *OD* tables for a description of each individual channel).

5.3.2 ADC Data Conversion

The interpretation of the Hall sensor ADC values and conversion to physical values will be done offline using a set of calibration tables accompanying each individual B-sensor module. Until these tables are available the user himself must interpret the data.

The B-sensor module's T-sensor is an NTC, *Thermometrics* type number DC95F502W, with a nominal resistance of 5 kΩ. See Appendix B for datasheet and temperature data of the NTC.

Table 8 shows a list of resistance values $\mathbf{R_{NTC}}$ for this NTC at different temperatures, and the resulting B-sensor module ADC input voltage. In the shaded part of the table (between 0º and 70º C) the precision is ±0.2º C.

The ADC input voltage $\mathbf{V_{NTC}}$ can be expressed as:

$$\mathbf{V_{NTC} = V_{ref} - V_{cc}R_{NTC} / (R_{NTC} + R_{ref})}$$

which can be rewritten as:

$$\mathbf{R_{NTC} = R_{ref} (V_{ref} - V_{NTC}) / (V_{NTC} + Vcc - V_{ref})}$$

With $\mathbf{R_{ref}}$ = 23.2 kΩ, $\mathbf{V_{cc}}$ = 5 V and $\mathbf{V_{ref}}$ = 2.5 V this results in:

$$\mathbf{R_{NTC} = 23200 (2.5 - V_{NTC}) / (V_{NTC} + 2.5)}$$

$\mathbf{V_{NTC}}$ is the voltage value calculated from the 24-bit ADC value **A.**
The ADC input has been calibrated to give **A**=0 (000000h) at 0 ºC (i.e. at 0.4315 V) and **A**=16777215 (0xFFFFFF) at 100 ºC (i.e. at 2.4275 V), so that $\mathbf{V_{NTC}}$ can be expressed as:

$$\mathbf{V_{NTC} = 0.4315 + (2.4275 - 0.4315)A/FFFFFFh = 0.4315 + 1.996A/FFFFFFh}$$

So $\mathbf{R_{NTC}}$ can be calculated directly from ADC value **A** as follows:

$$\mathbf{R_{NTC} = 23200\ (2.0685 - a) / (2.9315 + a)}$$

with **a** = 1.996 **A** / 16777215.

With $\mathbf{R_{NTC}}$ known, the temperature (in ºC) can now be calculated using the equation(s) for **T** from the previous section.

The conversion equations described above are applied by the MDT-DCS firmware when temperature read-out is set to millidegrees centigrade, which is the default setting.

5.3.3 B-sensor Serial Number

Each B-sensor module comes equipped with a unique serial number, which is factory-lasered in the on-board Dallas DS2401 device.
The 64-bit (8-byte) serial number is used to uniquely identify each module, for instance, to match each module with its calibration data, which are stored off-line.

The serial numbers of the four B-sensor modules can be read from *OD* Objects 2900h, 2901h, 2902h or 2903h. The least significant 4 bytes are read from subindex 1 and the most significant 4 bytes from subindex 2. Starting with MDT-DCS firmware version 2.4.0 the least- or most-significant sets of 4 bytes can be read in any order.

The layout of the 64-bit serial number is as shown below:

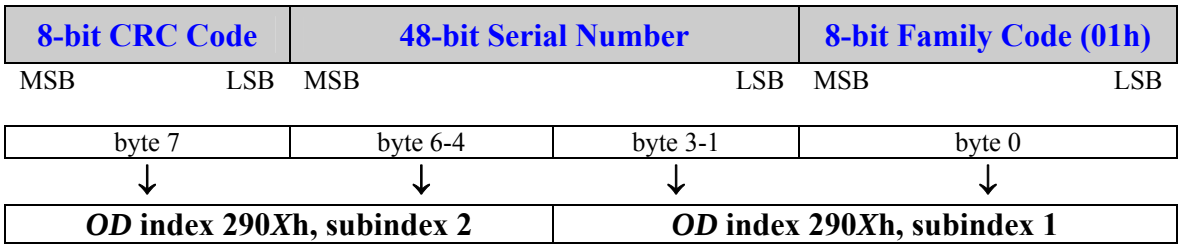


Figure 6. B-sensor 64-bit Serial Number and its mapping to Object Dictionary (*OD*) objects (with *X*=0 to 3).

The MDT-DCS module checks the correctness of the serial number CRC when *OD* Object 2900h to 2903h is read, so a valid reply implies the CRC was correct: it is not necessary for the host to recalculate the serial number CRC.

Temperature	Normalized Resistance	Resistance	AIN4 (ADC)
[C]	Ohm	Ohm	Volt
-50	68.60	343000.00	-2.1832
-45	48.16	240800.00	-2.0606
-40	34.23	171150.00	-1.9031
-35	24.62	123100.00	-1.7071
-30	17.91	89550.00	-1.4712
-25	13.17	65850.00	-1.1974
-20	9.782	48910.00	-0.8913
-15	7.339	36695.00	-0.5633
-10	5.558	27790.00	-0.2250
-5	4.247	21235.00	0.1106
0	3.274	16370.00	0.4315
5	2.544	12720.00	0.7294
10	1.992	9960.00	0.9982
15	1.572	7860.00	1.2347
20	1.250	6250.00	1.4389
25	1.000	5000.00	1.6135
30	0.8056	4028.00	1.7603
35	0.6530	3265.00	1.8831
40	0.5326	2663.00	1.9852
45	0.4369	2184.50	2.0697
50	0.3604	1802.00	2.1396
55	0.2989	1494.50	2.1974
60	0.2491	1245.50	2.2452
65	0.2087	1043.50	2.2848
70	0.1756	878.00	2.3177
75	0.1485	742.50	2.3449
80	0.1261	630.50	2.3677
85	0.1075	537.50	2.3868
90	0.09209	460.45	2.4027
95	0.07916	395.80	2.4161
100	0.06831	341.55	2.4275
105	0.05916	295.80	2.4371

Table 8. NTC resistance/temperature table, and resulting B-sensor ADC input voltage (Normalized resistance table taken from datasheets in Appendix B).

6 CSM Front-end Electronics Monitoring and Control

6.1 Analog Inputs

The *CSM-SPI* connector provides the interface to an ADC, identical to the ADC on the ELMB (which is used for the NTC temperature sensors on the MDT chamber), integrated in the CSM front-end electronics, capable of monitoring up to 64 analog input channels on the CSM. The analog values monitored include:

- Mezzanine analog voltage
- Mezzanine digital voltage
- Mezzanine temperature
- Motherboard 4.5V
- CSM +5V_{cc}, -5V_{EE}, 3.3V, 2.5V, 1.8V and 1.5V
- CSM temperature

A list of ADC channels and what parameter they represent, is shown in Table 9 below.

ADC ch	Source	ADC ch	Source	ADC ch	Source	ADC ch	Source
0	Mezz 16 Temp	16	Mezz 6 Temp	32	Mezz 10 Temp	48	Mezz 0 Temp
1	Mezz 16 Analog	17	Mezz 6 Analog	33	Mezz 10 Analog	49	Mezz 0 Analog
2	Mezz 16 Digital	18	Mezz 6 Digital	34	Mezz 10 Digital	50	Mezz 0 Digital
3	CSM 2.5V	19	CSM 3.3V	35	CSM 1.8V	51	CSM V_{CC}
4	Mezz 15 Temp	20	Mezz 5 Temp	36	Mezz 12 Temp	52	Mezz 2 Temp
5	Mezz 15 Analog	21	Mezz 5 Analog	37	Mezz 12 Analog	53	Mezz 2 Analog
6	Mezz 15 Digital	22	Mezz 5 Digital	38	Mezz 12 Digital	54	Mezz 2 Digital
7	CSM 1.5V	23	Mezz 7 Temp	39	Mezz 11 Temp	55	Mezz 1 Temp
8	Mezz 17 Temp	24	Mezz 8 Temp	40	Mezz 14 Temp	56	Mezz 4 Temp
9	Mezz 17 Analog	25	Mezz 8 Analog	41	Mezz 14 Analog	57	Mezz 4 Analog
10	Mezz 17 Digital	26	Mezz 8 Digital	42	Mezz 14 Digital	58	Mezz 4 Digital
11	CSM 2.5V Ref	27	Mezz 7 Analog	43	Mezz 11 Analog	59	Mezz 1 Analog
12	Half CSM +5V_{CC}	28	Mezz 9 Temp	44	Mezz 13 Temp	60	Mezz 3 Temp
13	CSM Temp	29	Mezz 9 Analog	45	Mezz 13 Analog	61	Mezz 3 Analog
14	Half CSM -5V_{EE}	30	Mezz 9 Digital	46	Mezz 13 Digital	62	Mezz 3 Digital
15	CSM 2.5V Ref	31	Mezz 7 Digital	47	Mezz 11 Digital	63	Mezz 1 Digital

Table 9. Mapping of CSM FE-electronics voltages/temperatures to CSM-ADC channels.

A 2.5V reference connected to one of the 64 analog input channels is used to calibrate the ADC's 5V input range (at each power-up and reset). The input channel number where the 2.5V reference is connected can be selected in *OD* index 2101h, subindex 19. The default setting is ADC channel 11. When a value > 63 is set, an ADC-internal calibration is done; in that case only the ADC's 2.5V voltage range would give accurate readings.

The temperature sensor device used on the CSM and Mezzanine cards is the Analog Devices TMP36 (offset 0.5V, 10 mV/°C, i.e. output = 0.75V @ 25°C, range -40°C to +125°C, accuracy ±2°C). With the ADC set to 5V bipolar range, the conversion from raw ADC count **A** to degrees Celcius can be done using:

0.5+0.01*Celcius = Volts or Celcius = 100*Volts-50,
with Volts = 5.0***A**/0x7FFF = 5.0***A**/32767.0,
this leads to **Celcius = 0.01526*A – 50**.

The MDT-DCS module sends one PDO message containing 4 bytes for every ADC input. The CAN-identifier used for this PDO is the so-called *3rd-transmit-PDO* (**TPDO3**) of the CANopen *Predefined Connection Set*.

The number of analog inputs read out can be set by configuring it to any value up to 64 by writing to *OD* index 2101h, subindex 1.

The MDT-DCS module produces the following 4-databyte TPDO3:

MDT-DCS module → Host

COB-ID	Data Byte 0	Data Byte 1	Data Byte 2-3
380h + *NodeID*	Channel number	Status + ADC-config	ADC count

with:

ADC count: 16-bits value, LSB in byte 2, MSB in byte 3. Using the default ADC configuration setttings: value 7FFFh corresponds to +5.0V, value 0000h to 0.0V, and value 8000h to -5.0V (signed 16-bits value).

Channel number: number between 0 and 63.

Status+ADC-config: **bit 7**: Conversion status: 1=ERROR (overflow or oscillation), 0=OKAY.
bits 6-0: ADC configuration: conversion word rate (bits W0, W1 and W2), gain range (bits G0, G1 and G2) and unipolar or bipolar (bit U/B); see below. For definitions see *OD* index 2101h, sub 2, 3 and 4.

BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Meaning	*Error*	W2	W1	W0	G2	G1	G0	U/B

The method by which all (64 or less) analog inputs are read out depends on the *transmission-type* of TPDO3, which can be set in *OD* index 1802h, subindex 2 of the MDT-DCS module. The options for the *transmission-type* are the same as described for the read-out of the T-sensors in section 5.2.

Optionally a reset and calibration sequence can be done before each ADC channel scan. This feature can be enabled via *OD* index 2301h (useful perhaps for increasing radiation tolerance).

Individual analog inputs can be read out using CANopen **SDO** messages by reading from *OD* index 4100h. Note that the data in objects 4100h contains a 'flags' byte (generated by the ADC), which is described in section 5.2.1.

6.1.1 Readout-on-Change

Starting with MDT-DCS firmware version 2.3.1 a so-called 'readout-on-change' feature was added. It means that the MDT-DCS module automatically and periodically scans the CSM analog channels and sends a message (a TPDO3) only for a channel that changed its value with a preset minimum value (the *'delta'*). This *delta* value is one of the ADC's configuration parameters, and can be set to any value. There is one delta that applies to all CSM channels.

To enable this feature for the CSM analog inputs, do the following:

- set the TPDO3 event timer (Object 1802h, sub 5) to a value > 0: this will be the period (in seconds) between two consecutive CSM channel scans,
- set the CSM ADC delta value (Object 2101h, sub 21) to a value > 0,
- set the MDT-DCS module to *Operational*.

For further details see section 5.2.4.

6.2 Configuration and Control

The *CSM-ADC, JTAG* and *SPI-AUX* connectors provide interfaces for additional configuration and control of the CSM front-end electronics.

The *CSM-ADC* connector provides, in addition to the serial interface to an ADC on the CSM as described in section 6.1, three general-purpose I/Os, which are available as bits 4-6 through the CANopen mechanism for Digital I/O (see section 6.2.2).

The *JTAG* connector implements a JTAG host interface and is used to configure the CSM electronics. In addition, this connector provides four general-purpose I/Os, which are available as bits 0-3 through the CANopen mechanisms for Digital I/O (see section 6.2.2).

The MDT-DCS firmware does not have any support for the interface provided by the *SPI-AUX* connector. It is spare and may be used to implement extra Digital I/Os or to drive a serial interface. Appropriate additions to the module's firmware and *Object Dictionary* would have to be made and possibly additional PDOs defined, to satisfy the requirements of the task foreseen.

6.2.1 JTAG

6.2.1.1 Implementation Overview

The MDT-DCS module supports 2 methods of uploading a JTAG bit string to the CSM electronics:

1. Relay method: the host system sends the JTAG bit string to the ELMB, in chunks; MDT-DCS shifts out each bit string chunk, immediately upon reception, into the JTAG chain. This method enables upload of arbitrary JTAG bit strings to the CSM.
2. Storage method: the host system sends a single message to trigger the upload of one (actually a pair) from a number of JTAG bit strings permanently stored in the MDT-DCS module's memory. This method enables fast (compared to the relay method) upload of any selection in any order of previously stored bit strings to the CSM.

A JTAG bit string contains always either data bits (upload takes place in TAP state *Shift-DR*), or instruction bits (upload takes place in TAP state *Shift-IR*).
Usually an instruction bit string is shifted into a JTAG chain, followed by a data bit string; the combination of such an instruction and data bit string is called here: a *JTAG-action*.

Upload method 1 supports both instruction and data bit string upload, in chunks of 32 bits (in principle up to any bit string length), or, in a socalled ‘segmented transfer’ (MDT-DCS firmware version 2.4 and newer) up to 1024*8 bits; the host system controls the order of uploading. If required, the host system may retrieve return bits for inspection. In case the 32-bit chunk upload method is used, the host must retrieve return bits after every (32-bit) chunk upload (a new chunk overwrites the previous return bits). In case of a segmented transfer a host must finish the segmented upload, and then can download all the return bits from the MDT-DCS module, in one go, also by means of a segmented transfer.

The MDT-DCS *Object Dictionary* provides objects for instruction bit string upload (OD index 4800h, 4801h and 480Ah) and data bit string upload (OD index 4803h, 4804h and 480Bh), which are accessed using standard SDO messages. See example 1 in the next section.

For efficient individual MDT-DCS and CAN-bus-wide broadcast-style bit string uploading, four *RPDO*s have been defined:

- with *COB-ID* 300h+*NodeID* and 400h+*NodeID* for instruction and data bit strings respectively, for upload to individual MDT-DCS modules,
- with *COB-ID* 500h and 580h for instruction and data bit strings respectively, for broadcast to all MDT-DCS modules on the CAN-bus.

NB: return bits are not available for read-back when using RPDOs for uploading!
See examples 2 and 3 in the next section.

Upload method 2 triggers the execution of a *JTAG-action*: an instruction bit string upload is followed by a data bit string upload into the JTAG chain; both bit strings were previously stored in the onboard non-volatile memory (EEPROM) of the MDT-DCS module and both strings have to be present and valid for the JTAG-action to be successfully executed.

With each stored JTAG-action it is possible to save (and check against a reference string) up to 32 consecutive bits of the return data bit string (*not* so for the instruction bit string). The start bit, a reference bit string and a mask are stored in the MDT-DCS module's non-volatile memory. The return status bits and/or return status error word (as one 32-bit item) of the last executed JTAG-action can be inspected (*OD* index 49F0h, sub 2 and 3). The MDT-DCS module can be configured to automatically send the status error word after completion of each JTAG-action (*OD* index 49F1h). See example 4 in the next section.

For efficient individual MDT-DCS and CAN-bus-wide broadcast requests to execute one or more of the stored JTAG-actions, two RPDOs have been defined. Due to the limited number of CAN-message buffers in the ELMB hardware the RPDO with *COB-ID* 400h+*NodeID* and *COB-ID* 580h (already used for data bit string upload) are reused for this purpose. An upload sequence of up to 7 JTAG-actions may be triggered in this way, with just one message. See examples 5 and 6 in the next section.

Note that the MDT-DCS module receives and sends PDO messages only when in (*CANopen*) state *Operational*.

6.2.1.2 JTAG-action Storage

NOTE: the JTAG-action storage features are supported starting from MDT-DCS firmware version **2.3**. Versions **2.1** and **2.2** support all JTAG features, *except* JTAG-actions. Versions **1.x** do *not* have any support for JTAG operations. Versions **2.4** and newer also support JTAG string transfer using the *CANopen Segmented SDO* protocol.

The MDT-DCS module has storage space for a total of 13 JTAG-actions, each with up to 128 instruction bits, ten of them with up to 512 data bits and three of them with up to 6272 data bits. A 16-bit CRC is stored with each bit string and checked before every upload into the JTAG chain. Measurements have shown that the MDT-DCS module can shift stored strings into the JTAG chain at a rate of about 1000 bits per 25 ms, or 40 kbits/s.

A host system sends bit strings for storage in MDT-DCS module memory in basically the same way as it sends bit strings using upload method 1; it only takes (much) more time for each chunk (of 32 bits) to be stored onboard permanently (ca.30 ms) than to be shifted into the JTAG-chain (ca.0.4 ms), in other words: to store a string of 6272 bits may take up to 8 s !

Note that writing bit strings to storage can only be done with SDO messages (both Expedited or Segmented Transfer), not with PDOs. A series of objects in the MDT-DCS *Object Dictionary* for each of the 13 JTAG-action storage spaces (*OD* indices 491Xh to 49DXh), provide access to the storage spaces and operations on the bit strings, as well as the parameters for a return bits check. See examples 7 and 8 in the next section.

6.2.1.3 Examples of MDT-DCS JTAG Operations

Examples of JTAG operations and the *CANopen* messages required are shown in the tables below.

In case of SDO messages they only show the messages that carry the data read from or written to the *Object Dictionary*. So the message with data is either generated by the host (the SDO client) or by the MDT-DCS module (the SDO server), but in all cases an SDO 'message exchange' is always initiated by the host (being the client), either writing to or reading from the *Object Dictionary* of the MDT-DCS module (being the server of the request).

For receiving and sending PDO messages the MDT-DCS module must be in state *Operational*. PDO messages are not confirmed (by a reply) by the receiver(s).

1. **Sending a JTAG data bit string and loading it into one CSM, using SDO messages.**
   Assume the bit string contains 68 bits and can be written as a number (hexadecimal) as: AFEDCBA9876543210, with the least significant bit of this number to be shifted out into the JTAG chain first. Note that each host SDO message results in an SDO reply from the MDT-DCS module, not shown in the table below, and also note that here the data is received by one and only one MDT-DCS module or CSM (according to the *NodeID* set in the SDO CAN-message sent by the host). The following sequence of messages performs the upload operation (messages in rows):

Source	SDO	Byte 4	Byte 5	Byte 6	Byte 7
host	Write OD 4803h, 0	10h	32h	54h	76h
host	Write OD 4803h, 0	98h	BAh	DCh	FEh
host	Write OD 4804h, 4	0Ah	00h	00h	00h

   Note that the final SDO message writes to Object 4804h, sub 4 in order to shift exactly 4 bits, being the final bits of the uploaded bit string. Non-significant bits in the last message *must* be zero.
   If the host wants to check the JTAG return bits, it has to request the MDT-DCS module to send the return bits after each bit string chunk written:

Source	SDO	Byte 4	Byte 5	Byte 6	Byte 7
host	Write OD 4803h, 0	10h	32h	54h	76h
MDT	Read OD 4803h, 0	XXh	XXh	XXh	XXh
host	Write OD 4803h, 0	98h	BAh	DCh	FEh
MDT	Read OD 4803h, 0	XXh	XXh	XXh	XXh

host	Write OD 4804h, 4	0Ah	00h	00h	00h
MDT	Read OD 4804h, 4	0Xh	00h	00h	00h

(Note: the final read of OD 4804h can be read from any of the subindices, and also by a read of OD 4803h; they return the same data).

Starting with MDT-DCS firmware version 2.4, objects have been added to the Object Dictionary enabling JTAG bit string up- and download by means of ***Segmented-SDO***, the standard CANopen protocol for transferring data items larger than 4 bytes.
The upload/download operation shown above is done with the following sequence of messages (messages in rows):

Source	Segmented-SDO	Bt 0	Bt 1	Bt 2	Bt 3	Bt 4	Bt 5	Bt 6	Bt 7
host	Write OD 480Bh, 0, ini	p'col	0Bh	48h	00h	0Bh	00h	00h	00h
host	Write OD 480Bh, 0	p'col	44h	00h	10h	32h	54h	76h	98h
host	Write OD 480Bh, 0, last	p'col	BAh	DCh	FEh	0Ah	00h	00h	00h
MDT	Read OD 480Bh, 0, ini	p'col	0Bh	48h	00h	0Bh	00h	00h	00h
MDT	Read OD 480Bh, 0	p'col	44h	00h	XXh	XXh	XXh	XXh	XXh
MDT	Read OD 480Bh, 0, last	p'col	XXh	XXh	XXh	0Xh	00h	00h	00h

Notes on the message sequence shown above:

- The first SDO message contains the object index (0Bh, 48h) and subindex (00h), as well as the number of bytes to be sent in this Segmented SDO, i.e 11 bytes (0Bh); this message is part of the Segmented-SDO protocol and carries no JTAG bit string data.
- The second SDO message contains in its first 2 bytes the length of the JTAG bit string to be sent in this Segmented-SDO, i.e. 68 bits (44h), part of the JTAG bit string upload protocol; bytes 4 to 7 contain the first 32 bits of the bit string to upload. (Reading a bit string by Segmented-SDO similarly results in a byte-array returned in which the first 2 bytes contain the number of bits in the contained bit string; see 5th message from top in the table above).
- Data byte 0 –not shown in the table above– contains the SDO protocol byte, not further explained in detail here.
- A JTAG bit string upload by Segmented-SDO must be completed before the return bits can be read/downloaded (by Segmented-SDO); this implies that all return bits must be stored by the MDT-DCS module; the maximum number of return bits stored is therefor limited to 1024*8, so this is also the allowed maximum length of a JTAG bit string upload by Segmented-SDO.

2. Sending a JTAG data bit string and loading it into one CSM, using PDO messages.
The bit string from example 1 is written to the same (single) MDT-DCS module:

Source	PDO(COB-ID)	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
host	400h + *NodeID*	00h	10h	32h	54h	76h	98h	BAh	DCh
host	400h + *NodeID*	0Ch	FEh	0Ah					

Note that byte 0 signifies whether the last bits are to be uploaded (being unequal to zero). A single such PDO message can have any number of bytes >= 2. There are no replies to these messages. If required the host may inspect *OD* index 4805h to make sure all bits have been received. Non-significant bits in the last message *must* be set to zero.

3. Sending a JTAG data bit string and loading it into <u>all</u> CSMs connected to the CAN-bus, using PDO messages.
The bit string from example 1 and 2 is written to all MDT-DCS modules:

Source	PDO(COB-ID)	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
host	580h	00h	10h	32h	54h	76h	98h	BAh	DCh
host	580h	0Ch	FEh	0Ah					

There are no replies to these messages. If required the host may inspect *OD* index 4805h on each MDT-DCS module to make sure all bits have been received on each module.
See also example 5 where PDO 580h is used for its other purpose: triggering a JTAG-action execution. Non-significant bits in the last message *must* be set to zero.

An error in the PDO syntax (example 2 and 3) would result in the MDT-DCS module sending the following Emergency message (see section 10):
with *XX* = TAP state, *YY* = number of bits in shift, *ZZ* = 1 (final shift) or 0 (not final shift).

Source	Emergency (COB-ID)	Byte 0-1	Byte 2	Byte 3-7
MDT	080h + *NodeID*	Emergency Error Code (00h 81h)	Error Register (Object 1001h)	Manufacturer specific error field (71h *XX YY ZZ* 00h)

4. Loading JTAG instruction and data bit strings from JTAG-action #2 storage to one CSM (i.e. executing JTAG-action #2), using an SDO message.
The host sends the following message:

Source	SDO	Byte 4	Byte 5	Byte 6	Byte 7
host	Write OD 4927h, 0	55h	–	–	–

Note that the data is received by one and only one MDT-DCS module or CSM (according to the *NodeID* set in the SDO CAN-message sent by the host).
If both the JTAG-action's stored bit strings are present and valid, and after the bit strings have been sent to the CSM, MDT-DCS sends the standard SDO reply.

If the global enable of reporting the JTAG-action status error has been set (*OD* index 49F1h set to 1), and the MDT-DCS module is in state *Operational*, the completion of a JTAG-action also results in the MDT-DCS module sending its Digital Input PDO message with bit 7 of databyte 0 acting as a toggle bit (at each JTAG-action completion the bit is toggled) and databytes 1 to 4 containing the status error word (*OD* index 4920h, sub 3):

Source	PDO (COB-ID)	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
MDT	180h + *NodeID*	00h +DigIn	00h	00h	00h	00h

If the JTAG-action #2 storage is empty or another JTAG-action is in progress (being shifted into JTAG), the MDT-DCS module sends the following SDO *Abort Transfer* reply:

Source	SDO	Byte 4	Byte 5	Byte 6	Byte 7
MDT	Abort OD 4927h, 0	00h	00h	1 (code): Access	6 (class): Access

as well as the following Emergency message (see section 10):

Source	Emergency (COB-ID)	Byte 0-1	Byte 2	Byte 3-7
MDT	080h + *NodeID*	Emergency Error Code (00h 50h)	Error Register (Object 1001h)	Manufacturer specific error field (81h 02h 00h 00h 00h)

If the stored JTAG-action is invalid (i.e. a CRC does not match the corresponding stored bit string) the MDT-DCS module sends the following SDO *Abort Transfer* reply:

Source	SDO	Byte 4	Byte 5	Byte 6	Byte 7
MDT	Abort OD 4927h, 0	00h	00h	6 (code): Hardware	6 (class): Access

as well as the following Emergency message (see section 10):

Source	Emergency (COB-ID)	Byte 0-1	Byte 2	Byte 3-7
MDT	080h + *NodeID*	Emergency Error Code (00h 50h)	Error Register (Object 1001h)	Manufacturer specific error field (82h 02h 01h 00h 00h)

5. Loading JTAG instruction and data bit strings from JTAG-action #2 storage, followed by JTAG-action #8 and #6 to one CSM, using (a) PDO message(s).
The host sends the following message:

Source	PDO (COB-ID)	Byte 0	Byte 1	Byte 2	Byte 3
host	400h + *NodeID*	FEh	02h	08h	06h

Value FEh in byte 0 of the PDO signifies to the MDT-DCS module that the numbers following are the indices of JTAG-actions to execute in the order they appear in the message. (Compare to example 2 where PDO 400h+*NodeID* is used for its other purpose: data bit string upload).

The PDO, when used for triggering JTAG-action uploads, may be any length (from 2 up to 8 bytes), which means up to 7 JTAG actions may be executed in sequence, triggered by one such PDO message. *(In future versions this number may be extended to 14 JTAG-actions, if just 4 bits per JTAG-action number are assigned*). Any number of bytes may be present in the PDO, even more than the number of JTAG-action to execute, but then the last JTAG-action number to be executed must be followed by a zero. The following PDO does exactly the same as the one above:

Source	PDO (COB-ID)	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
host	400h + *NodeID*	FEh	02h	08h	06h	00h

Note that there is no direct reply to the PDO message from the host.
However, if the global enable of reporting the JTAG-action status error has been set (*OD* index 49F1h set to 1), and the MDT-DCS module is in state *Operational*, the completion of each JTAG-action will result in the MDT-DCS module sending its Digital Input PDO message, in which there are 4 bytes containing the status error word (*OD* index 49F0h, sub 3). In this PDO message bit 7 of byte 0 is a toggle bit that is toggled after every JTAG-action completion, and byte 1 to 4 contains the status error word resulting from the JTAG-action. When all returned status error words are zero the sequence of JTAG-actions was successfully executed.
The resulting message sequence would look like this:

Source	PDO (COB-ID)	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
host	400h + *NodeID*	FEh	02h	08h	06h	-
MDT	180h + *NodeID*	00h +DigIn	00h	00h	00h	00h
MDT	180h + *NodeID*	80h +DigIn	00h	00h	00h	00h
MDT	180h + *NodeID*	00h +DigIn	00h	00h	00h	00h

Now if the host would like to wait for each JTAG-action to complete it would trigger the upload of only one JTAG-action at a time and wait for the Digital Input PDO message, with the toggle bit toggled! In that case the message sequence might look like this, with alternating host and MDT-DCS messages:

Source	PDO (COB-ID)	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
host	400h + *NodeID*	FEh	02h	-	-	-
MDT	180h + *NodeID*	00h +DigIn	00h	00h	00h	00h
host	400h + *NodeID*	FEh	06h	-	-	-
MDT	180h + *NodeID*	80h +DigIn	00h	00h	00h	00h
host	400h + *NodeID*	FEh	08h	-	-	-
MDT	180h + *NodeID*	00h +DigIn	00h	00h	00h	00h

As mentioned before, the PDO sent by the host may be any length from 2 up to 8 bytes, but in this case bytes 2 to 7 (if present) should be zero. A new sequence can only be started if the previous one has finished; if a sequence is in progress, the PDO sent by the host is simply ignored, with one exception: to abort an ongoing sequence, the host may sent the PDO containing only byte 0 (with value FEh).

Note that if the status mask of a JTAG-action is set to 0 (*OD* index 49X8h, sub 2), there is actually no 'return status bits check' taking place. In that case the PDO message sent by the MDT-DCS module just serves to signify completion of that JTAG-action and the status error word returned is always 0.

Any problems detected with the JTAG-action's stored bit strings are reported by Emergency messages as shown in example 4.

6. **Loading JTAG instruction and data bit strings from JTAG-action #2 storage, followed by JTAG-action #8 and #6 to <u>all</u> CSMs connected to the CAN-bus using (a) PDO message(s).**
   The host sends the following message:

Source	PDO (COB-ID)	Byte 0	Byte 1	Byte 2	Byte 3
host	580h	FEh	02h	08h	06h

The same options for getting a PDO reply from the MDT-DCS modules and controlling the JTAG-actions sequence as described in the previous example are valid here, but take into account that now <u>each</u> MDT-DCS module on the CAN-bus may send a reply !

7. **Writing/replacing a JTAG data bit string in JTAG-action #2 storage, using SDO messages.**
   The bit string from example 1 is written:

Source	SDO	Byte 4	Byte 5	Byte 6	Byte 7
host	Write OD 4926h, 0	00h	–	–	–
host	Write OD 4923h, 0	10h	32h	54h	76h
host	Write OD 4923h, 0	98h	BAh	DCh	FEh
host	Write OD 4924h, 4	0Ah	00h	00h	00h

Note that basically the only difference with the direct JTAG bit string upload in example 1 is a write access to *OD* index 4926, to make sure the next bit string write operation starts at the first bit of storage. The first write to storage immediately invalidates any previously stored bit string. Non-significant bits in the last message *must* be set to zero.
It is the responsibility of the host to make sure instruction and data bit string form a valid JTAG-action from the CSM's point of view. Note that the data is received by one and only one MDT-DCS module (according to the *NodeID* set in the SDO CAN-message sent by the host).

If the JTAG-action #2 storage is full, the MDT-DCS module sends the following SDO *Abort Transfer* reply:

Source	SDO	Byte 4	Byte 5	Byte 6	Byte 7
MDT	Abort OD 4923/4h, 0	00h	00h	1 (code): Access	6 (class): Access

If an error occurs during writing of the JTAG-action #2 storage, the MDT-DCS module sends the following SDO *Abort Transfer* reply:

Source	SDO	Byte 4	Byte 5	Byte 6	Byte 7
MDT	Abort OD 4923/4h, 0	00h	00h	6 (code): Hardware	6 (class): Access

8. Reading the JTAG data bit string from JTAG-action #2 storage, using SDO messages.

Source	SDO	Byte 4	Byte 5	Byte 6	Byte 7
host	Write OD 4926h, 0	00h	–	–	–
MDT	Read OD 4923h, 0	10h	32h	54h	76h
MDT	Read OD 4923h, 0	98h	BAh	DCh	Feh
MDT	Read OD 4923h, 0	0Ah	00h	00h	00h

The MDT-DCS module's SDO replies in the above table are in response to SDO read messages from the host system, which are not shown here. Note that the data is received by one and only one MDT-DCS module (according to the *NodeID* set in the SDO CAN-messages sent by the host).

Since all bits of the string have been read after the 3rd SDO read operation, any subsequent SDO read request by the host will result in an SDO *Abort Transfer* reply, as follows:

Source	SDO	Byte 4	Byte 5	Byte 6	Byte 7
MDT	Abort OD 4923h, 0	00h	00h	1 (code): Access	6 (class): Access

If the #2 storage is empty the same SDO *Abort Transfer* reply results at the first read operation attempted.

If the #2 storage has a CRC error the MDT-DCS module sends the following SDO *Abort Transfer* reply:

Source	SDO	Byte 4	Byte 5	Byte 6	Byte 7
MDT	Abort OD 4923h, 0	00h	00h	6 (code): Hardware	6 (class): Access

6.2.1.4 JTAG TAP States

The JTAG TAP states are defined in the MDT-DCS module by an identifier value, as listed in Table 10; the JTAG state transition diagram with TAP states is shown next to the table on the right. The TAP state can be read from or -if required- set by a host application through *OD* index 4830h. By writing to *OD* index 4840h a JTAG TAP reset and subsequent transition to state *Run-Test/Idle* may be triggered.

TAP State	Identifier
SELECT_DR_SCAN	0
CAPTURE_DR	1
SHIFT_DR	2
EXIT1_DR	3
PAUSE_DR	4
EXIT2_DR	5
UPDATE_DR	6
TEST_LOGIC_RESET	7
RUN_TEST_IDLE	8
SELECT_IR_SCAN	9
CAPTURE_IR	10
SHIFT_IR	11
EXIT1_IR	12
PAUSE_IR	13
EXIT2_IR	14
UPDATE_IR	15

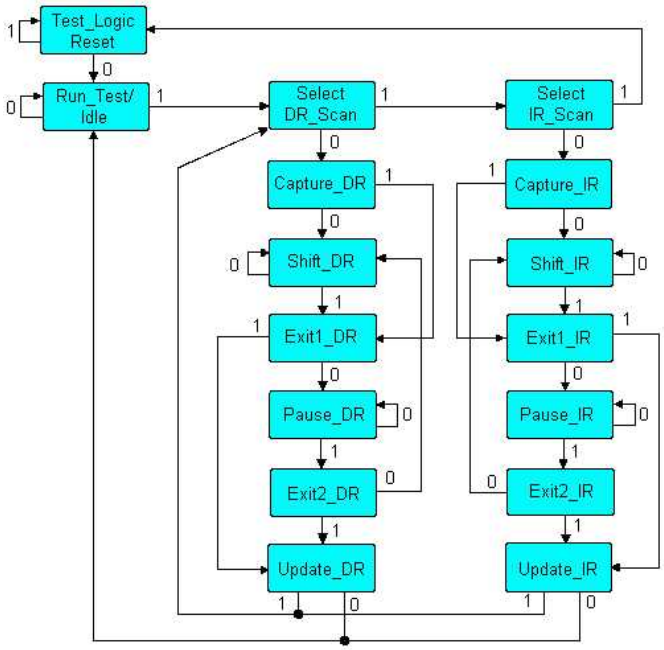


Table 10. JTAG TAP State identifiers in the MDT-DCS module and the JTAG state transition diagram (right) with '0' and '1' indicating the value of TMS during the TCK-controlled state transition.

6.2.1.5 JTAG Signal Timing

OD index 4860h can be used to control the period of the JTAG TCK clock signal, or actually the time the TCK signal is high, and the *minimum* time TCK is low (low time is much longer due to software overhead; for instance: shifting out 32 bits, and reading 32 return bits at the same time, in total roughly takes about 400 µs for *OD* index 4860h equal to 0, and about 500 µs for *OD* index 4860h equal to 3).

OD index 4860h setting	TCK high [µs]
0	*1.5*
1	*2.0*
2	*2.5*
3	*3.0*

Table 11. JTAG TCK cycle period options.

6.2.1.6 Additional JTAG Functionality

By writing number *n* to *OD* index 4850h, *n* cycles of TCK are generated with the TAP state remaining unchanged. This is only possible with the TAP in one of the states *Test-Logic-Reset*, *Run-Test/Idle*, *Shift-IR*, *Shift-DR*, *Pause-IR* or *Pause-DR*. If required, the TAPs can be set to the required state as described in section 6.2.1.4. The TAP state after string uploads and JTAG actions is *Run-Test/Idle*.

Reading *OD* index 4850h provides the number of cycles still to be generated. Cycles are generated in bursts of 32 (in between MDT-DCS performs other tasks). Any write access to a JTAG object aborts an ongoing TCK cycle sequence.

By reading *OD* index 4870h the number of TAPs in the JTAG chain are counted. This is achieved by loading the *BYPASS* instruction in all instruction registers. This feature may be useful for testing JTAG chain integrity.

6.2.2 Digital I/O

The MDT-DCS module has a total of 7 Digital I/Os, numbered from 1 to 7, of which number 1 to 4 are to be found on the *JTAG* connector and numbers 5 to 7 on the *CSM-ADC* connector. See Table 2 and Table 4 in section 2.1 for the mapping of connector pin to Digital I/O number.

Digital inputs can be read out using the ***PDO*** mechanism. The CAN-identifier used for this PDO is the *1st-Transmit-PDO* (see *OD* index 1800h and 1A00h for configuration and mapping resp.). In this application the PDO message contains 1 data byte containing the state of up to 8 digital inputs (7 bits significant). The message also carries the JTAG-action status error information:

MDT-DCS module → Host

COB-ID	Data Byte 0	Data Byte 1-4
180h + *NodeID*	8-bit Digital Input	JTAG-action status error

Note that the 8-bit digital port is shared between digital inputs and outputs. Whether an I/O-line is used as input or output is set through *OD* index 6208h, subindex 1. An I/O-line defined as output shows up as a zero in a digital input read operation.
On the MDT-DCS module bits 1-4 are defined as outputs and bits 5-7 as inputs, by default.

The following modes of TPDO1 transmission are supported (see *OD* index 1800h, subindex 2 and 5):

- ***PDO* transmission type = 1:**
  after every socalled **SYNC** message issued on the CAN-bus the MDT-DCS module sends the PDO message as well as *on change*.
- ***PDO* transmission type = 255**:
  the MDT-DCS module sends the PDO message *on change*.

- **Event Timer > 0**:
  if the PDO's *event timer* (*OD* index 1804h, subindex 5) is set to a value unequal to zero (*event timer* is expressed in units of 1 s, <255 s) the MDT-DCS module automatically sends the PDO message periodically, triggered by a timer, as well as *on change*.

The transmission of the PDO *on change* only occurs if this feature has been enabled globally (*OD* index 6005h; default is *not* enabled) and per digital input (*OD* index 6006h; default is enabled). Once the MDT-DCS module is put into state *Operational,* it continuously monitors the state of the digital inputs and immediately sends the PDO message if it detects a level change of any of the inputs. A kind of debounce time-out is in effect and can be set (also to zero) by writing to *OD* index 2200h.

The digital inputs can of course also be read out using CANopen **SDO** messages by reading from *OD* index 6000h.

Digital outputs can be set using the ***PDO*** mechanism. The CAN-identifier used for this PDO is the *1st-Receive-PDO* (*OD* index 1400h and 1600h for configuration and mapping resp.). In this application the PDO message contains 1 or 2 data bytes containing the setting for one set of 8 digital outputs (7 bits significant) or a single bit (in case of the 2-byte PDO):

Host → MDT-DCS module

COB-ID	Data Byte 0
200h + *NodeID*	8-bit Digital Output

or

Host → MDT-DCS module

COB-ID	Data Byte 0	Data Byte 1
200h + *NodeID*	Dig Out Number (1-7)	0 or 1

Note that the digital port is shared between digital inputs and outputs. Whether an I/O-line is used as input or output is set by *OD* index 6208h, subindex 1. In the default setting only Digital Out 1, 2, 3 and 4 are available.

Digital outputs can of course also be set using ***SDO*** messages by writing to *OD* index 6200h (all digital outputs in one 8-bit parameter) or to *OD* index 6220h (digital outputs individually).

At power-up a digital output is initialized to either low or high, which can be configured for each bit individually in *OD* index 2F00h.

7 Configuration Storage

7.1 Storing Parameters and Settings

Parameters and settings can be stored permanently onboard in non-volatile memory (EEPROM) by writing string "save" to *OD* index 1010h. The ***SDO*** mechanism is used to accomplish this, shown here:

Host → MDT-DCS module

COB-ID	**Data Byte**							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
600h + *NodeID*	0x23	0x10	0x10	*subindex*	73h ('s')	61h ('a')	76h ('v')	65h ('e')

with *OD* index 1010h in byte 1+2 and *subindex* in byte 3 with *subindex*:
= 1: store all parameters (as listed for *subindex* 2 and 3).
= 2: store communication parameters (concerning CAN, PDOs and Node- and Life Guarding).
= 3: store application parameters (concerning ADCs, Digital I/O and JTAG).
= 4: see next section.

If the store-operation succeeded the MDT-DCS module sends the following reply:

MDT-DCS module → Host

COB-ID	**Data Byte**						
	0	**1**	**2**	**3**	**4**	**5**	**6-7**
580h + *NodeID*	0x60	0x10	0x10	*subindex*	–	–	–

If the store-operation did <u>*not*</u> succeed the MDT-DCS module sends the following reply (***SDO** Abort Domain Transfer*, error reason: 'hardware fault' (for more details see [2])):

MDT-DCS module → Host

COB-ID	**Data Byte**							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
580h + *NodeID*	80h	10h	10h	*subindex*	0	0	6 (Error Code)	6 (Error Class)

Parameters can be reset to their default values (by invalidating the corresponding contents of the EEPROM) by writing to *OD* index 1011h, using this time the string "load" (6Ch, 6Fh, 61h, 64h) in bytes 4 to 7 of the ***SDO***. Note that the default values take effect only after a subsequent reset of the node. The default parameter values are listed in the *OD* tables in section 9.

The Object Dictionary tables in section 9 show which settings can be stored in EEPROM: these are marked by an asterisk (*) in the first column
(Note that storage of ADC calibration constants, the ELMB Serial Number and JTAG strings for MDT front-end electronics configuration are handled separately).

7.2 Auto-configure

Starting with MDT-DCS firmware version 2.4.0, a so-called *autoconfigure* capability was added. What it does is that the MDT-DCS module determines itself how and how many B-sensor modules are connected to it (see section 5.3), and also if there is a connection to an ADC on the CSM (for frontend monitoring, see section 6.1). The configuration found is stored in EEPROM and used after subsequent power-up and resets.

An autoconfigure sequence is initiated by writing string "save" to *OD* index 1010h, sub-index 4, using an ***SDO*** message, as described in the previous section.

7.3 EEPROM Memory Map

Table 12 and Table 13 below detail the layout of the ELMB's EEPROM usage by the MDT-DCS application firmware.

EEPROM	ADDR	DESCRIPTION
not used	0000h	
MDT-DCS configuration parameters	0001h 00A0h	Holds permanently saved application configuration and settings, stored in up to 8 blocks of up to 16 bytes each; includes a CRC checksum for each data block.
Rad-tolerant working copy of global settings and parameters	00A1h 00FEh	Holds a copy of most application configuration and settings and some other parameters that don't change very often; parameters are reread from EEPROM each time before being used; this is an optional feature to counter the effects of SEE (Single Event Upset).
not used	00FFh	
ELMB Serial Number	0100h 0106h	Holds the ELMB Serial Number given to it at production time; serves to uniquely identify the ELMB and retrieve its calibration constants and/or production data in the ELMB production database.
Node-ID (opt)	0107h	The 'Node-ID' location may optionally contain a CAN Node-ID for the module, replacing the DIP-switch setting; if the location contains a valid number (0<=val<=127) it *must* be used.
not used	011F	
ELMB Analog-in calib consts	0120h 01CFh	Holds the calibration constants, which were determined at production time, for all 6 voltage ranges (note: only present for ELMBs with an analog input part).
not used	01E0h 01FFh	
Storage space for JTAG strings (see Table 13 for details)	0200h 0FFFh	Space to store JTAG strings: up to 3584 bytes. Up to 3 spaces of 6400 bits each: 3*800=2400 bytes. Up to 10 spaces of 640 bits each: 10*80=800 bytes. (one 'space' is for a IR + DR string: one *JTAG-action*) For administration (per string): 2*(3+10)*4=104 bytes (16-bits number of bits, 16-bits CRC). For status info (per action): (3+10)*(3*4)=156 bytes (32-bits each: start bit, status mask, status expected) *Total: 2400 + 800 + 104 + 156 = 3460 bytes used.*

Table 12. EEPROM memory map of the MDT-DCS ELMB application firmware.

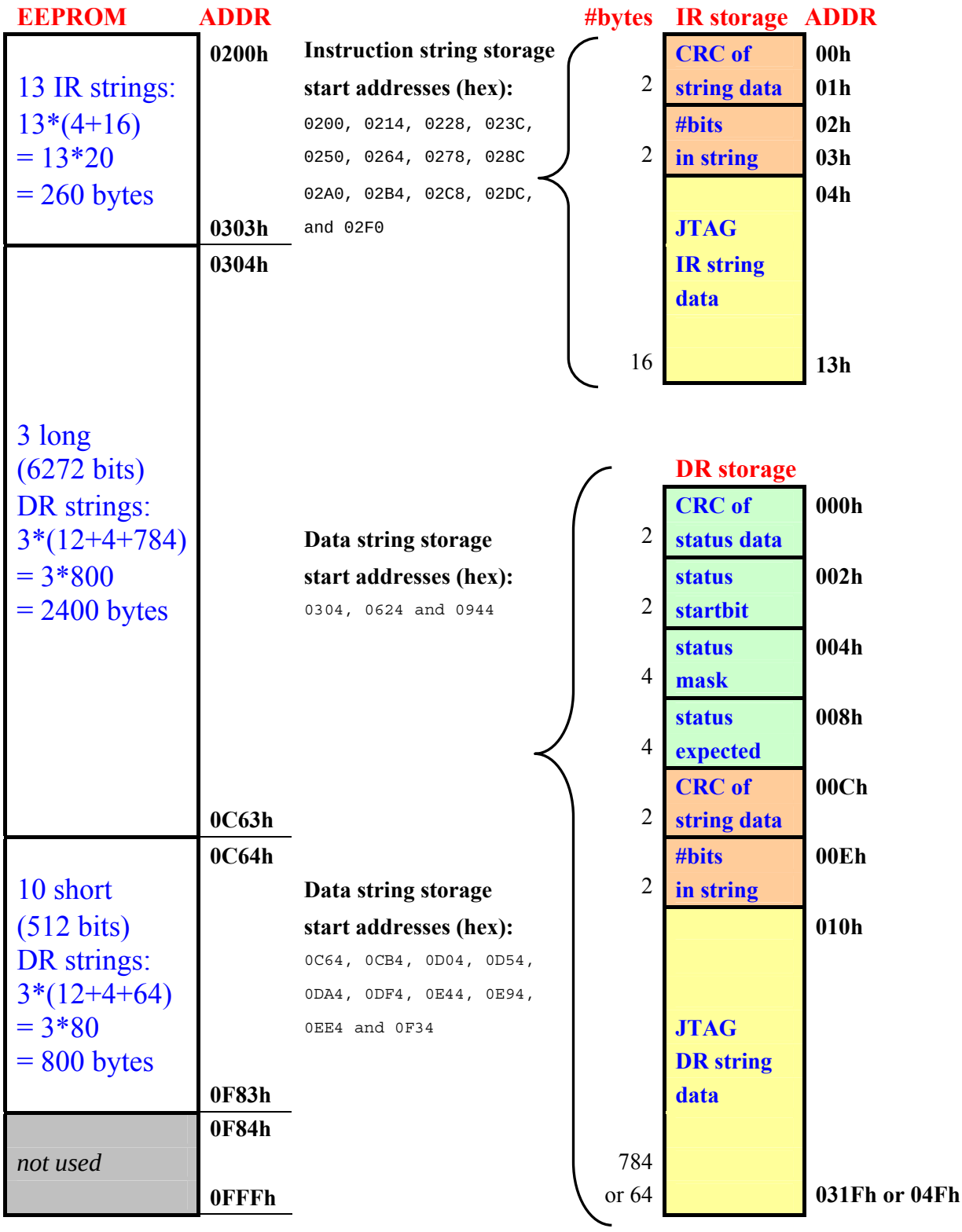


Table 13. EEPROM memory map of the MDT-DCS module JTAG strings storage space (DR = Data Register, IR = Instruction Register).

8 Upgrading the Firmware

The application program in the MDT-DCS (ELMB) microcontroller can be replaced or upgraded by uploading new program code to the MDT-DCS node via the CAN-bus.
On the ELMB resources webpage [6] PC-tools called **ELMBmldr** (with command-line interface) and **ELMBloader** (with graphical user-interface) can be found, to perform a firmware upgrade. The upgrade process leaves the EEPROM intact, in other words: configuration settings are preserved.

The ***Bootloader*** [5] is an application program stored in a separate section of the ELMB microcontroller (the ELMB comes preinstalled with this application). It handles the firmware upgrade process on the ELMB, receiving series of CAN(open) messages containing the programming instructions.
At power-up of the MDT-DCS module it is initially the *Bootloader*, which is in control of the module. After about 4 s the *Bootloader* automatically jumps to the start of the MDT-DCS application program, or immediately, when it receives a CANopen ***NMT** Reset-Node* message. However, the *Bootloader* remains in control if it receives a valid programming command within those 4 s. The firmware upgrade process may then begin.

The MDT-DCS application program can transfer control of the module explicitly to the *Bootloader* by writing any value to the 8-bit object 5E00h in the Object Dictionary of the MDT-DCS application. In this case the *Bootloader* does *not* automatically jump back to the MDT-DCS application program after 4 s. The firmware upgrade process may now begin.

After the upgrade process, the reception of a CANopen ***NMT** Reset-Node* message causes the *Bootloader* to jump to the start of the new MDT-DCS application program.

If the MDT-DCS module sends an *Emergency* message as shown below, it signifies that the *Bootloader* is in control of the module. Note that the same Emergency message is also sent as the first message after power-up, when the *Bootloader* is in control for the first 4 s after power-up, before jumping to the application program. The *Bootloader* can be forced to jump to the application immediately, by sending it a CANopen ***NMT*** Reset-Node message.

COB-ID	Byte 0-1	Byte 2	Byte 3-7
080h + *NodeID*	Emergency Error Code (00h 50h)	Error Register (Object 1001h) (80h)	Manufacturer specific error field (5 bytes: FEh,01h,28h,*ZZh*,00h, with *ZZh* = MCUCSR)

(*MCUCSR* = MCU Control and Status Register contents; for details see section 10).

9 MDT-DCS Object Dictionary

The values of objects marked with ∗ in the *Index* column can be stored permanently in EEPROM. They are retrieved from EEPROM at reset and power-up.

Communication Profile Area (*MDT-DCS*)						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
1000	-	Device type	U32	RO	00070191h	Meaning: DSP-401 device profile, analogue inputs, digital in- and outputs on device
1001	-	Error register	U8	RO	0	
1002	-	Manufacturer status reg	U32	RO	0	[1] (see footnote)
1008	-	Manufacturer device name	VisStr	RO	"ELMB"	= Embedded Local Monitor Board
1009	-	Manufacturer hw version	VisStr	RO	"el40"	= ELMB v4
100A	0	Manufacturer software version	VisStr	RO	"MD24"	***MDT-DCS*** application v2.4.0
	1	minor version number	VisStr	RO	"0000"	
100C	-	Guard time [ms]	U16	RO	1000	= 1 second
100D *	-	Life time factor	U8	RW	0	Life Guarding timeout in seconds; 0 → no life guarding timeout
1010		Store parameters	Array			Save stuff in onboard EEPROM
	0	Highest index supported	U8	RO	3	
	1	Save all parameters	U32	RW	1	Read: 1; Write "save": store all
	2	Save communication parameters	U32	RW	1	Read: 1; Write "save": store PDO par's, Life time factor, …
	3	Save application par's	U32	RW	1	Read: 1; Write "save": store ADCs config, Dig.I/O config, …
	4	Detect B-sensors and CSM-ADC and save configuration found	U32	RW	1	Read: 1; Write "save": store B-sensor and CSM ADC config (so-called 'autoconfigure')
1011		Restore default parameters	Array			Invalidate stuff in onboard EEPROM; use defaults
	0	Highest index supported	U8	RO	3	
	1	Restore all parameters	U32	RW	1	Read: 1; Write "load": invalidate all parameters stored
	2	Restore communication parameters	U32	RW	1	Read: 1; Write "load": invalidate stored PDO par's, etc.
	3	Restore application par's	U32	RW	1	Read: 1; Write "load": invalidate stored ADCs config, etc.
1017 *	-	Producer Heartbeat Time [1 s]	U16	RW	0	In units of seconds (but <=255 !), (NB: actually should be in ms according to CANopen!); 0 → Heartbeat is disabled

[1] Manufacturer Status Register: byte0 = **NTC-ADC**, byte1 = **B0/B1-ADC**, byte2 = **B2/B3-ADC**, byte3 = **CSM-ADC**. Status byte/nibble: **01**: ADC reset error, **02**: ADC calibration error, **04**: ADC conversion time-out, **FF**: ADC absent / not used.

Communication Profile Area (*MDT-DCS*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
1018		Identity	Record			Mandatory CANopen object
	0	Number of entries	1..4	RO	1	
	1	Vendor ID	U32	RO	12345678h	*to be ordered from CiA*
1400		1st Receive PDO par's	Record			Data type = PDOCommPar
	0	Number of entries	U8	RO	5	
	1	COB-ID used by PDO	U32	RO	200h + *NodeID*	According to CANopen Predefined Connection Set
	2	Transmission type	U8	RO	255	
	3,4,5	*Not used*		RO	0	
1401		2nd Receive PDO par's	Record			Data type = PDOCommPar
	0	Number of entries	U8	RO	5	
	1	COB-ID used by PDO	U32	RO	300h + *NodeID*	According to CANopen Predefined Connection Set
	2	Transmission type	U8	RO	255	
	3,4,5	*Not used*		RO	0	
1402		3rd Receive PDO par's	Record			Data type = PDOCommPar
	0	Number of entries	U8	RO	5	
	1	COB-ID used by PDO	U32	RO	400h + *NodeID*	According to CANopen Predefined Connection Set
	2	Transmission type	U8	RO	255	
	3,4,5	*Not used*		RO	0	
1403		4th Receive PDO par's	Record			Data type = PDOCommPar
	0	Number of entries	U8	RO	5	
	1	COB-ID used by PDO	U32	RO	500h	
	2	Transmission type	U8	RO	255	
	3,4,5	*Not used*		RO	0	
1404		5th Receive PDO par's	Record			Data type = PDOCommPar
	0	Number of entries	U8	RO	5	
	1	COB-ID used by PDO	U32	RO	580h	
	2	Transmission type	U8	RO	255	
	3,4,5	*Not used*		RO	0	

Communication Profile Area (*MDT-DCS*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
1600		1st Receive PDO mapping	Record			Data type = PDOMapping
	0	Number of entries	U8	RO	1	
	1	Digital outputs 1-8	U32	RO	62000108h	OD-index 6200, sub-index 1: Outputs 1-8 (see DSP-401), size = 8 bits
1601		2nd Receive PDO mapping	Record			Data type = PDOMapping
	0	Number of entries	U8	RO	2	
	1	Number of JTAG IR bits	U32	RO	48010008h	OD-index 4801, sub-index 0: final number of JTAG IR bits, size = 8 bits
	2	JTAG IR bits	U32	RO	48000020h	OD-index 4801, sub-index 1: JTAG IR bits, size = 32 bits
1602		3rd Receive PDO mapping	Record			Data type = PDOMapping
	0	Number of entries	U8	RO	2	
	1	Number of JTAG DR bits	U32	RO	48040008h	OD-index 4804, sub-index 0: final number of JTAG DR bits, size = 8 bits
	2	JTAG DR bits	U32	RO	48030020h	OD-index 4803, sub-index 1: JTAG DR bits, size = 32 bits
1603		4th Receive PDO mapping	Record			idem Object 1601
1604		5th Receive PDO mapping	Record			idem Object 1602

Communication Profile Area (*MDT-DCS*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
1800		1st Transmit PDO par's	Record			Data type = PDOCommPar
	0	Number of entries	U8	RO	5	
	1	COB-ID used by PDO	U32	RO	180h + *NodeID*	According to CANopen Predefined Connection Set
*	2	Transmission type	U8	RW	255	Only 1 and 255 allowed
	3	Inhibit time [100 µs]	U16	RO	0	*not used*
	4	*Not used*	U8	RO	0	
*	5	Event timer [1 s]	U16	RW	0	In units of secs, must be <= 255; active for all transmission-types!
1801		2nd Transmit PDO par's	Record			Data type = PDOCommPar
	0	Number of entries	U8	RO	5	
	1	COB-ID used by PDO	U32	RO	280h + *NodeID*	According to CANopen Predefined Connection Set
*	2	Transmission type	U8	RW	1	Only 1 and 255 allowed
	3	Inhibit time [100 µs]	U16	RO	0	*not used*
	4	*Not used*	U8	RO	0	
*	5	Event timer [1 s]	U16	RW	0	In units of secs, must be <= 255; active for all transmission-types!
1802		3rd Transmit PDO par's	Record			Data type = PDOCommPar
	0	Number of entries	U8	RO	5	
	1	COB-ID used by PDO	U32	RO	380h + *NodeID*	According to CANopen Predefined Connection Set
*	2	Transmission type	U8	RW	1	Only 1 and 255 allowed
	3	Inhibit time [100 µs]	U16	RO	0	*not used*
	4	*Not used*	U8	RO	0	
*	5	Event timer [1 s]	U16	RW	0	In units of secs, must be <= 255; active for all transmission-types!
1803		4th Transmit PDO par's	Record			Data type = PDOCommPar
	0	Number of entries	U8	RO	5	
	1	COB-ID used by PDO	U32	RO	480h + *NodeID*	According to CANopen Predefined Connection Set
*	2	Transmission type	U8	RW	1	Only 1 and 255 allowed
	3	Inhibit time [100 µs]	U16	RO	0	*not used*
	4	*Not used*	U8	RO	0	
*	5	Event timer [1 s]	U16	RW	0	In units of secs, must be <= 255; active for all transmission-types!

Communication Profile Area (*MDT-DCS*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
1A00		1st Transmit PDO mapping	Record			Data type = PDOMapping
	0	Number of entries	U8	RO	2	
	1	Digital inputs 1-8	U32	RO	60000108h	OD-index 6000, sub-index 1: Inputs 1-8 (see DSP-401), size = 8 bits
	2	JTAG status return bits error	U32	RO	49F00320h	OD-index 49F0, sub-index 3: size = 32 bits
1A01		2nd Transmit PDO mapping	Record			Data type = PDOMapping
	0	Number of entries	U8	RO	2	*should be 255 for MuxPDO, but this is not a CANopen MPDO…*
	1	NTC number	U32	RO	40000008h	*actually not allowed, but…*
	2	24-bit analogue input + stat	U32	RO	40000*x*18h	OD-index 4000, sub-index *x*: Analogue inputs, multiplexed, size = 24 bits
1A02		3rd Transmit PDO mapping	Record			Data type = PDOMapping
	0	Number of entries	U8	RO	2	*should be 255 for MuxPDO, but this is not a CANopen MPDO…*
	1	CSM ADC channel no	U32	RO	41000008h	*actually not allowed, but…*
	2	24-bit analogue input + stat	U32	RO	41000*x*18h	OD-index 4100, sub-index *x*: Analogue inputs, multiplexed, size = 24 bits
1A03		4th Transmit PDO mapping	Record			Data type = PDOMapping
	0	Number of entries	U8	RO	2	*should be 255 for MuxPDO, but this is not a CANopen MPDO…*
	1	B-sensor ADC channel number	U32	RO	42000008h	*actually not allowed, but…*
	2	24-bit analogue input	U32	RO	420*x*0*x*20h	OD-index 4200/4201,subindex *x*, Analogue inputs, multiplexed, size = 32 bits

Manufacturer-specific Profile Area (*MDT-DCS*)						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
2100		ADC-configuration NTC	Record			CRYSTAL CS5523 16-bit ADC
	0	Number of entries	U8	RO	22	
*	1	Number of input channels	U8	RW	60	64 maximum; can be set to actual number of channels used (2 channels per NTC; last 2 NTCs are calib/reference inputs)
*	2	Conversion Word Rate	U8	RW	0	3-bit code [1]
*	3	Input Voltage Range	U8	RW	5	3-bit code [2]
*	4	Unipolar/Bipolar Measurement Mode	U8	RW	1	0 = bipolar, 1 = unipolar
	5	Power Save Mode	Bool	WO		1 = set ADC to power save mode 0 = take ADC out of this mode
	6	Configuration Register	U32	RW		CS5523 Config Register
	7	Offset Register #1	U32	RW		CS5523 physical channel AIN1
	8	Gain Register #1	U32	RW		CS5523 physical channel AIN1
	9	Offset Register #2	U32	RW		CS5523 physical channel AIN2
	10	Gain Register #2	U32	RW		CS5523 physical channel AIN2
	11	Offset Register #3	U32	RW		CS5523 physical channel AIN3
	12	Gain Register #3	U32	RW		CS5523 physical channel AIN3
	13	Offset Register #4	U32	RW		CS5523 physical channel AIN4
	14	Gain Register #4	U32	RW		CS5523 physical channel AIN4
	15	Channel-Setup Register #1	U32	RW		LC 1 (12-bits) in lower 2 bytes, LC 2 (12-bits) in upper 2 bytes
	16	Channel-Setup Register #2	U32	RW		LC 3 (12-bits) in lower 2 bytes, LC 4 (12-bits) in upper 2 bytes
	17	Channel-Setup Register #3	U32	RW		LC 5 (12-bits) in lower 2 bytes, LC 6 (12-bits) in upper 2 bytes
	18	Channel-Setup Register #4	U32	RW		LC 7 (12-bits) in lower 2 bytes, LC 8 (12-bits) in upper 2 bytes
	19	Conversion Word Rate	U8	RO	15	in Hz
	20	Input Voltage Range	U32	RO	2500000	in µV
*	21	SPI SCLK signal high period (opto-coupler delay)	U8	RW	75	in µs, 10 <= value <= 255
*	22	Delta value for automatic on-change readout	U32	RW	0	in currently used units (millidegrees, Ohms or ADC-counts); 0 = readout-on-change disabled; works in combination with the PDO timer (Object 1801h, sub 5)

[1] **000**: 15.0 Hz, **001**: 30.0 Hz, **010**: 61.6 Hz, **011**: 84.5 Hz,
100: 101.1 Hz, **101**: 1.88Hz, **110**: 3.76 Hz, **111**: 7.51 Hz

[2] **000**: 100 mV, **001**: 55 mV, **010**: 25 mV, **011**: 1 V, **100**: 5 V, **101**: 2.5 V

Manufacturer-specific Profile Area (*MDT-DCS*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
2101		ADC-configuration CSM	Record			CRYSTAL CS5523 16-bit ADC
	0	Number of entries	U8	RO	21	
*	1	Number of input channels	U8	RW	64	64 maximum; can be set to actual number of channels used
*	2	Conversion Word Rate	U8	RW	0	3-bit code [1]
*	3	Input Voltage Range	U8	RW	4	3-bit code [2]
*	4	Unipolar/Bipolar Measurement Mode	U8	RW	0	0 = bipolar, 1 = unipolar; default changed from 1 to 0 in version **2.3.2**
	etc.	*…as above…*	…	…	…	
*	19	2.5V ref input channel	U8	RW	11	if >=64 the ADC calibration procedure done is a socalled 'self-calibration' (using an ADC internal reference), which is inaccurate for all ADC voltage ranges except 2.5V; changed from 255 to 11 in version **2.3.2**
*	20	SPI SCLK signal high period (opto-coupler delay)	U8	RW	75	in µs, 10 <= value <= 255
*	21	Delta value for automatic on-change readout	U32	RW	0	in currently used units (millidegrees, Ohms or ADC-counts); 0 = readout-on-change disabled; works in combination with the PDO timer (Object 1802h, sub 5)
2200	-	ADC-reset-and-calibrate NTC	U8	WO		Writing triggers a reset and calibration sequence with the current NTC-ADC settings
2201	-	ADC-reset-and-calibrate CSM	U8	WO		Writing triggers a reset and calibration sequence with the current CSM-ADC settings
2300 *	-	ADC-reset-and-calibrate before each channel scan NTC	U8	RW	0	If =1 a reset/calibration sequence is performed before every NTC-ADC input channel scan
2301 *	-	ADC-reset-and-calibrate before each channel scan CSM	U8	RW	0	If =1 a reset/calibration sequence is performed before every CSM-ADC input channel scan
2400 *	-	ADC enabled NTC	U8	RW	1	Set to 0 if the ADC is not used or not present
2401 *	-	ADC enabled CSM	U8	RW	0	Set to 0 if the ADC is not used or not present

[1] **000**: 15.0 Hz, **001**: 30.0 Hz, **010**: 61.6 Hz, **011**: 84.5 Hz, **100**: 101.1 Hz, **101**: 1.88Hz, **110**: 3.76 Hz, **111**: 7.51 Hz

[2] **000**: 100 mV, **001**: 55 mV, **010**: 25 mV, **011**: 1 V, **100**: 5 V, **101**: 2.5 V

Manufacturer-specific Profile Area (*MDT-DCS*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
2500		B-sensor #0 ADC-config	Record			CRYSTAL CS5524 24-bit ADC [1]
	0	Number of entries	U8	RO	22	
	1	Number of input channels	U8	RO	7	
*	2	Conversion Word Rate Hall	U8	RW	0	3-bit code [2]
*	3	Input Voltage Range Hall	U8	RW	0	3-bit code [3]
*	4	Unipolar/Bipolar Measurement Mode Hall	U8	RW	0	0 = bipolar, 1 = unipolar
*	5	Conversion Word Rate Temp	U8	RW	0	3-bit code [2]
*	6	Input Voltage Range Temp	U8	RW	5	3-bit code [3]
*	7	Unipolar/Bipolar Measurement Mode Temp	U8	RW	1	0 = bipolar, 1 = unipolar
	8	Power Save Mode	Bool	WO		1 = set ADC to power save mode 0 = take ADC out of this mode
	9	Configuration Register	U32	RW		CS5523 Config Register
	10	Offset Register #1	U32	RW		CS5523 physical channel AIN1
	11	Gain Register #1	U32	RW		CS5523 physical channel AIN1
	12	Offset Register #2	U32	RW		CS5523 physical channel AIN2
	13	Gain Register #2	U32	RW		CS5523 physical channel AIN2
	14	Offset Register #3	U32	RW		CS5523 physical channel AIN3
	15	Gain Register #3	U32	RW		CS5523 physical channel AIN3
	16	Offset Register #4	U32	RW		CS5523 physical channel AIN4
	17	Gain Register #4	U32	RW		CS5523 physical channel AIN4
	18	Channel-Setup Register #1	U32	RW		LC 1 (12-bits) in lower 2 bytes, LC 2 (12-bits) in upper 2 bytes
	19	Channel-Setup Register #2	U32	RW		LC 3 (12-bits) in lower 2 bytes, LC 4 (12-bits) in upper 2 bytes
	20	Channel-Setup Register #3	U32	RW		LC 5 (12-bits) in lower 2 bytes, LC 6 (12-bits) in upper 2 bytes
	21	Channel-Setup Register #4	U32	RW		LC 7 (12-bits) in lower 2 bytes, LC 8 (12-bits) in upper 2 bytes
*	22	SPI SCLK signal high period (opto-coupler delay)	U8	RW	10	in µs, 10 <= value <= 255
2501		B-sensor #1 ADC-config	Record			CRYSTAL CS5524 24-bit ADC
	0	Number of entries	U8	RO	22	
	1	Number of input channels	U8	RO	7	
*	2	Conversion Word Rate Hall	U8	RW	0	3-bit code [2]
	etc.	*…as above…*	…	…	…	
2502		B-sensor #2 ADC-config	Record			CRYSTAL CS5524 24-bit ADC
2503		B-sensor #3 ADC-config	Record			CRYSTAL CS5524 24-bit ADC

[1] Subindex 2-7 and 22 are common to all B-sensor modules ! (If you change them for one, you change them for all). Writing to subindex 8 and 9 applies to all B-sensor modules.

[2] **000**: 15.0 Hz, **001**: 30.0 Hz, **010**: 61.6 Hz, **011**: 84.5 Hz,
100: 101.1 Hz, **101**: 1.88Hz, **110**: 3.76 Hz, **111**: 7.51 Hz

[3] **000**: 100 mV, **001**: 55 mV, **010**: 25 mV, **011**: 1 V, **100**: 5 V, **101**: 2.5 V

Manufacturer-specific Profile Area (*MDT-DCS*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
2600	-	ADC-reset-and-calibrate B-sensor #0	U8	WO		Writing any value triggers a reset and calibration sequence on B-sensor #0 with its current ADC settings
2601	-	ADC-reset-and-calibrate B-sensor #1	U8	WO		
2602	-	ADC-reset-and-calibrate B-sensor #2	U8	WO		
2603	-	ADC-reset-and-calibrate B-sensor #3	U8	WO		
2700 *	-	ADC-reset-and-calibrate before each channel scan all B-sensors	Bool	RW	0	If =1 a reset/calibration sequence is performed before every B-sensor ADC input channel scan
2800 *	-	B-sensor presence mask	U8	RW	0	Must be <= 15; if a bit=1 the corresponding B-sensor module must be installed
2900		B-sensor #0 identification	Record			DS2401 Identification chip: unique 8-byte serial number
	0	Number of entries	U8	RO	2	
	1	First 4 bytes	U32	RO		
	2	Second 4 bytes	U32	RO		
2901		B-sensor #1 identification	Record			DS2401 Identification chip: unique 8-byte serial number
2902		B-sensor #2 identification	Record			DS2401 Identification chip: unique 8-byte serial number
2903		B-sensor #3 identification	Record			DS2401 Identification chip: unique 8-byte serial number

Manufacturer-Specific Profile Area *(continued…)*						
Index (hex)	**Sub Index**	**Name**	**Data/ Object**	**Attr**	**Default**	**Comment**
2A00		ADC range calibration	Array		**EXPERT ONLY**	For now triggers a 'pure' self-calibration procedure only [1]
	0	Number of entries	U8	RO	6	
	1	Calibrate 25 mV	U32	WO		Write any value…
	2	Calibrate 55 mV	U32	WO		Write any value…
	3	Calibrate 100 mV	U32	WO		Write any value…
	4	Calibrate 1 V	U32	WO		Write any value…
	5	Calibrate 2.5 V	U32	WO		Write any value…
	6	Calibrate 5 V	U32	WO		Write any value…
2B00		ADC calibration parameters 25 mV	Array			Calibration constants (always stored in EEPROM); enable by first writing to 2D00
	0	Number of entries	U8	RO	4	
	1	Gain Factor phys. chan. 1	U32	RW		actual gain factor * 1000000
	2	Gain Factor phys. chan. 2	U32	RW		actual gain factor * 1000000
	3	Gain Factor phys. chan. 3	U32	RW		actual gain factor * 1000000
	4	Gain Factor phys. chan. 4	U32	RW		actual gain factor * 1000000
2B01		ADC calibration parameters 55 mV	Array			Calibration constants (as above)
2B02		ADC calibration parameters 100 mV	Array			"
2B03		ADC calibration parameters 1 V	Array			"
2B04		ADC calibration parameters 2.5 V	Array			"
2B05		ADC calibration parameters 5 V	Array			"
2C00	-	Erase ADC calibration parameters 25 mV	U8	WO	**EXPERT ONLY**	Write EEh to erase; enable by first writing to 2D00
2C01	-	Erase ADC calibration parameters 55 mV	U8	WO	**EXPERT ONLY**	"
2C02	-	Erase ADC calibration parameters 100 mV	U8	WO	**EXPERT ONLY**	"
2C03	-	Erase ADC calibration parameters 1 V	U8	WO	**EXPERT ONLY**	"
2C04	-	Erase ADC calibration parameters 2.5 V	U8	WO	**EXPERT ONLY**	"
2C05	-	Erase ADC calibration parameters 5 V	U8	WO	**EXPERT ONLY**	"
2D00	-	Enable calibration parameter write/erase operation	U8	WO	**EXPERT ONLY**	Writing 0xA5 enables one write or erase operation to any of the Objects 2B00 to 2B05 or 2C00 to 2C05.

[1] In other words: resets the ADC and does a 'self-calibration', i.e. does **NOT** apply the gain factors ('calibration constants'), which already may have been stored in EEPROM earlier. This type of ADC initialisation is essential if the voltage range in question ever needs to be *re*calibrated. Note for MDT-DCS: ADC calibration is not essential, since the T-sensor measurements are ratio measurements, using precision resistors.

Manufacturer-specific Profile Area (*MDT-DCS*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
2E00 *	-	Digital Input debounce timer	U8	RW	10	In units of ca. 400 µs (set to 0 there is ca. 400 µs between consecutive input polls).
2F00 *	-	Digital Output Init	U8	RW	01h	After a hard reset: bits defined as Digital Output will be initialised to the setting corresponding to the bit in this byte (1=high, 0=low)
3000		Program Code CRC	Record			
	0	Number of entries	U8	RO	3	
	1	Check 16-bit CRC of program code in FLASH memory	U16	RO	0	SDO reply unequal to zero means there is a checksum error; absence of CRC results in SDO *Abort* with *Error Code* 1; error while accessing FLASH results in SDO *Abort* with *Error Code* 6.
	2		U16	RO	0	*not used*
	3	Get CRC	U16	RO		Return CRC from flash
3100	-	ELMB Serial Number	U32	RW		Number or 4-byte string uniquely identifying an ELMB, given during production.
3101	-	Enable ELMB Serial Number write operation	U8	WO	**EXPERT ONLY**	Writing 5Ah enables one write operation on the Serial Number (Object 3100).
3200		CAN-controller settings	Record			
	0	Number of entries	U8	RO	3	
*	1	Disable Remote Frames	Bool	RW	0	[1]
*	2	Enable auto-start	U8	RW	0	If =1 go to *Operational* at startup
*	3	Bus-off max retry counter	U8	RW	5	Counter is decremented every 1s, but if the node reaches this maximum value it abandons regaining CAN-bus access
3300	-	CAN Node Identifier	U8	WO		The new CAN Node Identifier is used after the next reset. (ELMB *Bootloader* firmware version 1.3 and later supports this feature, otherwise don't use it !)
3301	-	Enable CAN Node Identifier write operation	U32	WO	**EXPERT ONLY**	Writing a number that matches the ELMB Serial Number (Object 3100) enables one write operation on the CAN Node Identifier (Object 3300).

[1] Due to the way the ELMB's CAN-controller handles Remote Frames, it is recommended to disable Remote Frames permanently if not needed (for PDO read-out). A special provision in the software has been made to ensure that the CANopen Node Guard Remote Frame is still handled properly.

Manufacturer-specific Profile Area (*MDT-DCS*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
4000		Read NTC resistor value	Array			in Ohm (division of 2 consecutive analogue inputs)
	0	Number of entries	U8	RO	32	This value fixed, but actual hardware configuration may vary (depends on OD-index 2100, sub 1)
	1	NTC 0	U16	RO		
	2	NTC 1	U16	RO		
	…	…	…	…		
	…	…	…	…		
	30	NTC 29	U16	RO		
	31	reference resistor (*NTC* 30)	U16	RO		R=16369 Ohm (± 1%)
	32	reference resistor (*NTC* 31)	U16	RO		R=341.6 Ohm (± 1%)
4010		Read NTC temperature	Array			in millidegrees centigrade
	0	Number of entries	U8	RO	32	This value fixed, but actual hardware configuration may vary (depends on OD-index 2100, sub 1)
	1	NTC 0 temperature	I32	RO		
	2	NTC 1 temperature	I32	RO		
	…	…	…	…		
	…	…	…	…		
	30	NTC 29 temperature	I32	RO		
	31	ref temperature (*NTC* 30)	I32	RO		T = 0 m°C
	32	ref temperature (*NTC* 31)	I32	RO		T = 100000 m°C
4100		Read analogue input CSM-ADC	Record			8 bits flags [1], 16 bits analogue value (CSM)
	0	Number of entries	U8	RO	64	This value fixed, but actual hardware configuration may vary (see OD-index 2101, sub 1)
	1	Input 1 (CSM-ADC)	I24	RO		1st analog input:16-bit+8-bit flgs[1]
	2	Input 2 (CSM-ADC)	I24	RO		2nd " " " "
	…	…	…	…		…
	…	…	…	…		…
	64	Input 64 (CSM-ADC)	I24	RO		64th " " " "

[1] See section 5.2.1 for a description of the ADC 'flags' byte.

Manufacturer-specific Profile Area (*MDT-DCS*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
4200		Read analogue input B-sensor #0	Record			24 bits analogue value (B-sensor #0)
	0	Number of entries	U8	RO	7	Fixed value (see OD-index 2500, subindex 1)
	1	Input 1 (B-sensor ADC #0)	I24	RO		1st analog input:24-bit (Hall H1)
	2	Input 2 (B-sensor ADC #0)	I24	RO		2nd " " " (Hall H2)
	3	Input 3 (B-sensor ADC #0)	I24	RO		3rd " " " (Hall H3)
	4	Input 4 (B-sensor ADC #0)	I24	RO		4th " " (fullscale Hall)
	5	Input 5 (B-sensor ADC #0)	I24	RO		5th " " " (NTC)
	6	Input 6 (B-sensor ADC #0)	I24	RO		6th " " " (0ºC ref)
	7	Input 7 (B-sensor ADC #0)	I24	RO		7th " " " (100ºC ref)
4201		Read analogue input B-sensor #1	Record			24 bits ADC count (B-sensor #1)
4202		Read analogue input B-sensor #2	Record			24 bits ADC count (B-sensor #2)
4203		Read analogue input B-sensor #3	Record			24 bits ADC count (B-sensor #3)
4300		Read Analogue Input of NTC, calibrated	Record			8 bits flags [1], 24 bits analogue value, in µV; odd ch: NTC voltage, even ch: NTC current (I=V/10kΩ) NB: read-out is denied if there are no valid calibration constants for the current ADC settings
	0	Number of analog inputs	U8	RO	64	Fixed, but actual hardware configuration may vary (set in Object 2100, sub 1)
	1	Input 1	U32	RO		1st analog input: 8-bit flags + 24-bit (signed) data
	2	Input 2	U32	RO		2nd " " " "
	…	…	…	…		…
	…	…	…	…		…
	64	Input 64	U32	RO		64th " " " "
4400 *	-	T- and B-sensor NTC readings in PDO messages in degrees centigrade	Bool	RW	1	If =1 NTC readings in PDO messages are converted to millidegrees centigrade instead of Ohms (using hardcoded conversion formulas; see text)
4401 *	-	Raw T-sensor data readout in PDO	Bool	RW	0	If =1 TPDO2 messages contain individual analog input channel data

[1] See section 5.2.1 for a description of the ADC 'flags' byte.

Manufacturer-specific Profile Area (*MDT-DCS*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
4800	-	Shift IR (Instruction Register)	U32	RW		- Go to state *Shift-IR*. - **W**: shift in 32 bits. - **R**: read the 32 bits (or less) that were shifted out in the previous write (**W**) operation. - Remain in state *Shift-IR*.
4801		Final IR shift	Record			
	0	Number of entries	U8	RO	32	
	1	Final IR bit shift (1 bit)	U32	RW		- Go to state *Shift-IR*. - **W**: shift in 1 bit. - **R**: read 32 bits (or less) that were shifted out in the previous write operation. (identical to Object 4800) - Go to state *Run-Test/Idle*.
	2	Final IR bit shift (2 bits)	U32	RW		*idem*, but shift in 2 bits
	…	…	…	…		…etc
	…	…	…	…		…etc
	32	Final IR bit shift (32 bits)	U32	RW		*idem*, but shift in 32 bits
4802	-	Current total number of IR bits shifted	U32	RO	0	Can be used by host system to check if all uploaded bits up to now have been received by MDT-DCS
4803		Shift DR (Data Register)	U32	RW		- Go to state *Shift-DR*. - **W**: shift in 32 bits. - **R**: read the 32 bits (or less) that were shifted out in the previous write (**W**) operation. - Remain in state *Shift-DR*.
4804		Final DR shift	Record			
	0	Number of entries	U8	RO	32	
	1	Final DR bit shift (1 bit)	U32	RW		- Go to state *Shift-DR*. - **W**: shift in 1 bit.. - **R**: read 32 bits (or less) that were shifted out in the previous write operation (same as Object 4803) - Go to state *Run-Test/Idle*..
	2	Final DR bit shift (2 bits)	U32	RW		*idem*, but shift in 2 bits
	…	…	…	…		…etc
	…	…	…	…		…etc
	32	Final DR bit shift (32 bits)	U32	RW		*idem*, but shift in 32 bits
4805	-	Current total number of DR bits shifted	U32	RO	0	Can be used by host system to check if all uploaded bits up to now have been received by MDT-DCS

Manufacturer-specific Profile Area (*MDT-DCS*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
480A	-	Shift IR (Instruction Register)	Domain	RW		**Segmented SDO only** - First 2 data bytes must contain number of bits in JTAG string to follow, max 8192 bits - **W**: shift in N bits. - **R**: read the N bits that were shifted out in the previous write (**W**) operation.
480B	-	Shift DR (Data Register)	Domain	RW		**Segmented SDO only** - First 2 data bytes must contain number of bits in JTAG string to follow, max 8192 bits - **W**: shift in N bits. - **R**: read the N bits that were shifted out in the previous write (**W**) operation.
480C	-	Shift IR (Instruction Register)	Domain	RO		**Segmented SDO only** - Read-only copy of Obj 480Ah
480D	-	Shift DR (Data Register)	Domain	RO		**Segmented SDO only** - Read-only copy of Obj 480Bh
4830	-	JTAG TAP state	U8	RW	8	read or set JTAG TAP state; see text for definitions of states
4831	-	JTAG TAP state after a Shift IR operation	U8	RW	8	see text for definitions of states
4832	-	JTAG TAP state after a Shift DR operation	U8	RW	8	see text for definitions of states
4840	-	JTAG TAP reset	U8	WO		Trigger JTAG TAP reset sequence, then go to TAP state *Run-Test/Idle*
4850	-	Generate JTAG TCK cycles	U32	RW		Write value *n*: *n* TCK cycles are generated without changing state: only possible while in certain JTAG TAP states (see text). Read: returns the remaining number of clock ticks to generate
4860 *	-	JTAG TCK signal high period	U8	RW	0	0 <= value <= 3 width = (1.5 + value*0.5) µs
4870	-	TAP count	U8	RO		*For test purposes*: triggers a procedure to count the number of TAPs (BYPASS instruction is shifted into each TAP) and returns the number found; maximum number of TAPs: 31

Manufacturer-specific Profile Area (*MDT-DCS*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
4910	-	JTAG-action #1 JTAG Instruction String storage (<=128 bits total)	U32	RW		**W**: write 32 bits to storage, increment string index by 32. **R**: read <=32 bits from storage.
4911		Storage #1 completion JTAG Instruction String	Record			<=32 IR bits
	0	Number of entries	U8	RO	32	
	1	Final IR bits (1 bit)	U32	RW		**R**: reset string index for reading. **W**: write 1 bit, store string length and CRC, reset string index for writing.
	2	Final IR bits (2 bits)	U32	RW		*idem*, but write 2 bits
	…	…	…	…		…etc
	32	Final IR bits (32 bits)	U32	RW		*idem*, but write 32 bits
4912	-	String length (number of bits)	U32	RO	0	Length of stored string
4913	-	JTAG-action #1 JTAG Data String storage (<= 6272 bits total)	U32	RW		**W**: write 32 bits to storage, increment string index by 32. **R**: read <=32 bits from storage.
4914		Storage #1 completion JTAG Data String	Record			<=32 DR bits
	0	Number of entries	U8	RO	32	
	1	Final DR bits (1 bit)	U32	RW		**R**: reset string index for reading. **W**: write up to 32 bits (the number of bits to write is in sub-index 0), store string length and CRC, reset string index for writing.
	2	Final DR bits (2 bits)	U32	RW		*idem*, but write 2 bits
	…	…	…	…		…etc
	32	Final DR bits (32 bits)	U32	RW		*idem*, but write 32 bits
4915	-	String length (number of bits)	U32	RO	0	Length of stored string
4916	-	Reset string indices	U8	WO		To restart a read or write string operation: resets string indices for reading and writing
4917	-	Execute JTAG-action #1: upload Instruction/Data Strings	U8	WO		Write 55h to trigger upload of the instruction string followed by the data string
4918		JTAG-action #1 DR Status Return Bits Configuration	Record			'RW' pars are stored in EEPROM
	0	Number of entries	U8	RO	3	
	1	Start bit in DR out string	U32	RW	0	
	2	Status mask	U32	RW	00000000h	Bits != 0 are checked
	3	Status expected	U32	RW	00000000h	Expected status bits

Manufacturer-specific Profile Area (*MDT-DCS*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
491A	-	JTAG-action #1 JTAG Instruction String storage (<=128 bits total)	Domain	RW		**Segmented SDO only** - First 2 data bytes must contain number of bits in JTAG string to follow - **W**: write N bits to storage. - **R**: read the N bits from storage.
491B	-	JTAG-action #1 JTAG Data String storage (<= 6272 bits total)	Domain	RW		**Segmented SDO only** - First 2 data bytes must contain number of bits in JTAG string to follow - **W**: write N bits to storage. - **R**: read the N bits from storage.
491C	-	JTAG-action #1 JTAG Instruction String storage (<=128 bits total)	Domain	RO		**Segmented SDO only** - Read-only copy of Obj 491Ah
491D	-	JTAG-action #1 JTAG Data String storage (<= 6272 bits total)	Domain	RO		**Segmented SDO only** - Read-only copy of Obj 491Bh
491E	-	JTAG-action #1 IR and DR String CRC	U32	RO		IR CRC in byte 0 (LSB) and 1 DR CRC in byte 2 (LSB) and 3
491F	-	Erase JTAG-action #1 Instruction/Data Strings and/or Status Return Bits Configuration	U8	WO		Write to erase the stored strings and/or status return config; resets string indices for reading and writing. AAh: erase all; ABh: erase strings; ACh: erase status

The list of objects in 4910h to 491Fh is repeated for every JTAG-action in storage:

Manufacturer-specific Profile Area (*MDT-DCS*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
492x	-	JTAG-action #2				<=128 bits IR, <=6272 bits DR
493x	-	JTAG-action #3				<=128 bits IR, <=6272 bits DR
494x	-	JTAG-action #4				<=128 bits IR, <=512 bits DR
495x	-	JTAG-action #5				<=128 bits IR, <=512 bits DR
496x	-	JTAG-action #6				<=128 bits IR, <=512 bits DR
497x	-	JTAG-action #7				<=128 bits IR, <=512 bits DR
498x	-	JTAG-action #8				<=128 bits IR, <=512 bits DR
499x	-	JTAG-action #9				<=128 bits IR, <=512 bits DR
49Ax	-	JTAG-action #10				<=128 bits IR, <=512 bits DR
49Bx	-	JTAG-action #11				<=128 bits IR, <=512 bits DR
49Cx	-	JTAG-action #12				<=128 bits IR, <=512 bits DR
49Dx	-	JTAG-action #13				<=128 bits IR, <=512 bits DR
49F0		JTAG-action Data Register status return	Record			
	0	Number of entries	U8	RO	3	
	1	Last JTAG-action completed	U8	RO	0	1 <= action <= 13 = 0: no actions completed since last reset
	2	Status return bits	U32	RO	00000000h	Status bits of last JTAG-action
	3	Status return bits error	U32	RO	00000000h	Status error of last JTAG-action: bits != 0 are in error; result of: (stat & mask) ^ (expect & mask)
49F1 *	-	Report JTAG-action DR status error	Bool	RW	1	=1: PDO message is sent on every JTAG-action completion

Manufacturer-specific Profile Area (*MDT-DCS*) *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
5C00	-	Compile-time Options	U32	RO		Bitmask denoting which compile options were used when the application code was generated (see Table 14 below for details)
5DFF		ELMB/MDT-DCS board tests	Array		**EXPERT ONLY**	
	0	Number of test objects	U8	RO	6	
	1	Test of I/O-pins	U32	RO		*For use in ATLAS DCS ELMB production and test stand only; described elsewhere*
	2	Generate Watchdog Timer reset	U32	RO	-	firmware goes into an endless loop
	3	Test of MDT I/O-pins	U32	RO		*For use in ATLAS MDT-DCS module production and test stand only described in section 11*
	4,5,6	Additional tests of MDT I/O-pins	U32	RO	0	*For use in ATLAS MDT-DCS module production and test stand only described in section 11*
5E00	-	Transfer control to Bootloader	U8	WO		

Object 5C00: Compile Options		
Bit	***Option***	***Comment***
0	–	–
1	–	–
2	–	–
3	–	–
4	–	–
5	–	–
6	–	–
7	ELMB103	the ELMB is an ELMB103 type (with ATmega103 microcontroller); by default an ELMB128 (with ATmega128 microcontroller) is assumed
8	VARS_IN_EEPROM	Store/retrieve working copies of configuration parameters in/from EEPROM
9	–	–
10	INCLUDE_TESTS	Include an OD object through which (board) tests can be executed
11	–	–
12	CAN_REFRESH	Refresh CAN-controller descriptor register (at each buffer write/read)
13	–	–

Table 14. Optional compile-time macro defines, which can be read from Object 5C00h. (in the source code individual options are surrounded by a double underscore '__').

Standardised Device Profile Area (*MDT-DCS*)						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
6000		Read state 8 input lines	Array			
	0	Number of 8-bit inputs	U8	RO	1	
	1	Read inputs 1-8	U8	RO		ELMB ATmega128 PORTA/F (Port shared with Object 6200,1)
6005 *	-	Global Digital Input Interrupt Enable	Bool	RW	0	Enables/disables *change-of-state* TPDO1 transmissions
6006		Interrupt Mask Any Change 8 input lines	Array			Enables/disables on a per-input-bit basis *change-of-state* TPDO1 transmissions
	0	Number of 8-bit inputs	U8	RO	1	
	1	Interrupt Mask Inputs 1-8	U8	RW	FFh	
6200		Write state 8 output lines	Array			
	0	Number of 8-bit outputs	U8	RO	1	
	1	Write outputs 1-8	U8	RW		ELMB ATmega128 PORTA/F (Port shared with Object 6000,1)
6208		Filter mask 8 output lines	Array			
	0	Number of 8-bit masks	U8	RO	1	
*	1	Filter mask outputs 1-8	U8	RW	0Fh	maskbit=1: I/O is an output; pins not defined as outputs are inputs, to be accessed thru Object 6000, 1
6220		Write output bit	Array			Only bits defined as output (Object 6208, sub 1) can be written
	0	Number of 1-bit outputs	U8	RO	7	
	1	Write output 1	Bool	RW		DIGIO1 (PORTA pin 4)
	2	Write output 2	Bool	RW		DIGIO2 (PORTA pin 5)
	3	Write output 3	Bool	RW		DIGIO3 (PORTA pin 6)
	4	Write output 4	Bool	RW		DIGIO4 (PORTA pin 7)
	5	Write output 5	Bool	RW		DIGIO5 (PORTF pin 2)
	6	Write output 6	Bool	RW		DIGIO6 (PORTF pin 3)
	7	Write output 7	Bool	RW		DIGIO7 (PORTF pin 4)
6404		Read analogue input manufacturer-specific (NTC-ADC)	Record			8 bits flags [1], 16 bits analogue value (NTC); odd ch: NTC voltage, even ch: NTC current (I=V/10kΩ)
	0	Number of entries	U8	RO	64	This value fixed, but actual hardware configuration may vary (see OD-index 2100, sub 1)
	1	Input 1 (NTC-ADC)	I24	RO		1st analog input:16-bit+8-bit flgs[1]
	2	Input 2 (NTC-ADC)	I24	RO		2nd " " " "
	…	…	…	…		…
	…	…	…	…		…
	64	Input 64 (NTC-ADC)	I24	RO		64th " " " "

[1] See section 5.2.1 for a description of the ADC 'flags' byte.

10 Emergency Objects

CANopen *Emergency* messages are triggered by the occurrence of an internal (fatal) error situation. An *Emergency* CAN-message has the following general syntax:

MDT-DCS → Host

COB-ID	Byte 0-1	Byte 2	Byte 3-7
080h + *NodeID*	Emergency Error Code	Error Register (Object 1001h)	Manufacturer specific error field

Starting from MDT-DCS firmware version 2.3 a toggle bit was added to byte 7 of the Emergency message. Byte 7 alternates between the values 00h and 80h from one Emergency message to the next.

The following Emergency messages can be generated by the **MDT-DCS** application:

Error Description	Emergency Error Code (byte 1-0; hex)	Manufacturer-specific Error Field (byte 3-7)
CAN communication	8100	Byte 3: 81C91 Interrupt Register content [1] Byte 4: 81C91 Mode/Status Register content [2] Byte 5: error counter Byte 6: bus-off counter (see OD index 3200, sub 3)
CAN buffer overrun	8110h	CAN message buffer in RAM full: at least 1 message was lost
Life Guarding time-out	8130	CAN-controller has been reinitialized
RPDO: too few bytes	8210	Byte 3: minimum DLC (Data Length Code)
NTC-ADC / CSM-ADC: conversion timeout	5000	Byte 3: 01h (NTC) / 61h (CSM) Byte 4: ADC channel number (0..63) Byte 5: 00h
NTC-ADC / CSM-ADC: reset failed	5000	Byte 3: 02h (NTC) / 62h (CSM) Byte 4: 00h Byte 5: Error id [3]
NTC-ADC / CSM-ADC: offset calibration failed	5000	Byte 3: 03h (NTC) / 63h (CSM) Byte 4: 00h
NTC-ADC / CSM-ADC: gain calibration failed	5000	Byte 3: 04h (NTC) / 64h (CSM) Byte 4: 00h
NTC-ADC / CSM-ADC: problem(s) during initialisation	5000	Byte 3: 05h (NTC) / 65h (CSM) Byte 4: ADC status (see OD index 1002h)
Slave processor not responding (only on ELMB103)	5000	Byte 3: 20h

…table continues on the next page…

[1] 81C91 *INT* register bits: **04h**: Warning Level, **20h**: Bus Off, **40h**: Error Passive, **80h**: Transmit Check

[2] 81C91 *MODE/STATUS* register bits: **01h**: Init Mode, **02h**: Reset State, **04h**: Bus Off, **08h**: Receive Error Counter >= 96; **10h**: Transmit Error Counter >= 96, **20h**: last Transmission Complete, **40h**: Receive Mode, **80h**: Auto Decrement Address

[3] **01**: Reset-Valid bit not set, **02**: Reset-Valid bit not reset, **04**: error in Offset Register value, **08**: error in Gain Register value

Error Description	Emergency Error Code (byte 1-0; hex)	Manufacturer-specific Error Field (byte 3-7)
CRC error	5000	Byte 3: 30h Byte 4: 1 (program FLASH), 2 (Slave FLASH; ELMB103 only)
EEPROM: write error	5000	Byte 3: 41h Byte 4: Parameter block index [1] Byte 5: 0 : writing block info > 0: size of parameter block to write
EEPROM: read error	5000	Byte 3: 42h Byte 4: Parameter block index [1] Byte 5: Error id (1=CRC, 2=length, 4=infoblock)
B-sensor ADC: conversion timeout	5000	Byte 3: 51h Byte 4: B-sensor number (0..3) Byte 5: ADC channel number (0..7)
B-sensor ADC: reset failed	5000	Byte 3: 52h Byte 4: B-sensor number (0..3) Byte 5: Error id [2]
B-sensor ADC: Hall-sensor calibration failed	5000	Byte 3: 53h Byte 4: B-sensor number (0..3)
B-sensor ADC: T-sensor calibration failed	5000	Byte 3: 54h Byte 4: B-sensor number (0..3)
B-sensor ADC problem(s) during initialisation (check OD 1002)	5000	Byte 3: 55h Byte 4: ADC 0+1 status (see OD index 1002) Byte 5: ADC 2+3 status (see OD index 1002)
JTAG: bit string shift protocol error	8200	Byte 3: 71h Byte 4: TAP state (0Bh = Shift-IR, 02h = Shift-DR) Byte 5: number of bits to shift Byte 6: 1 = final shift, 0 = not final shift
JTAG: sequence protocol error	8200	Byte 3: 72h
JTAG: segmented protocol error	8200	Byte 3: 73h Byte 4: 1 = segment too short, 2 = number of bits too large, 3 = received more bits than expected
JTAG: JTAG-action not available	5000	Byte 3: 81h Byte 4: JTAG-action number (1..13)
JTAG: JTAG-action CRC error	5000	Byte 3: 82h Byte 4: JTAG-action number (1..13) Byte 5: 5Ah = instruction string, A5h = data string
JTAG: JTAG-action status return CRC error	5000	Byte 3: 83h Byte 4: JTAG-action number (1..13)

…table continues on the next page…

[1] **0**: PDO communication parameters, **1**: Guarding parameters, **2**: Digital I/O configuration, **3**: NTC ADC configuration, **4**: B-sensor ADC configuration, **5**: CSM ADC configuration, **6**: CAN configuration parameters, **7**: JTAG parameters, **FEh**: Calibration constant(s), **FFh**: ELMB Serial Number.

[2] **01h**: Reset-Valid bit not set, **02h**: Reset-Valid bit not reset, **04h**: error in initial Offset Register value, **08h**: error in initial Gain Register value.

Error Description	Emergency Error Code (byte 1-0; hex)	Manufacturer-specific Error Field (byte 3-7)
Irregular reset (Watchdog, Brown-out or JTAG)	5000	Byte 3: F0h Byte 4: microcontroller MCUCSR register contents [1]
Bootloader: not present	5000	Byte 3: F1h
Bootloader is now in control [2]	5000	Byte 3: FEh Byte 4: 01h Byte 5: 28h Byte 6: microcontroller MCUCSR register contents [1] Byte 7: 00h
Bootloader cannot jump to application: invalid [2]	6000	Byte 3: FEh Byte 4: AAh Byte 5: AAh Byte 6: 00h Byte 7: 00h

Byte 2 of the *Emergency* message contains the value of the socalled *Error Register* (Object Dictionary index 1001h, a mandatory CANopen object). One or more bits of the 8-bit Error Register can be set to 1, depending on the node's history of errors since the last reset. The table below gives a description of the different bits.

Error Register (Object 1001h) bits	
Bit	**Error type**
0	generic
1	current
2	voltage
3	temperature
4	communication
5	device profile specific
6	*reserved (=0)*
7	manufacturer specific

11 Built-In Board Test

A connectivity test function for the I/O-lines has been implemented in the MDT-DCS application firmware, specifically for (*offline*) board test and acceptance test (after production) purposes, so that a full pin connection test can be done, in combination with some custom external hardware, i.e. an array of resistors for the NTC connections, some cables plus an interconnection board for all the other connectors; in addition an auxiliary (modified) MDT-DCS module is used for measuring voltages and currents of the module under test. See the pictures in Figure 7 below.

[1] ATmega128 *MCUCSR* register bits: **01h**: Power-On Reset, **02h**: External Reset, **04h**: Brown-Out Reset, **08h**: Watchdog Reset, **10h**: JTAG Reset, **80h**: JTAG Interface Disable.

[2] This Emergency message is generated by the Bootloader program !

The I/O test is integrated in the *standard* MDT-DCS application firmware, making it possible to do board (acceptance) testing without having to upload special software, i.e. the MDT-DCS application firmware provides objects to trigger the tests; test commands and test results are sent in messages via the CAN-bus.

The I/O test requires that the I/O-lines are interconnected in a predefined fashion, shown below; tested are PORTA and (parts of) PORTC, PORTD, PORTE and PORTF; in brackets the initial data-direction setting.

- PORTC3-7 connected to PORTE3-7:
  - CSM_CS (out) ↔ B_SCLK_0 (out)
  - AUX_IO4 (in) ↔ CSM_SCLK (out)
  - AUX_IO5 (in) ↔ CSM_SDI (out)
  - AUX_IO2 (in) ↔ CSM_SDO (in)
  - AUX_IO1 (in) ↔ CSM_MUX (out)

- PORTA0-7 connected to PORTF0-7:
  - TDO (out) ↔ B_ID0 (in)
  - TCK (out) ↔ B_ID1 (in)
  - TMS (out) ↔ DIGIO5 (in)
  - TDI (in) ↔ DIGIO6 (in)
  - DIGIO1 (out) ↔ DIGIO7 (in)
  - (for DIGIO2 see below)
  - DIGIO3 (out) ↔ AUX_IO3 (in)
  - DIGIO4 (out) ↔ B_SDO_0 (in)

- Remaining I/O-lines: B_CS0 (PC0), B_CS1 (PC1), B_SDI (PD3) en DIGIO2 (PA5):
  - B_CS0 (out) ↔ DIGIO2 (out)
  - B_CS1 (out) ↔ B_SDI_0 (out)

The test procedure comprises:

- all possible output values of PORTA(0-7) and PORTC(3-7), per port, using PORTF (0-7) and PORTE (3-7) as inputs, resp.
- all possible output values of PORTA(5) and PORTD(3), using PORTC(0-1) as inputs.
- a walking-1 and a walking-0 on the combined ports (the walking-1/0 'runs' from PORTA(0-7) to PORTC(0-7).

The test is triggered by reading CANopen Object 5DFFh sub 3, the returned 32-bit value contains 4 bytes with errorbits per port: Byte 0: PORTA, Byte 1: PORTC, Byte 2: PORTE, Byte 3: PORTF. The returned value is zero if no errors occurred.

A bit that is set means: the corresponding I/O line of the corresponding PORT has at least once been *read* incorrectly during the test sequence described above.

The total test time of this digital I/O test is in the order of 300 ms (using a signal settling time of 1 ms).

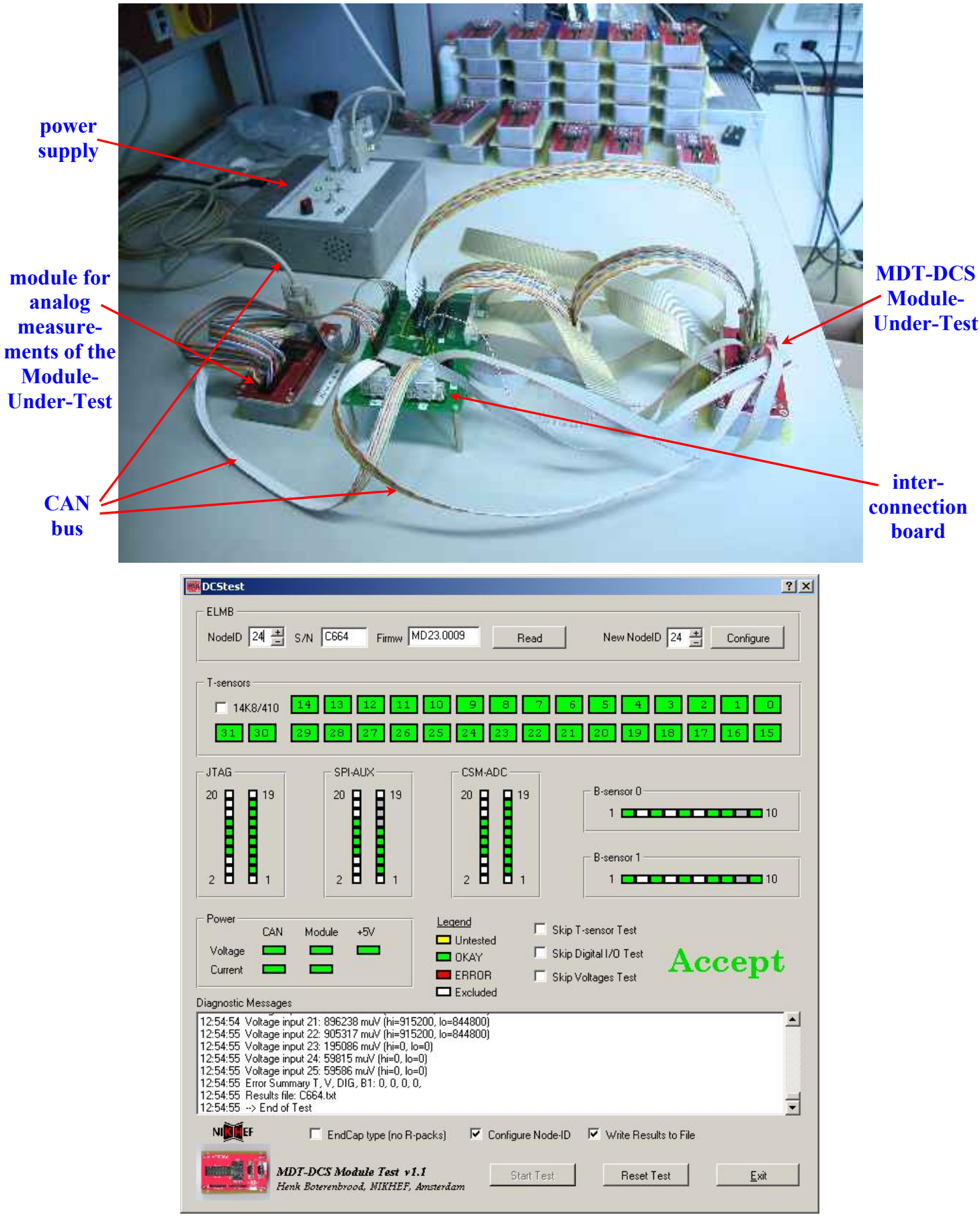


Figure 7. MDT-DCS module connectivity test: set-up (top) and user interface (bottom).

Missing in the digital test described above is a test of proper connectivity of the B_SCLK, B_SDI and B_SDO signals of the "B-sensor 1" connector. In the test set-up these signals have been routed to analog inputs on the auxiliary MDT-DCS module. Test objects 5DFFh, sub 4 and 5 have been added, which set these 3 signals as outputs to 0, 1 and 0 respectively, and to 1, 0 and 1 respectively, so that the signals can then be checked by reading analog inputs from the auxiliary MDT-DCS module. Reading object 5DFFh, sub 6 sets the signals back to their normal setting.

In this way all pins, except ground-pins, of an MDT-DCS module are checked for proper connectivity. A module that passes this test is accepted for use in the ATLAS MDT detector.

References

[1] H.Boterenbrood,
CANopen Application Software for the ELMB128,
Version 2.1, NIKHEF, Amsterdam, 2 March 2004.
http://www.nikhef.nl/pub/departments/ct/po/html/ELMB128/ELMBio.pdf

[2] H.Boterenbrood,
CANopen, high-level protocol for CAN-bus,
Version 3.0, NIKHEF, Amsterdam, 20 March 2000.
http://www.nikhef.nl/pub/departments/ct/po/doc/CANopen30.pdf

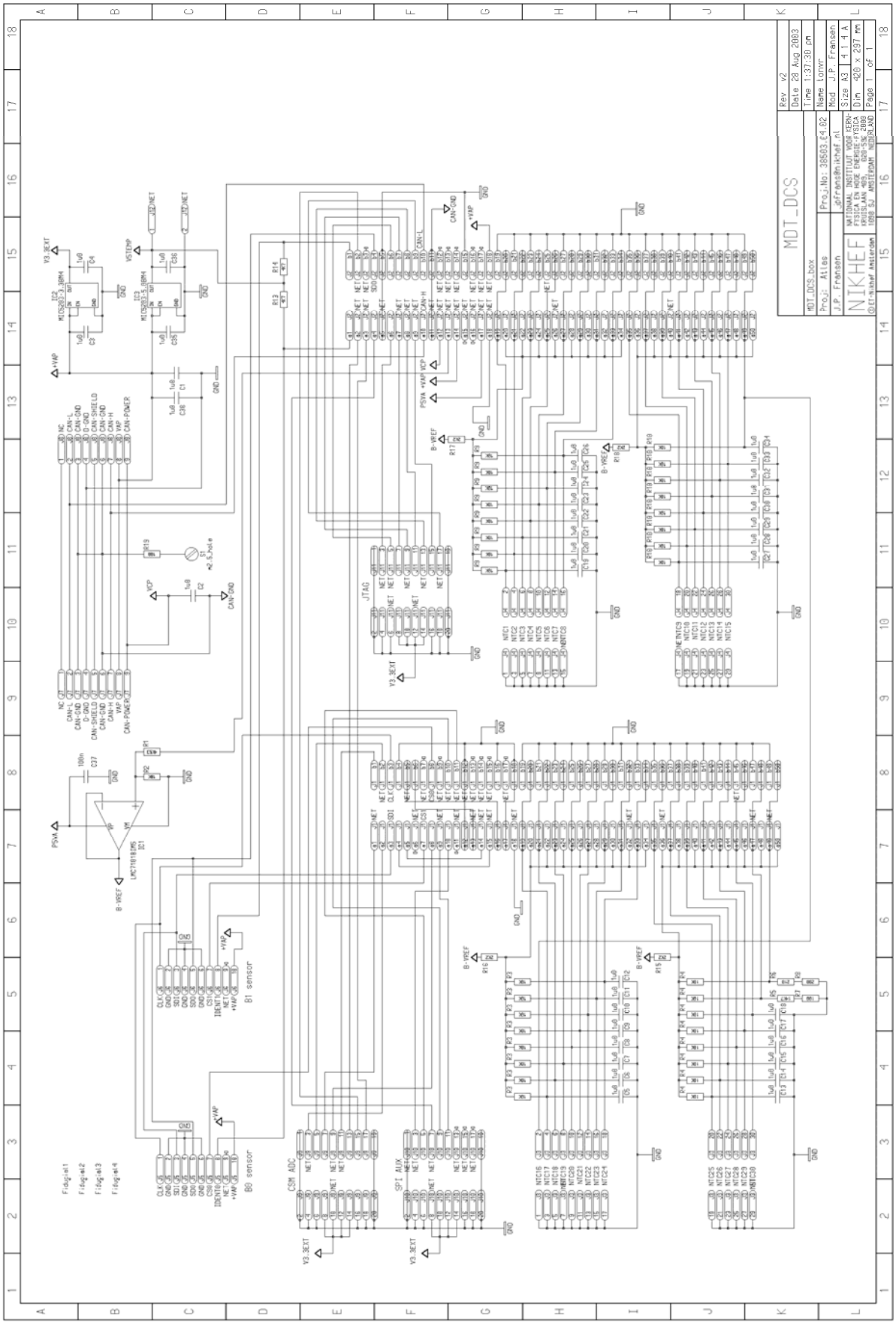
[3] CAN-in-Automation e.V.,
CANopen, Application Layer and Communication Profile,
CiA DS-301, Version 4.0, 16 June 1999.

[4] CAN-in-Automation e.V.,
CANopen Device Profile for Generic I/O Modules,
CiA DS-401, Version 2.0, 20 December 1999.

[5] H.Boterenbrood,
CANopen Bootloader for the ELMB ATmega128 microcontroller,
Version 1.1, NIKHEF, Amsterdam, 10 March 2004.
http://www.nikhef.nl/pub/departments/ct/po/html/ELMB128/ELMBbl-doc.pdf

[6] **ELMB software resources webpage:**
http://www.nikhef.nl/pub/departments/ct/po/html/ELMB/ELMBresources.html

Appendix A. MDT-DCS Motherboard Schematic



Appendix B. NTC Temperature Sensor Data

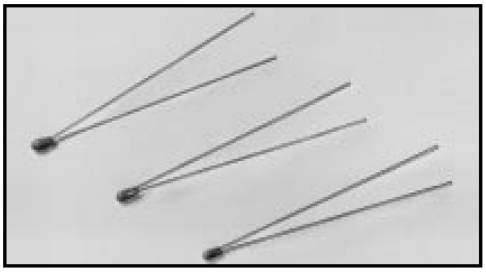
(datasheets taken from manufacturer website: http://www.thermometrics.com/)

NTC THERMISTORS: TYPE DC95

INTERCHANGEABLE CHIP THERMISTOR

DESCRIPTION:

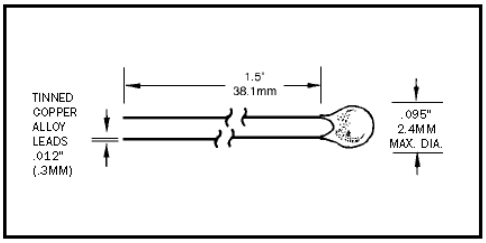
Epoxy coated interchangeable chip thermistors with bare tinned copper lead-wires.



FEATURES:

- Precision, solid state temperature sensor
- Interchangeability down to ±0.1 °C
- Suitable for use over the range of –80°C to 150°C
- High sensitivity greater than –4%/°C at 25°C
- Suitable for temperature measurement, control and compensation
- High reliability and stability over interchangeable range
- Most popular R-vs-T curves are available
- Resin coated for good mechanical strength and resistance to solvents
- .012" (.3 mm) dia. bare tinned copper lead-wires

DIMENSIONS:



Select appropriate part number below for resistance and temperature tolerance desired

$R_{25°C}$	MATERIAL SYSTEM	± .1°C 0°C to 70°C	± .2°C 0°C to 70°C	± .2°C 0°C to 100°C
2000	F	DC95F202V	DC95F202W	DC95F202Z
2252	F	DC95F232V	DC95F232W	DC95F232Z
3000	F	DC95F302V	DC95F302W	DC95F302Z
5000	F	DC95F502V	DC95F502W	DC95F502Z
10000	F	DC95F103V	DC95F103W	DC95F103Z
10000	Y	DC95Y103V	DC95Y103W	DC95Y103Z
30000	H	DC95H303V	DC95H303W	DC95H303Z
50000	G	DC95G503V	DC95G503W	DC95G503Z
100000	Y	DC95Y104V	DC95Y104W	DC95Y104Z
100000	G	DC95G104V	DC95G104W	DC95G104Z

OPTIONS:

Consult factory for availability of options:

- Other resistance values in the range of 100Ω - 100kΩ
- Other tolerances or ranges
- Alternative lead-wires or lengths
- Non standard R-vs-T curves
- Controlled dimensions

DATA:

THERMAL AND ELECTRICAL PROPERTIES:

Dissipation constant:.....(still air) 1 mW/°C
(stirred oil) 8 mW/°C

Thermal time constant:.....(still air) 10 sec.
(stirred oil) 1 sec.

Maximum power at 25°C 75mW
(derated from 100% at 25°C to 0% at 100°C)

BOWTHORPE THERMOMETRICS
Crown Industrial Estate, Priorswood Road
Taunton, Somerset TA2 8QY UK
Tel +44 (0) 1823 335200
Fax +44 (0) 1823 332637

THERMOMETRICS, INC.
808 US Highway 1
Edison, New Jersey 08817-4695 USA
Tel +1 (732) 287 2870
Fax +1 (732) 287 8847

KEYSTONE THERMOMETRICS CORPORATION
967 Windfall Road
St. Marys, Pennsylvania 15857-3397 USA
Tel +1 (814) 834 9140
Fax +1 (814) 781 7969

MATERIAL TYPE: F

AVAILABLE PRODUCTS:

HM, C100, EC95, DC95, MC65, MF65, SC30, SC50

Data for material type : F

Temp Range (°C)	Ratio	Beta
0 to 50	9.08	3895
0 to 70	18.64	3917
25 to 50	2.78	3933
25 to 85	9.30	3969
25 to 100	14.64	3981
25 to 125	29.05	3999
37.8 to 104.4	9.67	4000

To calculate Rt/R25 at temperatures other than those listed in the table, use the following equation:

$Rt/R25 = \exp\{A + B/T + C/T^2 + D/T^3\}$

where T = temperature in K

where K = °C + 273.15

Temp Range (°C)	A	B	C	D
-50 to 0	-1.4122478E+01	4.4136033E+03	-2.9034189E+04	-9.3875035E+06
0 to 50	-1.4141963E+01	4.4307830E+03	-3.4078983E+04	-8.8941929E+06
50 to 100	-1.4202172E+01	4.4975256E+03	-5.8421357E+04	-5.9658796E+06
100 to 150	-1.6154078E+01	6.8483992E+03	-1.0004049E+06	1.1961431E+08

To calculate the actual thermistor temperature as a function of the thermistor resistance, use the following equation:

$1/T = a + b(\text{Ln } Rt/R25) + c(\text{Ln } Rt/R25)^2 + d(\text{Ln } Rt/R25)^3$

Rt/R25 range	a	b	c	d
68.600 to 3.274	3.3538646E-03	2.5654090E-04	1.9243889E-06	1.0969244E-07
3.274 to 0.36036	3.3540154E-03	2.5627725E-04	2.0829210E-06	7.3003206E-08
0.36036 to 0.06831	3.3539264E-03	2.5609446E-04	1.9621987E-06	4.6045930E-08
0.06831 to 0.01872	3.3368620E-03	2.4057263E-04	-2.6687093E-06	-4.0719355E-07

†The deviation resulting from the tolerance on the material constant, Beta. The deviation must be added to the resistance tolerance of the part as specified at 25°C.

Temperature (°C)	Rt/R25 nominal	Temp Coef ∝ (%/°C)	β Deviation† (±%)
-50	68.60	7.21%	2.30%
-45	48.16	6.96%	2.68%
-40	34.23	6.71%	2.87%
-35	24.62	6.48%	2.92%
-30	17.91	6.26%	2.86%
-25	13.17	6.05%	2.71%
-20	9.782	5.85%	2.50%
-15	7.339	5.66%	2.25%
-10	5.558	5.47%	1.97%
-5	4.247	5.30%	1.68%
0	3.274	5.13%	1.37%
5	2.544	4.97%	1.07%
10	1.992	4.81%	0.78%
15	1.572	4.67%	0.50%
20	1.250	4.53%	0.24%
25	1.000	4.39%	0.00%
30	0.8056	4.26%	0.21%
35	0.6530	4.14%	0.40%
40	0.5326	4.02%	0.56%
45	0.4369	3.91%	0.69%
50	0.3604	3.80%	0.80%
55	0.2989	3.69%	0.87%
60	0.2491	3.59%	0.92%
65	0.2087	3.49%	0.93%
70	0.1756	3.40%	0.92%
75	0.1485	3.31%	0.88%
80	0.1261	3.23%	0.81%
85	0.1075	3.14%	0.72%
90	0.09209	3.06%	0.59%
95	0.07916	2.99%	0.45%
100	0.06831	2.91%	0.28%
105	0.05916	2.85%	0.08%
110	0.05141	2.77%	0.12%
115	0.04483	2.70%	0.36%
120	0.03922	2.64%	0.61%
125	0.03442	2.57%	0.87%
130	0.03030	2.51%	1.16%
135	0.02675	2.47%	1.46%
140	0.02369	2.41%	1.82%
145	0.02103	2.35%	2.14%
150	0.01872	2.35%	2.46%

BOWTHORPE THERMOMETRICS
Crown Industrial Estate, Priorswood Road
Taunton, Somerset TA2 8QY UK
Tel +44 (0) 1823 335200
Fax +44 (0) 1823 332637

THERMOMETRICS, INC.
808 US Highway 1
Edison, New Jersey 08817-4695 USA
Tel +1 (732) 287 2870
Fax +1 (732) 287 8847

KEYSTONE THERMOMETRICS CORPORATION
967 Windfall Road
St. Marys, Pennsylvania 15857-3397 USA
Tel +1 (814) 834 9140
Fax +1 (814) 781 7969