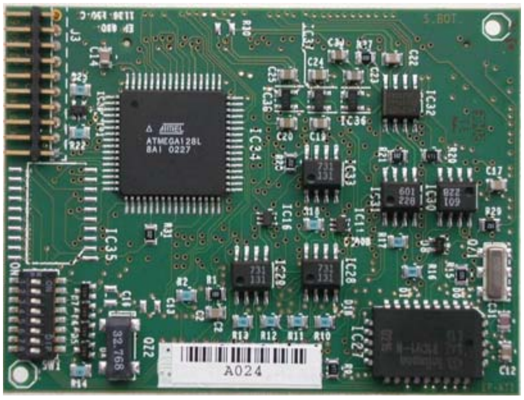
CANopen Application Software for the ELMB128

(Embedded Local Monitor Board)



(approx. true size)

Henk Boterenbrood
NIKHEF, Amsterdam
29 Nov 2006

Version 2.3

ABSTRACT

*The **ELMB128** is designed as a general-purpose plug-on module for distributed monitoring and control applications in the ATLAS experiment. After production it contains a CANopen application program for doing digital and analog input and output.*
This document provides a description and user manual of the application and includes a full listing of its Object Dictionary.

Contents

Version History		
Version	Date	Comments
2.3	29 Nov 2006	Describes *ELMBio* v4.4.2 for ELMB128: - Added remotely configurable CAN node identifier. - Added objects to ADC Configuration (OD 2100h), enabling multiple voltage ranges. - Added table with ADC resolution in micro-volts. - Digital in- and outputs increased to 24 (3*8). - Added SPI object 2600, sub 5 (bytes read while writing to SPI).
2.2	13 January 2005	Describes *ELMBio* v4.3.3 for ELMB128: - Fix error in EEPROM memory map (Table 5). - Added detailed EEPROM memory maps for Serial Number and calibration constants.
2.1	19 May 2004	Describes *ELMBio* v4.3.3 for ELMB128: - ADC 'delta' mode of readout now uses *exceed* counter too. - Objects for general-purpose SPI access added to OD. - Toggle bit added to Emergency message. - Life Guarding enabled by default (70 s)
2.0	12 Nov 2003	First version describing *ELMBio* v4.2 for ELMB128.
1.x	…	Versions describing *ELMBio* v3.x for ELMB.

Table 1. Document change record.

1 Introduction

The ***Embedded Local Monitor Board*** (**ELMB**) is a plug-on board designed for the ATLAS experiment, where it will be used for a range of different control and monitoring tasks.

Full details about the ELMB hardware can be obtained from the ELMB web pages[1].

During production the ELMB's ATmega128 microcontroller [3] flash memory is programmed with two application programs:

1. a Bootloader:
   in the ELMB microcontroller the upper 8 Kbytes of the 128 Kbytes flash memory is reserved for the so-called ***Bootloader***, a separate application that takes care the *In-Application-Programming*. At the time of production of the ELMB, a Bootloader is installed, called **ELMBbl**, which enables reprogramming of the ELMB microcontroller via the CAN-bus using the ***CANopen*** protocol [1] [2]. The ELMB Bootloader is described in a separate document.
2. a ***CANopen*** I/O application for production acceptance testing as well as subsequent use as a general-purpose analog/digital input and output application. This CANopen application, called **ELMBio**, is the subject of this document.

ELMBio has been developed to provide users of the ELMB with a ready-to-use CANopen module, when plugged onto the ELMB Motherboard.
It supports, by means of the ELMB's on-board ADC and multiplexors –if present–, 64 analog input channels, up to 24 digital inputs (microcontroller I/O ports PORTF, PORTA and PORTC) and up to 24 digital outputs (microcontroller I/O ports PORTC, PORTA and PORTF); each digital I/O can be configured as either input or output. In addition it has support for an external DAC-module (a separate module), and can handle up to 64 analog outputs

The **ELMBio** application conforms where possible to the CANopen DS-401 Device Profile for I/O-modules [5].

The so-called "process data" –in the case of ELMBio, the analog and digital inputs and outputs– can efficiently be read out (or written to) using CANopen **PDO** (Process Data Object) messages. A PDO message is a non-confirmed CAN-message with one sender and one or more receivers, containing no protocol overhead, only data (1 to 8 bytes). Receivers of a PDO message know the meaning of the data content of a PDO message (in any case, receivers should also be able to find out about the data content of a PDO by consulting the *PDO Mapping Parameters* in the Object Dictionary of the producer of the PDO).

ELMBio supports a total of 5 PDOs: two Transmit-PDOs for the analog inputs (one for raw ADC-counts and one for physical values in µVolts), one Transmit-PDO for the digital inputs, one Receive-PDO for the digital outputs and a Receive-PDO for the analog outputs.

For an overview of the complete Object Dictionary (*OD*) of the **ELMBio** application see section 4.

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

Many of the application's features are configurable using standard CANopen messages. Settings can be stored permanently in on-board EEPROM, also using standard CANopen messages.

ELMBio provides, apart from the standard CANopen and CANopen Device Profile features, additional support for *In-Application-Programming* via the CAN-bus through interaction with the Bootloader, and is equipped with a number of mechanisms to decrease the sensitivity of the running application to *SEE* (Single Event Effects) due to radiation.

The source code of **ELMBio** source code is freely available for users who want to customize the application to fit their needs. Alternatively, there is also a CANopen firmware framework for the ELMB available, for users who need to develop custom I/O and control themselves, but want to have the benefit of a ready-to-use framework that handles all CAN and CANopen communication.

2 Hardware Configuration

Using the onboard DIP-switches a node identifier may be set between 1 and 63 (the identifier must be unique on the CAN-bus the module is on), using 6 of the 8 switches, and a CAN-bus bit rate of 50, 125, 250 or 500 kbits/s, using the 2 remaining switches. See Figure 1 below for details.

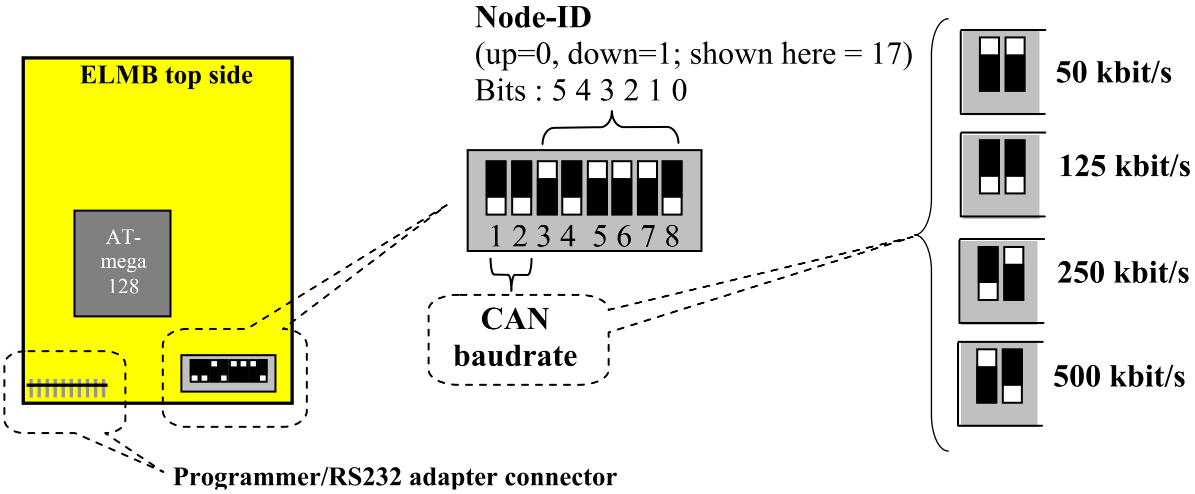


Figure 1. Location and function of ELMB DIP-switches and the 10-pin Programmer/RS232 adapter connector.

(Note that **ELMBio version 4.4.0** and newer supports a optional remotely configurable CAN node identifier, which –if set– overrides the DIP-switch setting).

Table 2 shows the functions of the ELMB microcontroller I/O pins. They match the I/O-pin layout for an ELMB plugged on the ELMB Motherboard (version 3).

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	*in8/out8*	—	*in16/out0*	—	—	*in0/out16*
pin 1	*in9/out9*	*SCLK*	*in17/out1*	—	—	*in1/out17*
pin 2	*in10/out10*	*SDI*	*in18/out2*	—	—	*in2/out18*
pin 3	*in11/out11*	*SDO*	*in19/out3*	*ADC_CS*	*DAC_CS1*	*in3/out19*
pin 4	*in12/out12*	—	*in20/out4*	*ADC_SCLK*	*DAC_CS2*	*in4/out20*
pin 5	*in13/out13*	—	*in21/out5*	*ADC_SDI*	*DAC_CS3*	*in5/out21*
pin 6	*in14/out14*	—	*in22/out6*	*ADC_SDO*	*DAC_CS4*	*in6/out22*
pin 7	*in15/out15*	—	*in23/out7*	*ADC_MUX*		*in7/out23*

Table 2. I/O pin functions on the ATmega128 processor in the ***ELMBio*** application (use the ELMB in combination with ELMB Motherboard v3):

— = used for various ELMB-specific functions (DIP-switch readout, etc).
SCLK/SDI/ SDO = lines carrying *SPI*-protocol for the onboard CAN-controller [4]; available externally; do *not* use for devices with slow opto-couplers.
AD_xxx = SPI signals for the ELMB onboard ADC and external DAC-modules.
ADC_xxx = control signals for the ELMB onboard ADC.
DAC_CSx = select signals for the external DAC modules.
inn, outn = digital input *n* $(0<n<23)$ or digital output *n* $(0<n<23)$, respectively; pins are individually configured as either input or output.

3 The *ELMBio* Application

3.1 Initialisation

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:

ELMBio (NMT-Slave) → Host (NMT-Master)

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

NodeID is the CAN node identifier set by means of the ELMB onboard DIP-switches, which according to the CANopen standard must be in the range between 1 and 127 and for **ELMBio** can be to set to a value between 1 and 63, as shown in Figure 1.

ELMBio version 4.4.0 and newer supports an optional remotely configurable CAN node identifier (using standard CANopen messages), stored in ELMB onboard EEPROM, overriding the DIP-switch setting. See the description of *OD* index 3300h and 3301h for details.

To *start* the **ELMBio** application in the CANopen sense of the word, the following CANopen NMT (*Network ManagemenT*) message must be sent:

Host (NMT-Master) → ELMBio (NMT-Slave)

COB-ID	DataByte 0	DataByte 1
000h	1 (*Start_Remote_Node*)	*NodeID* or 0 (all nodes on the bus)

There is no reply to this message.

Now **ELMBio** is *Operational*, meaning that it monitors I/O channels as required and sends and receives (and processes) ***PDO*** messages (carrying the application data).

Optionally a feature called *auto-start* may be enabled, so that **ELMBio** automatically and immediately 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 to **ELMBio** the following CANopen NMT message must be sent:

Host (NMT-Master) → ELMBio (NMT-Slave)

COB-ID	DataByte 0	DataByte 1
000h	1 (*Reset_Node*)	*NodeID* or 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 ELMB; it reports its presence by sending the following Emergency message (see also section 5):

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 5 or the ATmega128 datasheet [3]).

Having the Bootloader activated at power-up guarantees that it is always possible to download 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 **ELMBio** application. Alternatively, the Bootloader makes the jump to the **ELMBio** application immediately, if it receives an NMT *Reset-Node* message –as shown above- within this period.

3.2 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) → ELMBio (NMT-Slave)

COB-ID
700h + *NodeID*

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

   ELMBio (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:

   ELMBio (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 **ELMBio** application 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 **ELMBio** application resets its Life Guarding timer at each properly received message addressed to it.

Life Guarding is controlled through *OD* objects 100Ch and 100Dh. In **ELMBio** 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. **ELMBio** resets and reinitializes the CAN-controller, and (tries to) resume(s) normal operation, after sending an Emergency message (see section 5).

3.3 Analog Inputs

3.3.1 ADC Configuration

The ELMB's ADC [6] can be configured for full-scale measurement in the ranges 25 mV, 55 mV, 100 mV, 1V, 2.5V and 5V, unipolar or bipolar, with a programmable conversion word rate of 1.8 Hz, 7.5 Hz, 15 Hz, 30 Hz, 60 Hz, 85 Hz or 100 Hz (in practice the achievable rate of conversions is limited to about 30 Hz maximum due to the slow opto-couplers used in the (serial) interface between the processor and the ADC.

The ADC configuration in **ELMBio** can be read from and set in *OD* index 2100h. Note that starting with **ELMBio version 4.4.0** the ELMB's 64 analog input channels may be assigned to up to 4 different voltage ranges, in groups of 16 channels (see *OD* index 2100h, subindex 22 to 29).

3.3.2 Calibration Constants

During the ELMB production acceptance tests each of the ADC voltage ranges (for one conversion word rate, i.e. 15 Hz) is calibrated and the resulting calibration constants have been stored in the ELMB EEPROM. Depending on the voltage range(s) configured **ELMBio** applies the appropriate calibration constants. (After power-up or reset a so-called 'self-calibration' of the ADC is performed, resulting in certain register settings for gain and offset; the gain register settings are then multiplied with the appropriate stored calibration constants).

The calibration constants are also stored offline in a database, and can be retrieved on the basis of the serial number of the ELMB, which is a 32-bit number (actually, a 4-character string), stored in the ELMB's EEPROM, and also printed on a label located on the ELMB printed circuit board.

3.3.3 PDO Readout

ELMBio sends one PDO message for every analog input. It either sends PDO messages containing the ADC count or PDO messages containing the input voltage in microVolts.
The CAN-identifier used for the ADC readout in counts is the so-called *2nd-transmit-PDO* (**TPDO2**) of the CANopen Predefined Connection Set, i.e. COB-ID = 280h + *NodeID*.

The TPDO2 message is a 4-byte message and is formatted as follows:

ELMBio →Host

TPDO2 COB-ID	DataByte 0	DataByte 1	DataByte 2-3
280h+*NodeID*	Channel Number	Chan status+config	ADC value

with:

ADC value: 16-bits value, LSB in byte 2, MSB in byte 3.
Channel Number: number between 0 and 63.
Chan status+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.

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

The CAN-identifier used for the ADC readout in µV is the so-called *3rd-transmit-PDO* (**TPDO3**) of the CANopen Predefined Connection Set, i.e. COB-ID = 380h + *NodeID*.
The TPDO3 message is a 6-byte message and is formatted as follows:

ELMBio →Host

TPDO3 COB-ID	DataByte 0	DataByte 1	DataByte 2-5
380h+*NodeID*	Channel Number	Chan status+config	ADC value [µV]

with:
ADC value: 32-bits signed value in µV, LSB first.
Channel Number: number between 0 and 63.
Chan status+config: see above.

The ADC resolution in µVolts depends on its voltage range setting, taking into account that the ADC converts to 16-bit unipolar or bipolar. This is shown below in Table 3.

ADC voltage range	ADC resolution unipolar (bipolar) [µV]
25 mV	0.38 (0.76)
55 mV	0.84 (1.68)
100 mV	1.53 (3.05)
1 V	15.3 (30.5)
2.5 V	38.1 (76.3)
5 V	76.3 (152.6)

Table 3. ADC micro-volts resolution as a function of the ADC voltage range used.

The number of analog channels can be set to any value up to 64 by writing to *OD* index 2100h, sub 1.

The way in which all 64 (or less) analog inputs are read out depends on the *transmission-type* of TPDO2 or TPDO3. The analog inputs are read out according to the PDO transmission type after power-up. Alternatively the user can set the transmission type to the required value by writing to **ELMBio**'s Object Dictionary (to *OD* index 1801h, sub 2 or *OD* index 1802h, sub 2), and possibly stores it permanently in onboard EEPROM so that it will be the default transmission type after every subsequent reset or power-up.

The following modes of transmission are supported:

- **PDO transmission type 1**:
  after every so-called **SYNC** message issued on the CAN-bus, **ELMBio** starts an analog input channel scan and sends (up to) 64 TPDO messages, one message for every analog input channel, as shown above. An A/D conversion has to be done for every channel so it can take up to about 30 seconds before all TPDOs have been sent (the ADC conversion word rate can be set 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 <u>all</u> nodes that have PDOs configured to respond to a SYNC message will respond to it.
Note also that if both TPDO2 and TPDO3 have transmission type 1 only TPDO3 messages are produced (unless there are no (valid) calibration constants for the currently active ADC voltage range).

- **PDO transmission type 255 and Event Timer = 0**:
after every so-called Remote Transmission Request (**RTR**, or Remote Frame) for TPDO2/3, **ELMBio** starts an analog input channel scan and sends (up to) 64 TPDO2/3 messages, one message for every analog input channel. The CAN *Remote Frame* that constitutes this RTR has no data bytes:

Host → ELMBio

COB-ID
280h + *NodeID*

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

- **PDO transmission type 255 and Event Timer > 0**:
If TPDO2's *event timer* (*OD* index 1801h, sub 5) or TPDO3's *event timer* (*OD* index 1802h, sub 5) is set to a value unequal to zero (*event timer* is expressed in units of 1 s) **ELMBio** automatically starts an analog input channel scan (resulting in up to 64 TPDO2 or TPDO3 messages) triggered by a timer with a period equal to the *event timer* setting (in this mode an RTR also triggers such an input scan). If the timer expires while a channel scan is still in progress, the trigger is ignored until the next time the timer expires.

Optionally **ELMBio** performs a reset and calibration sequence before each ADC channel scan. This feature is controlled via *OD* index 2120h (which may be useful for increasing radiation tolerance of the ADC readout).

3.3.4 SDO Readout

Analog inputs can also be read individually using CANopen **SDO** messages (see *OD* index 6404h for readout of ADC channels in ADC counts and *OD* index 2404h for readout of ADC channels in μV). Note that the data in these objects contain a 'flags' byte (generated by the ADC hardware), 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.

3.3.5 Readout on Change

ELMBio has 2 modes of *readout-on-change* for analog inputs: ***delta-change*** mode and ***window*** mode. These modes can be enabled individually and both may be enabled at the same time. Use *OD* index 2130h to enable or disable *delta-change* mode and *OD* index 2140h to enable or disable *window* mode.

When the global *readout-on-change* interrupt for analog inputs is enabled (*OD* index 6423h set to 1) and any one of the *delta-change* or *window* modes is enabled, **ELMBio** starts a continuous loop doing conversions of the number of configured ADC channels (*OD* index 2100h, sub 1) as soon as it is put into *Operational* state.

Now every time a channel's 'status' changes (depending on the mode and settings), a TPDO3 is generated (containing the ADC channel reading in μV). *Readout-on-change* never generates a TPDO2 (with ADC reading in counts), unless the calibration constants are not present or invalid for the currently active ADC voltage range). If both modes are enabled and a channel satisfies both *readout-on-change* conditions in the same channel scan cycle, only one message is sent for this channel.

3.3.5.1 Delta-Change Mode

In *delta-change* mode analog input read-out **ELMBio** asynchronously sends a message when an analog input channel reading has increased or decreased by a certain preset amount. When the change is detected a message is sent only after a configurable number of consecutive readings confirm the change. We call this number the *exceed counter*. There is only one counter for all channels, set in *OD* index 2150h (value must be in the range 1 to 254).

After the message has been sent the current analog reading is taken as the new reference value for this channel in the scanning loop.

The analog channel input reference values are stored in RAM but not protected against SEE, since it is not necessarily considered a bad thing when an analog input channel reading is sent again just because the reference value in RAM has been corrupted by chance. The (corrupted) reference value is immediately overwritten by the new reading.

When the node is put into *Operational* state the ADC reference values are initialized by the readings from the first ADC-channel scan cycle.

An explicit request for data such as a SYNC or a RTR (Remote Frame) message stops/aborts the ongoing channel scan cycle and starts a new scan cycle in which all analog input values are forcibly sent as TPDO2 or TPDO3 messages (i.e. raw ADC counts or voltage values, depending on the TPDO2 and TPDO3 transmission modes and the RTR that triggered the action), i.e. a 'forced-readout' ADC scan cycle is started.

In addition, if the transmission mode of the appropriate PDO is set to 255 and the PDO Event Timer set to a value greater than zero the scan cycle is aborted at regular intervals according to the configured interval and a 'forced-readout' scan cycle is started. A 'forced-readout' scan cycle does not affect the currently stored analog input channel reference values.

The *delta-change* parameter (the amount by which an ADC input channel value has changed when its reading is sent in a message) can be changed on a per-channel basis (on-the-fly) and can be written or read, expressed only in units of µVolt (by writing to or reading from *OD* index 6426h). A value of zero for the *delta-change* parameter effectively disables the check for the channel in question.

In addition, the *delta-change* parameter for all channels can be set to the same value in one write-operation to *OD* index 6426h using subindex 255 (=FFh).

These parameters are stored onboard in non-volatile memory on request, in the CANopen standard way.

The *delta-change* mode of analog input readout is illustrated in Figure 2.

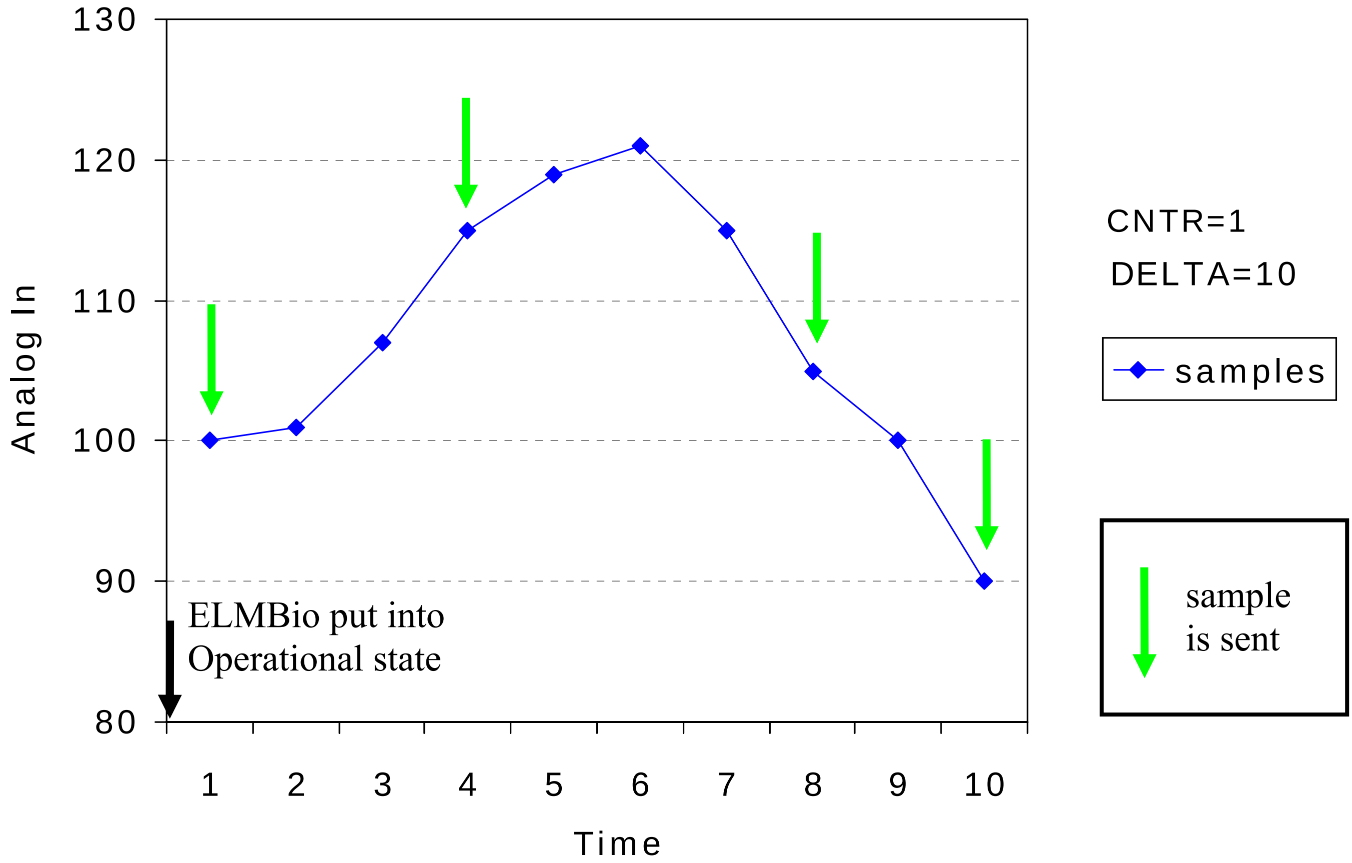


Figure 2. Illustration of the analog input *delta-change* mode. The delta-change parameter is set to 10. Analog input samples marked by an arrow are sent. The *exceed counter* in this example is set to 1.

3.3.5.2 Window Mode

In *window* mode analog input read-out **ELMBio** asynchronously sends a message when an analog input channel reading has gone below a certain preset lower limit or has exceeded a certain preset upper limit.

The response to a SYNC message or an RTR (Remote Frame) is similar to the behaviour described in the section on the *delta*-mode.

The *upper*- and *lower-limit* parameter can be changed on a per-channel basis (on-the-fly) and can be written or read, expressed only in units of µVolt (by writing to or reading from *OD* index 6424h or 6425h resp.). Upper and lower limit must differ by at least 1 ADC count to work, so this minimum difference in µVolt varies according to the ADC configuration.
In addition the *upper*- and *lower-limit* parameters for all channels can be set to the same value in one write-operation to *OD* index 6424h and 6425h resp. using subindex 255 (=FFh).
These parameters are stored onboard in non-volatile memory on request, in the CANopen standard way.

After a channel's reading has crossed a limit –either going outside the window or going back inside the window– a single message is sent. When going outside the window a message is sent only after a configurable number of consecutive readings outside the set window. We call this number the *exceed counter*. There is only one counter for all channels, set in *OD* index 2150h (value must be in the range 1 to 254). Note that this same counter applies to the delta-change mode.

When the channel reading returns inside the window a message is sent immediately (but only if the 'outside-window' situation was reported !).

Two readings are consecutive when they occur in 2 consecutive channel scan cycles. Note that if 64 ADC channels are scanned, there may be considerable time between 2 consecutive readings of the same channel (in the order of several seconds, depending on the number of channels in the scan cycle and the ADC conversion wordrate used).

The *window* mode of analog input readout is illustrated in Figure 3.

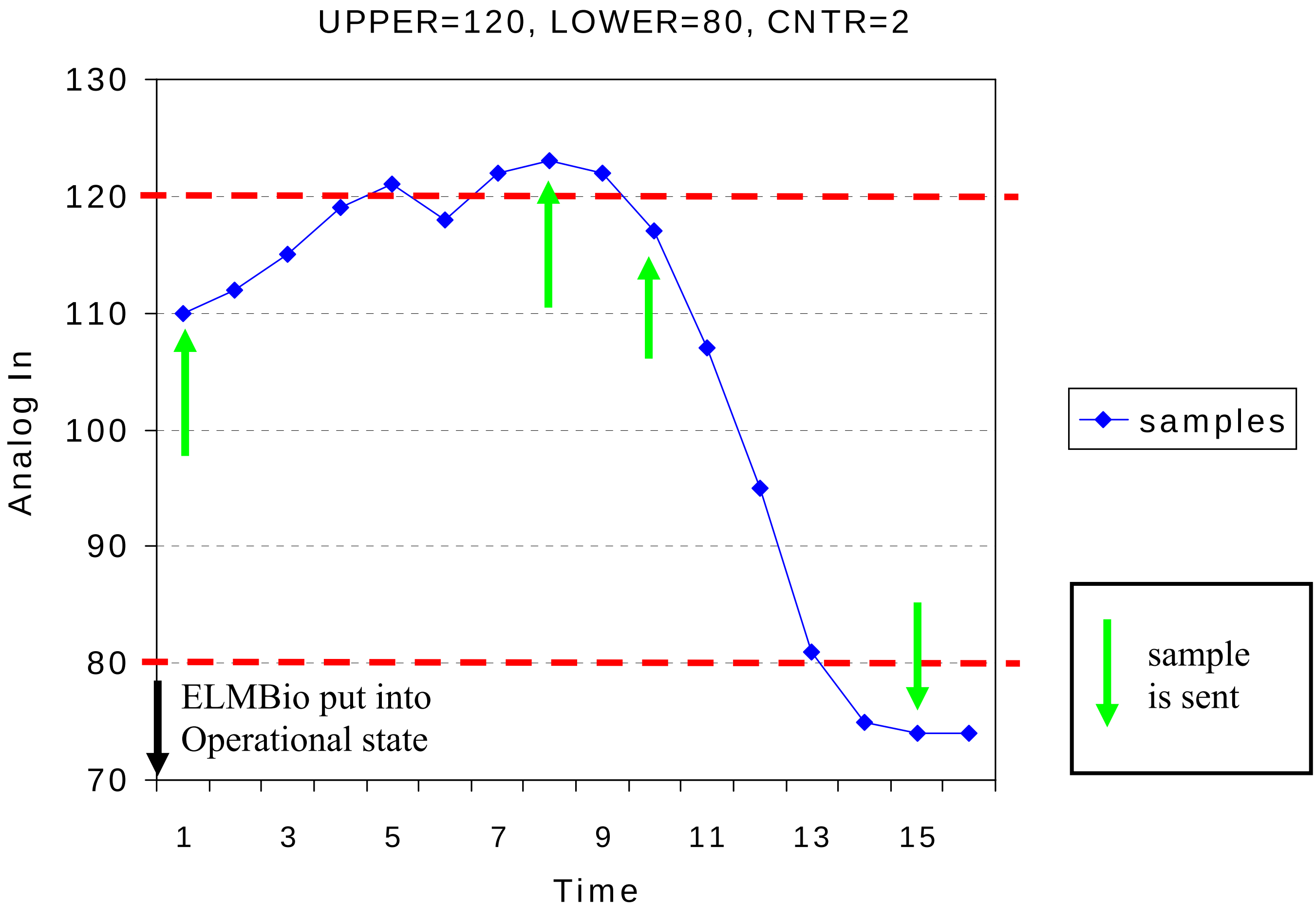


Figure 3. Illustration of the analog input *window* mode. The upper limit of the window is set to 120 and the lower limit to 80. The *exceed counter* is set to 2. Analog input samples marked by an arrow are sent.

3.4 Digital Inputs

The digital inputs are read out using the CANopen **PDO** mechanism. The CAN-identifier used for this PDO is the so-called *1st-transmit-PDO* (**TPDO1**) of the CANopen Predefined

Connection Set, which is the default PDO used for digital inputs according to the CANopen *Device Profile for I/O Modules* [5], i.e. COB-ID = 180h + *NodeID*. In this application TPDO1 contains 3 data bytes containing the state of the 3x8 digital inputs:

ELMBio →Host

TPDO1 COB-ID	DataByte 0	DataByte 1	DataByte 2
180h+*NodeID*	8-bit Digital Input (PORTF in)	8-bit Digital Input (PORTA in)	8-bit Digital Input (PORTC in)

The following modes of transmission are supported:

- **PDO transmission type 1:**
  after every so-called **SYNC** message issued on the CAN-bus **ELMBio** sends a TPDO1.

- **PDO transmission type 254/255 and Event Timer = 0**:
  ELMBio sends a TPDO1 after every so-called Remote Transmission Request (**RTR**, or Remote Frame) for this PDO.

- **PDO transmission type 254/255 and Event Timer > 0**:
  If TPDO1's *event timer* (*OD* index 1800h, sub 5) is set to a value unequal to zero (*event timer* is expressed in units of 1 ms, but here its value is truncated to a multiple of 1000) **ELMBio** automatically sends a TPDO1 on a regular basis triggered by a timer (TPDO1 is also sent after a RTR).

Automatic sending of a TPDO1 at 'change-of-state' of the digital inputs can be enabled through *OD* index 6005h; it is disabled by default.
If enabled, in each of the transmission modes listed above, **ELMBio**, once put into state *Operational,* continuously monitors the state of the digital I/O inputs and immediately sends a TPDO1 after it detects a change in any of the inputs. A debounce time-out is in effect and can be set (also to zero; see OD index 2200h). **ELMBio** polls the digital inputs roughly about every 0.5 ms, also depending on other activities.

All three digital input ports are shared between digital in- and outputs, configurable on a bit-by-bit basis. See the next section on how to define whether a bit is input or output. Bits defined as *output* show up as zeroes in data bytes in the TPDO1 message shown above.
Note: the ports have pull-up resistors enabled in their input circuits.

There is an interrupt mask for each input bit: if set, a change detected on the corresponding input will trigger a TPDO1 message (provided the *global digital input interrupt enable* mentioned above in *OD* index 6005h is set); the interrupt masks can be set in *OD* index 6006h, sub 1, 2 and 3.

Digital inputs can of course also be read using CANopen **SDO** messages (see *OD* index 6000h).

3.5 Digital Outputs

The digital outputs can be written using the CANopen **PDO** mechanism. The CAN-identifier used for this PDO is the so-called *1st-receive-PDO* (**RPDO1**) of the CANopen Pre-defined Connection Set, which is the default PDO used for digital outputs according to the CANopen *Device Profile for I/O Modules* [5], i.e. COB-ID = 200h + *NodeID*. RPDO1 has at least 1 data byte, containing in each byte the (required) state of 8 digital outputs:

Host → ELMBio

RPDO1 COB-ID	DataByte 0	DataByte 1	DataByte 2
200h + *NodeID*	8-bit Digital Output (PORTC out)	8-bit Digital Output (PORTA out)	8-bit Digital Output (PORTF out)

If RPDO1 carries only 1 data byte only PORTC gets new settings, PORTA remains unchanged.

Once **ELMBio** is put into state *Operational* it can receive the RPDO1 and immediately on reception sets its digital outputs according to the values in the RPDO1 data byte(s).

ELMBio retains the digital output settings only after a 'soft' reset (triggered by an NMT Reset-Node message). After a 'hard' reset (power-up, watchdog) the outputs are initialized to either 0 or 1 (low or high), which can be set by *OD* index 2300h.
As mentioned in the previous section the digital ports can be defined bit-by-bit as either input or output. This is done through the so-called Output Filter Mask (*OD* index 6208h): bits set to 1 in this mask are output; the other bits are automatically defined as input. Bits defined as *input* in byte 1 in the RDPO1 message shown above are ignored when setting the outputs.

Digital outputs can of course also be written to using CANopen **SDO** messages (see *OD* index 6200h).

3.6 Analog Outputs

Analog outputs are compatible with the DAC-module designed and built for ATLAS DCS applications (described elsewhere), either equipped with MAX5122 DACs or MAX525 DACs. With the MAX5122 one DAC-module has 4 channels, when equipped with the MAX525 one DAC-module has 16 channels. By default ELMBio assumes MAX5122-type DACs. By setting the proper parameter in *OD* index 2500h MAX525-type DACs can be selected. The two types of DAC-module cannot be mixed.
Four DAC-modules can be connected (i.e. directly to the 20-pin J8 connector on the ELMB Motherboard), for a total of up to 64 analog output channels when using MAX525 DACs, or 16 channels when using MAX5122 DACs.

The analog outputs can be written using the CANopen **PDO** mechanism. The CAN-identifier used for this PDO is the so-called *2nd-receive-PDO* (**RPDO2**) of the CANopen Pre-defined Connection Set, which is the default PDO used for analog outputs according to the CANopen *Device Profile for I/O Modules* [5], i.e. COB-ID = 300h + *NodeID*. RPDO2 has at least 3 data bytes, containing the DAC channel number and a 2-byte DAC-value:

Host → ELMBio

RPDO2 COB-ID	DataByte 0	DataByte 2-3
300h + *NodeID*	Channel Number	DAC value

Once **ELMBio** is put into state *Operational* it can receive the RPDO2 and on reception will immediately set analog outputs according to the values in the RPDO2 data byte(s).

Analog outputs can of course also be written to, using CANopen **SDO** messages (see *OD* index 6411h).

Note that MAX5122 DACs are 12-bit, but the DAC-value is set as a 13-bit value with bit 0 always equal to 0.

3.7 Storing Parameters and Settings

Parameters and settings can be stored permanently onboard (in an EEPROM) by writing string "save" to *OD* index 1010h. The CANopen **SDO** mechanism is used to do this:

Host → ELMBio

COB-ID	DataByte							
	0	1	2	3	4	5	6	7
600h + *NodeID*	23h	10h	10h	*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 PDO and Guarding).
= 3: store application parameters (concerning ADC, DAC and Digital I/O).
(check out the Object Dictionary tables in section 4 to find out which parameters are stored).

If the store-operation succeeded **ELMBio** sends the following reply:

ELMBio → Host

COB-ID	DataByte						
	0	1	2	3	4	5	6-7
580h + *NodeID*	60h	10h	10h	*subindex*	–	–	–

If the store-operation did NOT succeed **ELMBio** sends the following reply (SDO *Abort Domain Transfer*, error reason: 'hardware fault' (for details see [1])):

ELMBio → Host

COB-ID	DataByte							
	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. Default values are listed in the *OD* tables in section 4.

The tables with the Object Dictionary in section 4 show the settings stored in EEPROM as marked by an asterisk (*).

Note that storage of ELMB Serial Number and ADC calibration constants in EEPROM are handled separately.

See section 6 for an overview of the EEPROM usage.

4 Object Dictionary

The Object Dictionary (*OD*) of the **ELMBio version 4.4.0** application is listed in the tables on the next pages.

The values of objects marked with '*' in the *Index* column are stored in EEPROM for permanent non-volatile storage, on request. They are retrieved from EEPROM at reset and power-up.

Communication Profile Area						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
1000	-	Device type	U32	RO	000F0191h	Meaning: DSP-401 device profile, analogue in- and outputs, 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	"MA44"	***ELMBio*** application version 4.4 [2] (see footnote)
	1	minor version number	VisStr	RO	"0000"	
100C	-	Guard time [ms]	U16	RO	1000	= 1 second
100D *	-	Life time factor	U8	RW	70	Life Guarding timeout in seconds; 0 → no life guarding timeout

[1] Manufacturer Status Register bits:
00000001: ADC reset error,
00000002: ADC calibration error,
00000004: ADC conversion time-out,
00000008: error reading or writing ADC calibration constant(s),
00000010: error reading or writing ADC delta-change parameters,
00000020: error reading or writing ADC upper-limit parameters,
00000040: error reading or writing ADC lower-limit parameters.

[2] "**MA**": version using the ELMB onboard ADC, "**MV**": version using the ATmega128 on-chip 8-chan ADC, "**MN**": version without any ADC support.

Communication Profile Area

Index (hex)	Sub Index	Description	Data/ Object	Attr	Default	Comment
1010		Store parameters	Array			Save stuff in onboard EEPROM
	0	Highest index supported	U8	RO	5	
	1	Save all parameters	U32	RW	1	Read: 1; Write "save": store all (incl. ADC limits)
	2	Save communication parameters	U32	RW	1	Read: 1; Write "save": store PDO par's, Life time factor, …
	3	Save application parameters	U32	RW	1	Read: 1; Write "save": store ADC config, dig.I/O config, … (incl. ADC limits)
	4	Save ADC delta-change parameters	U32	RW	1	Read: 1; Write "save": store ADC deltas
	5	Save ADC upper/lower limit parameters	U32	RW	1	Read: 1; Write "save": store ADC upper/lower liimits

Index (hex)	Sub Index	Description	Data/ Object	Attr	Default	Comment
1011		Restore default parameters	Array			Invalidate stuff in onboard EEPROM; use defaults
	0	Highest index supported	U8	RO	5	
	1	Set all parameters to defaults	U32	RW	1	Read: 1; Write "load": invalidate all parameters stored (excl. ADC deltas/limits)
	2	Set communication parameters to defaults	U32	RW	1	Read: 1; Write "load": invalidate stored PDO par's, etc.
	3	Set application parameters to defaults	U32	RW	1	Read: 1; Write "load": invalidate stored ADC config, etc. (excl. ADC deltas/limits)
	4	Set ADC delta-change parameters to defaults	U32	RW	1	Read: 1; Write "load": invalidate ADC deltas
	5	Set ADC upper/lower limit parameters to defaults	U32	RW	1	Read: 1; Write "load": invalidate ADC upper/lower limits

Index (hex)	Sub Index	Description	Data/ Object	Attr	Default	Comment
1017 *	-	Producer Heartbeat Time [1 s]	U16	RW	0	In units of <u>seconds</u> (but ≤255 !), (NB: should be in ms according to CANopen!); 0 → Heartbeat is disabled

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*

Communication Profile Area *(continued…)*

Index (hex)	Sub Index	Description	Data/ Object	Attr	Default	Comment
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	Only 255 allowed
	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	

1600		1st Receive PDO mapping	Record			Data type = PDOMapping
	0	Number of entries	U8	RO	2	
	1	Digital outputs 1-8	U32	RO	62000108	OD index 6200, sub-index 1: Outputs 1-8 (see DSP-401), size = 8 bits
	2	Digital outputs 9-16	U32	RO	62000208	OD index 6200, sub-index 2: Outputs 9-16 (see DSP-401), size = 8 bits
	3	Digital outputs 17-24	U32	RO	62000308	OD index 6200, sub-index 3: Outputs 17-24 (see DSP-401), size = 8 bits
1601		2nd Receive PDO mapping	Record			Data type = PDOMapping
	0	Number of entries	U8	RO	2	
	1	DAC channel number	U32	RO	64110008	*actually not allowed, but…*
	2	16-bit analog output	U32	RO	64110110	OD index 6411, sub-index 1: 16-bits Analog Output (see DSP-401), size = 16 bits, multiplexed

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	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 seconds (NB: should be in ms according to CANopen!); active if >0 and transmission-type = 255
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 seconds (NB: should be in ms according to CANopen!); active if >0 and transm-type = 255

Communication Profile Area *(continued…)*

Index (hex)	Sub Index	Description	Data/ Object	Attr	Default	Comment
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*
*	5	Event timer [1 s]	U16	RW	0	In units of seconds (NB: should be in ms according to CANopen!); active if >0 and transmission-type = 255

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	Digital inputs 9-16	U32	RO	60000208h	OD index 6000, sub-index 2: Inputs 9-16 (see DSP-401), size = 8 bits
	3	Digital inputs 17-24	U32	RO	60000308h	OD index 6000, sub-index 2: Inputs 17-24 (see DSP-401), size = 8 bits

Index (hex)	Sub Index	Description	Data/ Object	Attr	Default	Comment
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	ADC channel number	U32	RO	64040008h	*actually not allowed, but…*
	2	24-bit analogue input	U32	RO	64040118h	OD index 6404, sub-index 1: Analogue inputs, multiplexed, size = 24 bits; actually the ADC flag bits (present in OD index 6404) have been replaced by a byte combining the ADC configuration and the two ADC error flags

Index (hex)	Sub Index	Description	Data/ Object	Attr	Default	Comment
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	ADC channel number	U32	RO	24040008h	*actually not allowed, but who cares… it's not important*
	2	32-bit analogue input	U32	RO	24040128h	Object 2404, sub-index 1: Analogue inputs in volts, multiplexed, size = 40 bits: actually the ADC flag bits (present in Object 2404) have been replaced by a byte combining the ADC configuration and the two ADC error flags; 24-bit data is replaced by a 32-bit signed long

Manufacturer-Specific Profile Area						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
2100		ADC-configuration	Record			CRYSTAL CS5523 16-bit ADC
	0	Number of entries	U8	RO	29	
*	1	Number of input channels	U8	RW	64	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] (all input channels [3])
*	4	Unipolar/Bipolar Measurement Mode	U8	RW	1	0 = bipolar, 1 = unipolar (all input channels [3])
	5	Power Save Mode	Bool	WO		1 = put ADC in 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	5000000	in µV
*	21	SPI SCLK signal high period (opto-coupler delay)	U8	RW	150	in µs, 10 ≤ value ≤ 255
*	22	Input Voltage Range (ADC physical chan #1)	U8	RW	4	3-bit code [2] (input channels 0-3, 16-19, 32-35 and 48-51)
*	23	Input Voltage Range (ADC physical chan #2)	U8	RW	4	3-bit code [2] (input channels 4-7, 20-23, 36-39 and 52-55)
*	24	Input Voltage Range (ADC physical chan #3)	U8	RW	4	3-bit code [2] (input channels 8-11, 24-27, 40-43 and 56-59)
*	25	Input Voltage Range (ADC physical chan #4)	U8	RW	4	3-bit code [2] (input channels 12-15, 28-31, 44-47 and 60-63)
*	26	Unipolar/Bipolar Mode (ADC physical chan #1)	U8	RW	1	(input channels 0-3, 16-19, 32-35 and 48-51)
*	27	Unipolar/Bipolar Mode (ADC physical chan #2)	U8	RW	1	(input channels 4-7, 20-23, 36-39 and 52-55)
*	28	Unipolar/Bipolar Mode (ADC physical chan #3)	U8	RW	1	(input channels 8-11, 24-27, 40-43 and 56-59)
*	29	Unipolar/Bipolar Mode (ADC physical chan #4)	U8	RW	1	(input channels 12-15, 28-31, 44-47 and 60-63)

[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

[3] Writing to subindex 3 or 4, overwrites subindices 22 to 25 or 26 to 29 respectively.

Manufacturer-Specific Profile Area *(continued…)*						
Index (hex)	**Sub Index**	**Name**	**Data/ Object**	**Attr**	**Default**	**Comment**
2110	-	ADC-reset-and-calibrate	U8	WO		Writing triggers a reset and calibrate sequence using the current ADC settings
2120 *	-	ADC-reset-and-calibrate before each channel scan	Bool	RW	0	If =1 a reset/calibration sequence is performed before every ADC input channel scan
2130 *	-	Enable Analogue Input Interrupt Delta-change Mode	Bool	RW	1	Enable/disable *delta* mode readout-on-change operation (global enable: Object 6423)
2140 *	-	Enable Analogue Input Interrupt Upper/Lower Limit Mode	Bool	RW	0	Enable/disable *window* mode readout-on-change operation (global enable: Object 6423)
2150 *	-	Upper/Lower Limit and Delta Exceed Counter	U8	RW	2	Number of consecutive readout values outside window before value is sent [1] (1≤value≤254)
2200 *	-	Digital Input debounce timer	U8	RW	10	In units of ca. 0.5 ms (set to 0 there is ca. 0.5 ms between consecutive input polls).
2300 *	-	Digital Output Init High	Bool	RW	1	After a hard reset: if set to 0 Digital Outputs will be initialised to all low; if set to 1 Digital Outputs will be initialised to all high
2404		Read Analogue Input Calibrated	Record			8 bits flags [2], 24 bits analogue value, in µV NB: read-out fails if there are no valid stored calibration constants for the current ADC range
	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 [2] + 24-bit (signed) data
	2	Input 2	U32	RO		2nd " " " "
	…	…	…	…		…
	…	…	…	…		…
	64	Input 64	U32	RO		64th " " " "

1 'Consecutive' here means: in consecutive input channel scans (of up to 64 channels).
2 See section 3.3.4 for a description of the ADC 'flags' byte.

Manufacturer-Specific Profile Area *(continued…)*						
Index (hex)	**Sub Index**	**Name**	**Data/ Object**	**Attr**	**Default**	**Comment**
2500		DAC configuration	Record			
	0	Number of entries	U8	RO	3	
	1	Number of output channels	U8	RO	16 or 64	=16 when MAX5122 DAC used, =64 when MAX525 DAC used
*	2	MAX525 DAC Type Select	Bool	RW	0	0: DAC-type is MAX5122 1: DAC-type is MAX525
*	3	SPI SCLK signal high period (opto-coupler delay)	U8	RW	75	in µs, 10 ≤ value ≤ 255
2600		SPI access	Record			Low-level access to the SPI-interface as defined for the ELMB-DAC (available on the Motherboard J8 connector). Beware of interfering with ADC operations ! (**v4.3+**)
	0	Number of entries	U8	RO	5	
	1	read or write 1 byte	U32	RW		MSB read or written first!
	2	read or write 2 bytes	U32	RW		"
	3	read or write 3 bytes	U32	RW		"
	4	read or write 4 bytes	U32	RW		"
	5	read up to 4 bytes read during previous write operation	U32	RW		" (**v4.4.2+**)
2601		SPI chip-select (active LOW)	U8	RW		0 = deselect all *n* = activate CS*n* (1<=*n*<=4) (**v4.3+**)
2602		SPI configuration	Record			(**v4.3+**)
	0	Number of entries	U8	RO	2	
	1	SPI SCLK signal high (low) period	U8	RW	10	opto-coupler delay in µs, 10 <= value <= 255
	2	SPI SCLK Rising Edge	Bool	RW	1	SDI data shifted on rising (=1) or falling edge (=0)

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, determined at production time; always stored in EEPROM; enable write operation 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 A5h enables one write or erase operation to any of the Objects 2B00h to 2B05h or 2C00h to 2C05h.

[1] In other words: reset the ADC and do a 'self-calibration', i.e. do **NOT** apply the gain factors ('calibration constants'), which already might have been stored in EEPROM. This type of ADC initialisation is essential if the voltage range in question ever needs to be *re*calibrated.

Manufacturer-Specific Profile Area ***(continued…)***

Index (hex)	Sub Index	Description	Data/ Object	Attr	Default	Comment
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 in flash results in SDO *Abort* with *Error Code* 1; error while accessing FLASH results in SDO *Abort* with *Error Code* 6.
	2	*not used*	U16	RO	0	
	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 3100h).
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	A counter is decremented every 1s and incremented every time bus-off occurs, but if it reaches this maximum value the node abandons regaining CAN-bus access at bus-off; if value=255 the node retries indefinitely.
	4	Received message counter	U8	R		Counts received CAN messages modulo 256 (for debug purposes)
3300	-	CAN Node Identifier	U8	WO		The new CAN Node Identifier is used after the next reset. (The ELMB must have *Bootloader* firmware version 1.3 or newer, 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 [4] 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 *(continued…)*						
Index (hex)	**Sub Index**	**Description**	**Data/ Object**	**Attr**	**Default**	**Comment**
5C00	-	Compile Options	U32	RO		Bitmask denoting which compile options were used when the application was generated (see table below for details)
5DFF		ELMB Tests	Record		**EXPERT ONLY**	*For use in ATLAS DCS production and test stand only*
	0	Number of test objects	U8	RO	2	
	1	Test of I/O-pins	U32	RO	00000000h	see description in another doc
	2	Generate Watchdog Timer reset	U32	RO	-	ELMBio goes into an endless loop
5E00	-	Transfer control to ELMB Bootloader	U8	WO		ELMBio jumps to the Boot-loader application

Object 5C00: Compile Options		
Bit	***Compile Option***	***Comment***
0	ALL_MOTHERBOARDS	assume ELMB is plugged on Motherboard v3; if this fails try assuming ELMB is plugged on Motherboard v1/v2 (no option = Motherboard v3)
1	MOTHERBOARD1	assume ELMB is plugged on Motherboard v1, v2 (no option = Motherboard v3)
2	–	*(was option ADC_ELMB)*
3	ADC_AVR	use the ATmega128 processor's integrated on-chip 8-chan 10-bit ADC, instead of the ELMB's onboard 64-chan 16-bit ADC (type CS5523)
4	ADC_NONE	no ADC used
5	7BIT_NODEID	only DIP-switch 1 used for CAN baudrate (125 or 250 kbaud); other 7 switches used for setting the Node-ID: 1-127 (when this option is not set a 6-bit Node-ID is used and 2 bits are used for selecting a baudrate) NB: can not be used, as it clashes with the DIP-switch usage by the Bootloader.
6	RS232	include stuff to be able to use 'printf()' and such; requires the *Programmer* or other RS232 adapter to be connected to the ELMB programmer connector
7	ELMB103	the ELMB is an ELMB103 type (with ATmega103 processor); by default an ELMB128 (with ATmega128 processor) is assumed
8	VARS_IN_EEPROM	store/retrieve working copies of configuration parameters in/from EEPROM
9	–	*(was option HEARTBEAT)*
10	INCLUDE_TESTS	include an OD object through which (board) tests can be executed
11	–	(was option EEPROM_UINT16_ADDRESSES)
12	CAN_REFRESH	refresh CAN-controller descriptor register (at each buffer write/read)
13	2313_SLAVE_PRESENT	there is (probably) a Slave processor (usually when using an ELMB103, so in combination with compile option *ELMB103* shown above); this includes the code that deals with the Slave processor

Table 4. Optional compile-time macro defines, which can be read from Object 5C00h.
(in the source code individual options are surrounded by a double underscore '__').

Standardized Device Profile Area (according to CiA-*DS4O1*)						
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	2	
	1	Read inputs 1-8	U8	RO		ELMB PORTF; see Object 6208, 3
	2	Read inputs 9-16	U8	RO		ELMB PORTA; see Object 6208, 2
	3	Read inputs 17-24	U8	RO		ELMB PORTC; see Object 6208, 1
6005 *		Global Digital Input Interrupt Enable	Bool	RW	0	Enable/disable *change-of-state* TPDO1 transmission
6006		Interrupt Mask Any Change 8 input lines	Array			Only bits set to 1 will generate a TPDO1 on change
	0	Number of 8-bit inputs	U8	RO	2	
*	1	Interrupt Mask Inputs 1-8	U8	RW	FFh	
*	2	Interrupt Mask Inputs 9-16	U8	RW	FFh	
*	3	Interrupt Mask Inputs 17-24	U8	RW	FFh	
6200		Write state 8 Output lines	Array			
	0	Number of 8-bit outputs	U8	RO	2	
	1	Write outputs 1-8	U8	RW		ELMB PORTC
	2	Write outputs 9-16	U8	RW		ELMB PORTA
	3	Write outputs 17-24	U8	RW		ELMB PORTF
6208		Filter Mask 8 output lines	Array			
	0	Number of 8-bit masks	U8	RO	2	
*	1	Filter mask outputs 1-8	U8	RW	FFh	PORTC pins not defined as outputs (maskbit=1) are inputs, to be accessed thru Object 6000, 3
*	2	Filter mask outputs 9-16	U8	RW	FFh	PORTA pins not defined as outputs (maskbit=1) are inputs, to be accessed thru Object 6000, 2
*	3	Filter mask outputs 17-24	U8	RW	00h	PORTF pins not defined as outputs (maskbit=1) are inputs, to be accessed thru Object 6000, 1
6404		Read Analogue Input manufacturer-specific	Record			8 bits flags [1], 16 bits analogue value
	0	Number of analog inputs	U8	RO	64	Fixed, but actual hardware configuration may vary (see OD index 2100, sub 1)
	1	Input 1	I24	RO		1st analog input:8-bit flags [1] + 16-bit data
	2	Input 2	I24	RO		2nd " " " "
	…	…	…	…		…
	…	…	…	…		…
	64	Input 64	I24	RO		64th " " " "

[1] See section 3.3.4 for a description of the ADC 'flags' byte.

Standardized Device Profile Area (according to CiA-*DS4O1*)

Index (hex)	Sub Index	Description	Data/ Object	Attr	Default	Comment
6411		Write Analogue Out 16-bit	Array			
	0	Number of 16-bit outputs	U8	RO	16 or 64	=16 when MAX5122 DAC used, =64 when MAX525 DAC used (see OD index 2500)
	1	Output 1	U16	RW		1st analog output:16-bit
	2	Output 2	U16	RW		2nd analog output:16-bit
	…	…	…	…		…
	…	…	…	…		…
	64	Output 64	U16	RW		64th " " "
6423 *		Global Analog Input Interrupt Enable	Bool	RW	0	Enables/disables *readout-on-change* TPDO3 transmissions
6424		Analogue Input Interrupt Upper Limit	Array			(**v4.2+**)
	0	Number of analog inputs	U8	RO	64	
*	1	Input 1	I32	RW	-1	Voltage in µV (signed)
*	2	Input 2	I32	RW	-1	Voltage in µV (signed)
*	…	…	…	…		…
*	64	Input 64	I32	RW	-1	Voltage in µV (signed)
	255	All Inputs (1 to 64)	I32	WO	-1	Voltage in µV (signed)
6425		Analogue Input Interrupt Lower Limit	Array			(**v4.2+**)
	0	Number of analog inputs	U8	RO	64	
*	1	Input 1	I32	RW	0	Voltage in µV (signed)
*	2	Input 2	I32	RW	0	Voltage in µV (signed)
*	…	…	…	…		…
*	64	Input 64	I32	RW	0	Voltage in µV (signed)
	255	All Inputs (1 to 64)	I32	WO	0	Voltage in µV (signed)
6426		Analogue Input Interrupt Delta Unsigned	Array			
	0	Number of analog inputs	U8	RO	64	
*	1	Input 1	U32	RW	0	Voltage in µV (unsigned)
*	2	Input 2	U32	RW	0	Voltage in µV (unsigned)
*	…	…	…	…		…
*	64	Input 64	U32	RW	0	Voltage in µV (unsigned)
	255	All Inputs (1 to 64)	U32	WO	0	Voltage in µV (unsigned)

5 Emergency Objects

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

ELMB → 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 **ELMBio** version 4.3 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 may be generated by the **ELMBio** application (note that byte 2 containing the Error Register is not included in the table):

Error Description	**Emergency Error Code** (byte 1-0)	**Manufacturer-Specific Error Field** (byte 3-7)
CAN communication	8100h	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	8130h	(CAN-controller has been reinitialized)
RPDO: too few bytes	8210h	Byte 3: minimum DLC (Data Length Code) required
ADC: conversion timeout	5000h	Byte 3: 01h Byte 4: ADC channel number (0..63) Byte 5: 0
ADC: reset failed	5000h	Byte 3: 02h Byte 4: 00h Byte 5: Error id [3]
ADC: offset calibration failed	5000h	Byte 3: 03h Byte 4: 00h
ADC: gain calibration failed	5000h	Byte 3: 04h Byte 4: 00h
ADC problem(s) during initialisation	5000h	Byte 3: 10h Byte 4: ADC status (see OD index 1002)
ADC calibration constants: not available	5000h	Byte 3: 11h

…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] **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)	Manufacturer-Specific Error Field (byte 3-7)
Slave processor not responding (ELMB103 only)	5000h	Byte 3: 20h
CRC error	5000h	Byte 3: 30h Byte 4: 1 (program FLASH), 2 (Slave FLASH; ELMB103 only)
EEPROM: write error	5000h	Byte 3: 41h Byte 4: Parameter block index [1] Byte 5: = 0: while writing datablock info > 0: size of parameter block to write
EEPROM: read error	5000h	Byte 3: 42h Byte 4: Parameter block index [1] Byte 5: Error id (1=CRC, 2=length, 4=infoblock)
EEPROM: ADC-limits write error	5000h	Byte 3: 43h Byte 4: Parameter block ID [2] Byte 5: size of parameter block to write
Irregular reset (Watchdog, Brown-out or JTAG)	5000h	Byte 3: F0h Byte 4: microcontroller MCUCSR register contents [3]
Bootloader: not present	5000h	Byte 3: F1h
Bootloader is now in control [4]	5000h	Byte 3: FEh Byte 4: 01h Byte 5: 28h Byte 6: microcontroller MCUCSR register contents [3] Byte 7: 00h
Bootloader cannot jump to application: invalid [4]	6000h	Byte 3: FEh Byte 4: AAh Byte 5: AAh Byte 6: 00h Byte 7: 00h

[1] **0**: PDO communication parameters, **1**: Guarding parameters, **2**: ADC configuration, **3**: Digital I/O configuration, **4**: DAC configuration, **5**: CAN configuration parameters, **FEh**: Calibration constant(s), **FFh**: ELMB Serial Number.

[2] **1**: ADC delta-change values, **2**: ADC upper limits, **3**: ADC lower limits.

[3] 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.

[4] The Emergency message is actually generated by the Bootloader program !

Byte 2 of the Emergency message contains the value of the so-called *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 meaning 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

6 EEPROM Memory Map

Table 5 to Table 8 detail the layout of the ELMB's EEPROM in the **ELMBio** application. Addresses 800h - FFFh (2048 bytes) are not used.

EEPROM	ADDR	DESCRIPTION
not used	0000	
ELMBio configuration parameters	0001 – 00A1	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	00A2 – 00E8	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	00E9 – 00FF	
ELMB Serial Number [19]	0100 – 0106	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.
*reserved**	0107	
not used	0108 – 011F	
ELMB Analog-in calib consts [20]	0120 – 01DF	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).

Table 5. EEPROM memory map for *ELMBio* application (addresses 000h - 1DFh) (continued in the next table).

* reserved location for a remotely-configurable Node-ID, if an ELMB user application wishes to support this; a valid number in this location will be used by the Bootloader application (version 1.3), overruling the DIP-switch setting.

[19] See Table 7.
[20] See Table 8.

EEPROM	ADDR	DESCRIPTION
not used	01E0 01FF	
Deltas working copy	0200 02BF	Holds a 3-byte value (unsigned µVolts) for each analog input, i.e. 64x3 bytes = 192 bytes.
not used	02C0 02FF	
Deltas permanent storage + CRC	0300 03C2	Holds a 3-byte value (unsigned µVolts) for each analog input, i.e. 64x3 bytes = 192 bytes, plus a 2-byte CRC, plus a 'valid' token byte.
not used	03C3 03FF	
Upper Limits working copy	0400 04BF	Holds a 3-byte value (signed microVolts) for each analog input, i.e. 64x3 bytes = 192 bytes.
not used	04C0 04FF	
Upper Limits permanent storage + CRC	0500 05C2	Holds a 3-byte value (signed µVolts) for each analog input, i.e. 64x3 bytes = 192 bytes, plus a 2-byte CRC, plus a 'valid' token byte.
not used	05C3 05FF	
Lower Limits working copy	0600 06BF	Holds a 3-byte value (signed µVolts) for each analog input, i.e. 64x3 bytes = 192 bytes.
not used	06C0 06FF	
Lower Limits permanent storage + CRC	0700 07C2	Holds a 3-byte value (signed µVolts) for each analog input, i.e. 64x3 bytes = 192 bytes, plus a 2-byte CRC, plus a 'valid' token byte.
not used	07C3 07FF	

Table 6. EEPROM memory map for *ELMBio* application (addresses 1E0h - 7FFh).

EEPROM	ADDR
char 0 (ASCII)	0100
char 1 (ASCII)	0101
char 2 (ASCII)	0102
char 3 (ASCII)	0103
CRC MSB	0104
CRC LSB	0105
'V' (56h)	0106

Table 7. Detailed EEPROM memory map of the ELMB Serial Number storage space. Example: serial number "A123", is stored with 'A' in address 100h, '1' in 101h, '2' in 102h and '3' in 103h.

EEPROM	ADDR
Gain Factor *10^6 @ 25 mV range Phys Chan #1	0120 0122
Gain Factor *10^6 @ 25 mV range Phys Chan #2	0123 0125
Gain Factor *10^6 @ 25 mV range Phys Chan #3	0126 0128
Gain Factor *10^6 @ 25 mV range Phys Chan #4	0129 012B
5x3 = 15 bytes spare space (FFh)	012C 013A
CRC MSB	013B
CRC LSB	013C
'V' (56h)	013D
not *used*	013E 013F
Calib Constants 55 mV range	0140 015F
Calib Constants 100 mV range	0160 017F
Calib Constants 1V, 2.5V, 5V ranges	0180 01DF

Table 8. Detailed EEPROM memory map of the ELMB analog-in calibration constants storage space. Each gain factor is 3 bytes in size, stored LSB first. The ADC has 4 physical inputs (currently the gain factor is the same for all 4 inputs).

References

[1] CAN-in-Automation e.V.,
CANopen, Application Layer and Communication Profile,
CiA DS-301, Version 4.0, 16 June 1999,
http://www.can-cia.de

[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] **8-bit AVR Microcontroller with 128K Bytes In-System Programmable Flash, ATmega128, ATmega128L,**
ATMEL product datasheet.
http://www.atmel.com/products/AVR/

[4] **SAE81C90/91, Standalone Full-CAN Controller,**
SIEMENS product datasheet, preliminary, January 1997.

[5] CAN-in-Automation e.V.,
CANopen Device Profile for Generic I/O Modules,
CiA DS-401, Version 2.0, 20 December 1999,
http://www.can-cia.de

[6] **CRYSTAL CS5521/22/23/24/28, 16-Bit or 24-bit, 2/4/8-Channel ADCs with PGIA,** CIRRUS LOGIC product datasheet, DS317F2, May 2000,
http://www.cirrus.com