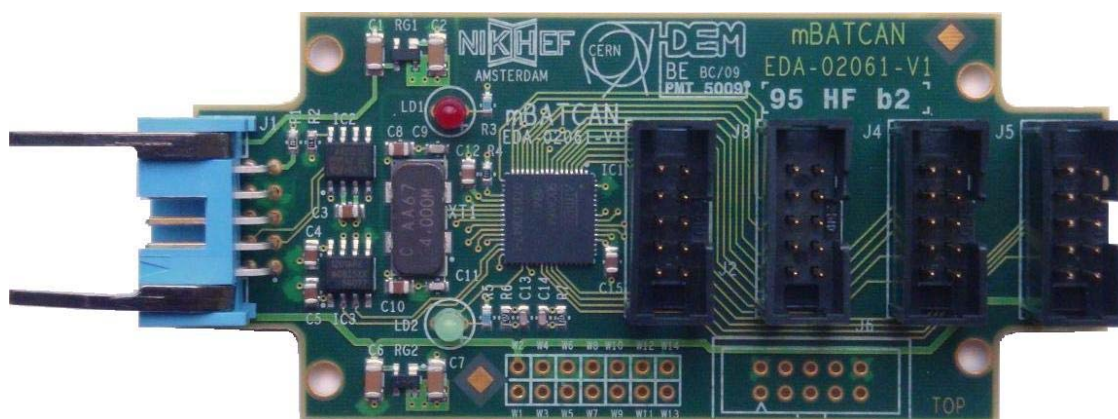


# mBATCAN

module for multiple B-sensor read-out  
by CAN bus and *CANopen*



## User Manual & Reference V1.0, 25 February 2010

Henk Boterenbrood



### ABSTRACT

*This document describes the mBATCAN, a module and its 'BATsCAN' application firmware for reading out up to 128 B-sensors modules of the type equipped with a Maxim DS2405 chip with a 64-bit identifier used for module identification as well as a digital output for B-sensor on-board ADC selection. Communication with the module is achieved by means of the CAN-bus and the CANopen protocol.*

## Table of Contents

1 OVERVIEW ..... 3

2 CONNECTORS AND INTERFACES ..... 5

3 INITIALISATION / NODE GUARDING AND LIFE GUARDING ..... 7

4 B-SENSOR STRING CONFIGURATION..... 7

5 B-SENSOR READ-OUT..... 9

6 CONFIGURATION STORAGE ..... 11

    6.1 EEPROM MEMORY MAP ..... 11

7 FIRMWARE UPGRADES..... 12

8 BATSCAN OBJECT DICTIONARY..... 12

9 EMERGENCY OBJECTS ..... 20

REFERENCES..... 23

Version History		
Version	Date	Comments
1.0	25 Feb 2010	Describes firmware version "Bs31.0004", originally derived from the BsCAN2 app for the MDT-DCS module, first written for the MDT-DCS then ported to mBATCAN.

**Table 1. Document change record (latest change first).**

## 1 Overview

The mBATCAN module running the *BATsCAN* application software allows to control and read-out 4 chains of up to 32 B-sensor modules each across a CAN bus using the *CANopen* protocol [1]. See Figure 1 for a picture of a small example system.

The B-sensor modules have to be of the type with the Maxim DS2405 addressable switch [2], used to uniquely identify each module by a 64-bit identifier (ID), as well as to select/deselect the B-sensor module on-board ADC by means of the chip's digital output.

In addition each B-sensor module should be plugged onto a BATSPI adapter board to allow the mBATCAN to issue broadcast conversion commands as well as for easy interconnection by cable with 10-pin boxheaders.

The mBATCAN design is an extension of the BATCAN module [3], which was designed to provide a CAN interface to a single B-sensor module, for the latest version of the B-sensor module (with DS2405 chip), as well as an older type (with DS2401 ID-chip). In fact, the mBATCAN could be programmed with the BATCAN application software unchanged with the single B-sensor module connected to chain connector #1 and it would behave just like a BATCAN module.

The *BATsCAN* application software for the mBATCAN allows for up to 32 B-sensors per chain, 128 modules in total. It addresses and selects individual B-sensor modules through their DS2405 chip using the *1-Wire* protocol. In addition, all B-sensors on a chain can be selected at the same time for broadcast upload of ADC conversion commands. Each B-sensor module is identified by an 8-bit index, which lies between 0 and 31 for the first chain, between 32 and 63 for the second chain, etc. This index can be used to retrieve the 64-bit unique ID of each B-sensor module which is stored in the DS2405 chip.

Each time the *BATsCAN* software starts up it scans for connected B-sensor modules and assigns each module an index based on the order in which the IDs from the DS2405 chips are found (according to the protocol described in the datasheet [2]). Alternatively the software can be configured to keep a certain configuration of B-sensor modules stored permanently in its memory, until explicitly instructed to re-examine the configuration. Keeping a certain configuration of B-sensor modules can be useful if one wants to be informed about missing modules in an otherwise fixed configuration immediately after a power-up.

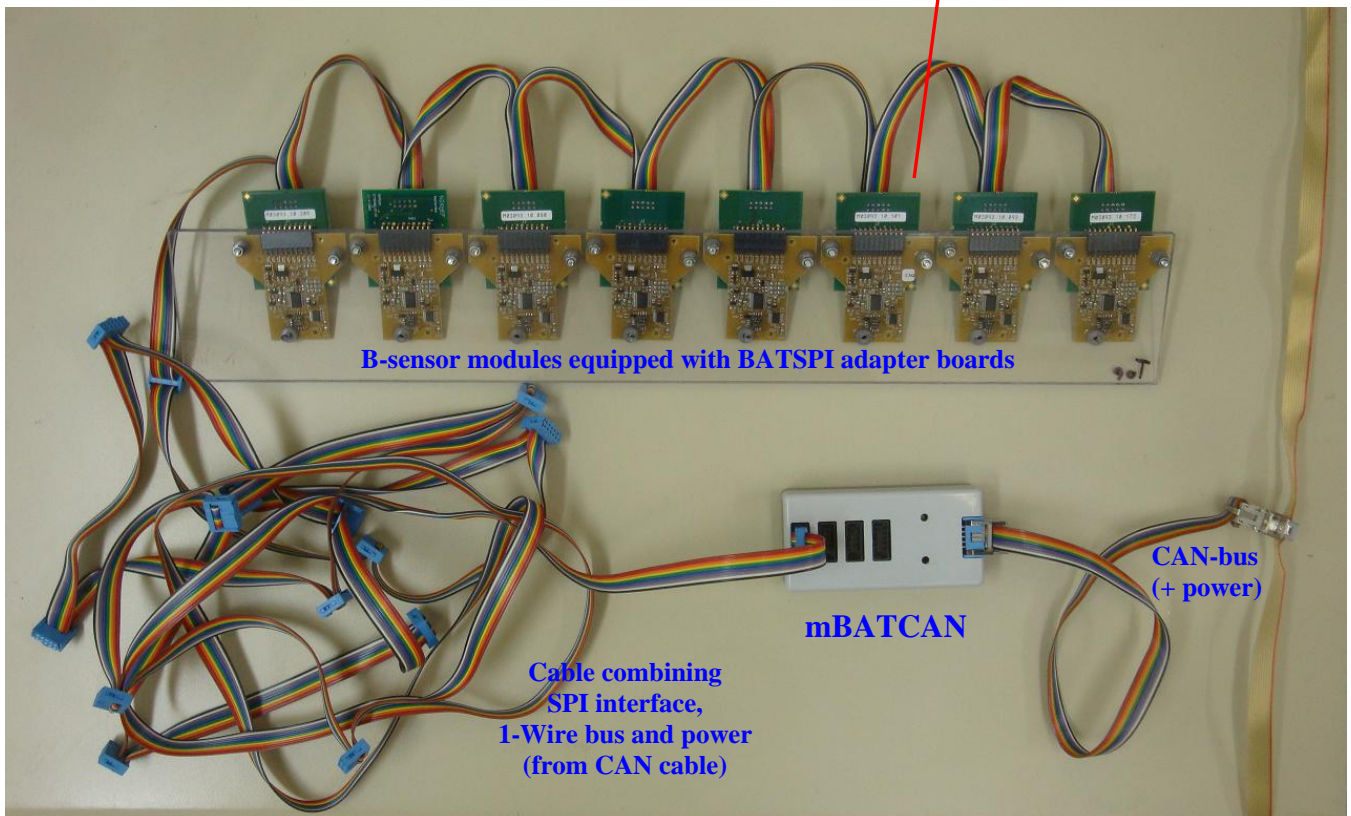
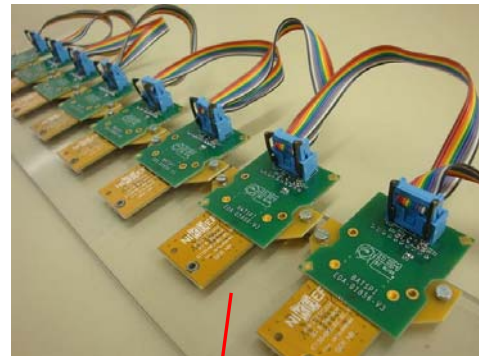
The mBATCAN module features two LEDs (one red, one green, see Figure 2) which are used to indicate the status of the module. The green LED is on when the module has properly initialized and the firmware is running<sup>1</sup>; it blinks briefly when CAN-messages are received or sent. The red LED comes on when there is a problem with (reading out) any of the B-sensor modules. The red LED remains on until the problem has been resolved.

The mBATCAN has a current consumption of about 22 mA.

---

<sup>1</sup> After power-up there is a delay of 4 s before the module starts, since a *Bootloader* application is in control of the module for the first 4 seconds before starting the *BATsCAN* application software.

Detail picture of the BATSPI adapter boards, cable and connectors



**Figure 1.** Example of an mBATCAN connected to one string of B-sensor modules. Up to 32 B-sensor modules can be connected to one string. The BATSPI adapter boards provide a solid 10-pin boxheader connector (shown in more detail in the top picture) and allows the mBATCAN to broadcast commands to the B-sensor ADCs ensuring simultaneous readings from all B-sensors. Power to the whole system (6-12V) is provided via the CAN cable. An mBATCAN connected to 128 B-sensor modules consumes about 500 mA.

## 2 Connectors and Interfaces

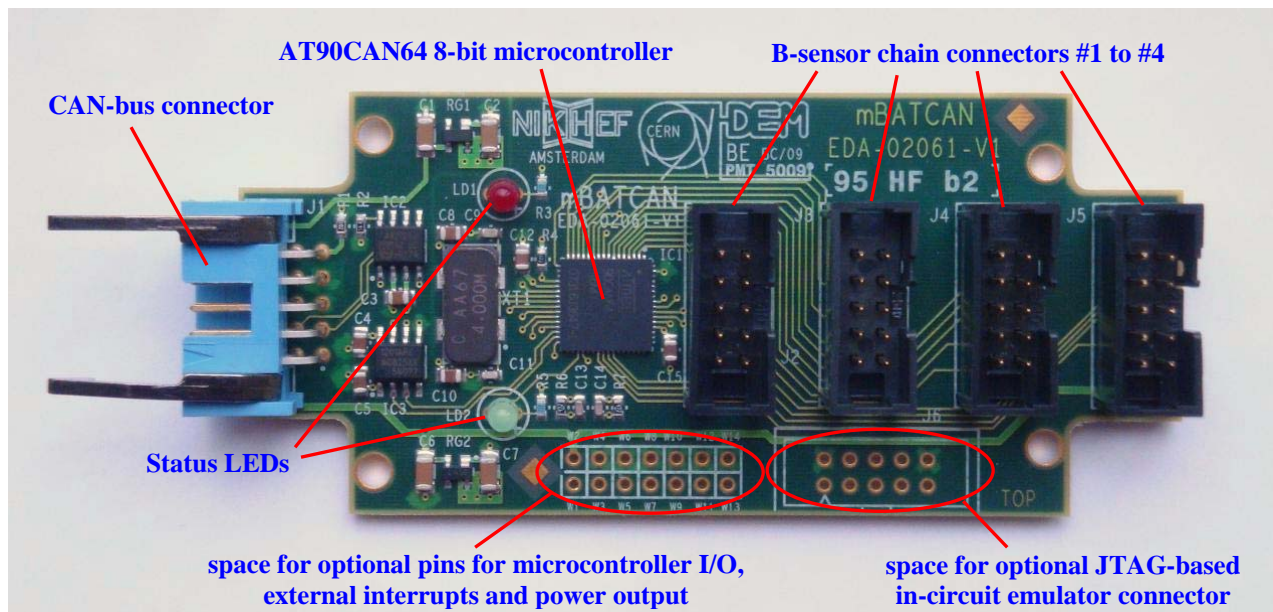
Figure 2 shows a picture of the mBATCAN circuit board (without its enclosure) and its connectors.

The mBATCAN module normally has 5 of the connectors installed, one CAN connector and four connectors for a B-sensor chain (of up to 32 B-sensor modules) each. The CAN connector layout is shown in Table 2. The B-sensor chain connector layout is shown in Table 3.

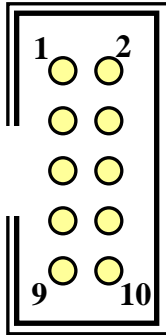
The first B-sensor chain connector also functions as a programmer connector enabling initial programming of the microcontroller using a serial programmer. The layout is shown in Table 4. Typically a so-called Bootloader is programmed in the microcontroller, enabling application software uploads via the CAN-bus, as described in section 7.

In addition there is space for an optional JTAG connector (normally not installed) enabling microcontroller programming, debugging and in-circuit emulation for mBATCAN software developers. The connector layout is shown in Table 5.

Also there is space for a number of pins that connect to microcontroller external interrupt inputs, which in case of the *BATsCAN* software (for B-sensor readout) can optionally be used to trigger read-out of the connected B-sensor modules (see the *Object Dictionary* in section 8). There are pins providing connections for power and ground as well, as shown in Table 6.

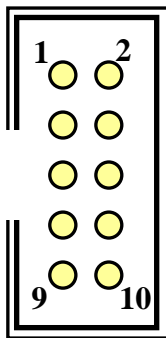


**Figure 2.** The mBATCAN circuit board, with the various connectors or connector locations indicated.



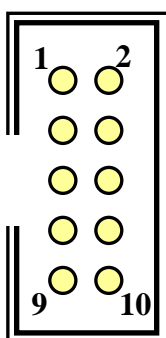
function	pin	pin	function
not connected	1	2	not connected (CAN-GND)
CAN-L	3	4	CAN-H
CAN-GND	5	6	+V (6-12V)
GND	7	8	CAN-POWER (6-12V)
CAN-SHIELD	9	10	not connected

Table 2. Layout of the mBATCAN CAN connector. CAN-POWER powers the CAN-driver part of the mBATCAN module. +V powers the rest of the mBATCAN module as well as the B-sensor modules.



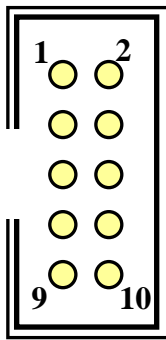
function	pin	pin	function
SCLK	1	2	GND
SDI	3	4	GND
SDO	5	6	GND
Chip-Select/Broadcast	7	8	ID I-Wire bus
-	9	10	+V (6-12V)

Table 3. Layout of the mBATCAN B-sensor chain connectors, combining SPI signals for controlling the B-sensor ADCs and I-Wire for controlling the B-sensor module DS2405 ID-chips.



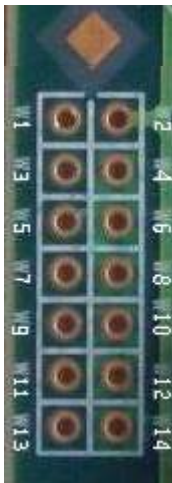
function	pin	pin	function
SCK (clock)	1	2	GND
PDI (data in)	3	4	GND
PDO (data out)	5	6	GND
-	7	8	-
RESET (to micro)	9	10	+V

Table 4. Layout of the mBATCAN programming connector. (= B-sensor chain connector #1). The programming connections are used only once, to program the Bootloader in the microcontroller's flash memory; from then on the CAN bus is used to upload application firmware. (Note that e.g. the AVRISP programmer needs power within the range 2.7-5.5V, so do not connect it directly to V+, which normally lies in the range 6-12V).



function	pin	pin	function
TCK	1	2	GND
TDO	3	4	+5V
TMS	5	6	RESET
–	7	8	–
TDI	9	10	GND

Table 5. Layout of the mBATCAN JTAG connector.



function	pin	pin	function
+5V	W1	W2	+V
PD0 (Intrpt 0)	W3	W4	GND
PD1 (Intrpt 1)	W5	W6	GND
PD2 (Intrpt 2)	W7	W8	GND
PD3 (Intrpt 3)	W9	W10	GND
PD4	W11	W12	GND
PD7	W13	W14	GND

Table 6. Layout of the mBATCAN connector with microcontroller I/O (PD0 to PD4 and PD7) and power.

### 3 Initialisation / Node Guarding and Life Guarding

The reader is referred to the corresponding sections in the BATCAN documentation [3].

Note that initialisation after power-up of an mBATCAN connected to –for example– 60 B-sensor modules takes close to 30 s to complete (and around 45 s in case ‘broadcast mode’ has been disabled).

### 4 B-sensor String Configuration

By default mBATCAN scans the connected strings for connected B-sensor modules at every power-up or hard-reset. It does this by ‘searching’ the *I-Wire* line that connects all the B-sensors’ DS2405 ID-chips on each of the strings. Any changes in the configuration of B-sensors is immediately reflected in the number of B-sensors it finds and on which string. This

means that if a B-sensor module (or at least its ID-chip) for whatever reason stops working it is no longer part of the configuration found, but the user would not be explicitly informed about the missing B-sensor. This may be undesirable and for this reason the user can optionally set a configuration parameter in the *BATsCAN* software to *not* scan for B-sensors at power-up/reset, but keep the configuration currently stored in the mBATCAN's non-volatile memory (EEPROM). This parameter is found in *Object Dictionary* index 5B05h (see section 8).

The user can instruct the *BATsCAN* software at any time to start a B-sensor scanning or probing operation, simply by reading *Object Dictionary* index 5B00h (see section 8). The configuration thus found overwrites the stored configuration.

B-sensors are assigned an 8-bit index in the order the DS2405 ID-chips were found. Index 0 to 31 are reserved for B-sensor string #1, index 32 to 63 for B-sensor string #2, index 64 to 95 for B-sensor string #3 and index 96 to 127 for B-sensor string #4. The index will be part of data read out from the B-sensor module, as shown in the next section. This same index must be used to retrieve the B-sensor module's 64-bit ID from *Object Dictionary* indices 5900h to 597Fh (see section 8).

Below is shown in some detail the CAN message exchange between a host and the mBATCAN module to trigger a 'probe' of the B-sensor strings. The CAN messages are formatted as defined by the *CANopen* standard [1].

The **SDO**-client request message (from the host system) that initiates such a 'scan' or 'probe' operation looks like this:

**Host → BATsCAN**

COB-ID	Byte						
	0	1	2	3	4	5	6-7
600h+ <i>NodeID</i>	40h	00h	5Bh	00h	–	–	–

Assuming everything went well and the number of B-sensor modules found is 30 (2Dh), for example, *BATsCAN* replies with the following **SDO**-server message:

**BATsCAN → Host**

COB-ID	Byte						
	0	1	2	3	4	5	6-7
580h+ <i>NodeID</i>	4Fh	00h	5Bh	00h	2Dh	–	–

If for any reason an **SDO**-client request from the host fails *BATsCAN* replies with a so-called **SDO Abort Transfer** message; this message has the following syntax:

**BATsCAN → Host**

COB-ID	Byte					
	0	1	2	3	4-7	
580h+ <i>NodeID</i>	80h	<i>index (LSB)</i>	<i>index (MSB)</i>	<i>subindex</i>	<i>Abort Code</i>	

## 5 B-sensor Read-out

To trigger the read-out of all B-sensor modules (their 3 Hall-sensors and 1 T-sensor) in the configuration a so-called **SYNC** message must be send.

The **SYNC** message is a *CANopen* CAN-message with a fixed COB-ID and no data bytes:

**Host** → **all (SYNC-)slave nodes**

COB-ID
080h

After receiving this message the *BATsCAN* node starts up a sequence of AD-conversions on all B-sensor modules simultaneously (really simultaneous per string, by a broadcast of a conversion command; per string the conversion commands are sent ca. 0.2 ms apart, which is short compared to the ADC conversion time so that one could say that ‘simultaneous’ still applies) to convert the *H1*, *H2* and *H3* Hall-sensor inputs and the *T* sensor input, and subsequently reads out the converted analog inputs and sends the values one-by-one in a message on the CAN-bus.

Strings are scanned for data from *BATsCAN* string connector #1 to connector #4, and B-sensor modules on one string are scanned in the order their ID-chips were found, i.e. in the order of their 8-bit index. Per B-sensor module the messages with conversion data arrive in the order *H1*, *H2*, *H3* and *T* respectively.

In practice this means that after a conversion sequence is started (a SYNC message is received by the *BATsCAN* node) it takes about  $4 \cdot 67 = 268$  ms (measured 272 ms) for the 4-channel conversion sequence to complete and the first message arrives (i.e. for an AD-conversion wordrate of 15 Hz, which is the default). The messages with Hall sensor data from one B-sensor module are about 1 ms apart, the message with temperature sensor data follows after 4 ms (due to the time it takes to make the calculations to convert to degrees Celcius). The time between the last message of one B-sensor module and the first message of the next B-sensor module is about 37 ms (due to the *I-Wire* protocol used to select individual B-sensor ADCs).

For example: read-out of 60 B-sensor modules divided over 4 strings by one *BATsCAN* box has been measured and takes  $272 + (60-1) \cdot 37 + 60 \cdot (3 \cdot 1 + 4)$  ms = 2775 ms, so just under 3 s from start to finish for 60 B-sensor modules.

If for any reason, for example if the B-sensor modules in the string are not equipped with a BATSPI adapter board (so that the ADC’s serial data output SDO can not be disabled), it is possible to configure *BATsCAN* to *not* broadcast conversion commands but issue them to the B-sensor modules one-by-one. This can be set in *Object Dictionary* index 5000h, sub-index 24. (Read-out of 60 modules takes just under 4 s in this case).

The message containing an ADC value is called a **TPDO** (*Transmit Process Data Object*) in *CANopen* jargon, which is a message without any further *CANopen* protocol overhead. The data bytes in the message contain application data only. Some example messages are shown below.

BATsCAN will produce –per channel– the following 6-databyte **TPDO** message containing the data for one ADC channel conversion:

**BATsCAN → Host**

TPDO COB-ID	Byte 0	Byte 1	Byte 2	Byte 3-5
480h+NodeID	Index	Chan no	ADC-config	ADC value (or T)

with:

- Index:* B-sensor module index (between 0 and 127).
- Chan no:* 0 = Hall H1, 1 = Hall H2, 2 = Hall H3, 3 = T-sensor.
- ADC value:* 24-bits value, LSB in byte 3, MSB in byte 5  
(Temperature in millidegrees centigrade, see example below).
- 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 4000h-40EFh, sub 2,3,4,5,6 and 7.

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

(NB: **BATsCAN** supports other modes of readout for this **PDO** (e.g. timer-triggered), but this is not further described in detail here).

Example messages with B-sensor ADC data:

**BATsCAN → Host**

TPDO COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
490h	13h	02h	00h	B0h	D6h	FFh

This is a message from *BATsCAN* node 16 (*NodeID*=490h-480h=10h, see COB-ID):

- B-sensor index 19 (=13h, Byte 0)
- Hall-sensor H2 (channel number 2, Byte 1)
- Gain range = 100 mV bipolar, conversion word rate = 15.0 Hz (Byte 2)
- 24-bit ADC-value -10576 (=FFD6B0h, Byte 3+4+5)
- *Note:* a Hall-sensor conversion value is a 24-bit signed number ! (note the negative value in this example)

**BATsCAN → Host**

TPDO COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
490h	13h	03h	0Bh	A0h	5Bh	00h

This is a message from *BATsCAN* node 16 (*NodeID*=490h-480h=10h, see COB-ID):

- B-sensor index 19 (=13h, Byte 0)
- T-sensor (channel number 3, Byte 1)
- Gain range = 2.5 V unipolar, conversion word rate = 15.0 Hz (Byte 2)
- 24-bit ADC-value 23456 (=005BA0h, Byte 3+4+5), i.e. temperature = 23.456 °C.

*Note:* a T-sensor conversion value is a 24-bit unsigned number !

## 6 Configuration Storage

### 6.1 EEPROM Memory Map

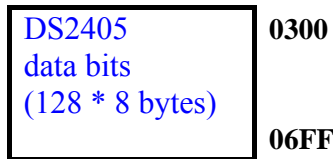


Table 7 shows in detail the layout of the microcontroller’s EEPROM usage by the BATsCAN application software.

EEPROM	ADDR	DESCRIPTION
<i>not used</i>	0000	
BATsCAN configuration parameters	0001 →  00A0	→ 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	00A1 →  00F0	→ 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).
<i>not used</i>	00F1  00FF	
Module Serial Number	0100 →  0106	→ Holds the module’s Serial Number given to it at production time; serves to uniquely identify the module.
Node-ID (opt)	0107 →	→ 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 <i>must</i> be used.
<i>not used</i>	01FE	
Storage space for B-sensor index-to- string map	01FF → 0201 → 027F	→ Length byte: 128 → Space to store a map of 128 string numbers, in order of B-sensor index, for each index a string number between 0 and 3. A B-sensor index not present in the current configuration has a string number of FFh (This is for historical reasons: it would be sufficient to store the number of connected modules per string).
Map valid byte Map CRC (LSB) Map CRC (MSB)	0280 0282 →	→ Bytes with integrity data concerning the B-sensor index-to-string map
<i>not used</i>	0283  02FF	
DS2405 data bits (128 * 8 bytes)	0300 →  06FF	→ Copy of the 64-bit IDs from all B-sensor modules in the current configuration; needed in case a configuration is to be saved between power-ups.

**Table 7. EEPROM memory map of the BATsCAN application firmware.**

## 7 Firmware Upgrades

The reader is referred to the corresponding section in the BATCAN documentation [3].

## 8 BATsCAN 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 ( <i>BATsCAN</i> )						
Index (hex)	Sub Index	Description	Data/Object	Attr	Default	Comment
1000	-	Device type	U32	RO	00000000h	Meaning: no particular device profile supported
1001	-	Error register	U8	RO	0	
1002	-	Manufacturer status reg	U32	RO	0	
1008	-	Manufacturer device name	VisStr	RO	"BATC"	= BATCAN
1009	-	Manufacturer hw version	VisStr	RO	"BC10"	= BATCAN v1.0
100A	-	Manufacturer software version	VisStr	RO	"Bs31"	<i>BATsCAN</i> application v3.1 ( <i>minor-version</i> number in 100A,1)
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 ADC config, dig.I/O config, ...
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 ADC config, etc.
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
1018		Identity	Record			Mandatory CANopen object
	0	Number of entries	1..4	RO	1	
	1	Vendor ID	U32	RO	12345678h	<i>to be ordered from CiA</i>

Communication Profile Area (BATsCAN) (continued...)						
Index (hex)	Sub Index	Description	Data/Object	Attr	Default	Comment
1803		4 <sup>th</sup> Transmit PDO par's	Record			Data type = PDOCommPar
	0	Number of entries	U8	RO	5	
	1	COB-ID used by PDO	U32	RO	480h + <i>NodeID</i>	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	<i>not used</i>
	4	<i>Not used</i>	U8	RO	0	
*	5	Event timer [1 s]	U16	RW	0	In units of secs, must be <= 255; active for all transmission-types!
1A03		4 <sup>th</sup> Transmit PDO mapping	Record			Data type = PDOMapping
	0	Number of entries	U8	RO	3	<i>should be 255 for MuxPDO, but this is not a CANopen MPDO...</i>
	1	B-sensor index	U32	RO	51000108h	
	2	B-sensor ADC channel number	U32	RO	50000008h	<i>actually not allowed, but...</i>
	3	24-bit analogue input	U32	RO	500x0x20h	OD-index 50xx, subindex x, Analogue inputs, multiplexed, size = 32 bits

Manufacturer-specific Profile Area (BATsCAN)						
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 results in SDO <i>Abort</i> with <i>Error Code 1</i> ; error while accessing FLASH results in SDO <i>Abort</i> with <i>Error Code 6</i> .
	2		U16	RO	0	<i>not used</i>
	3	Get CRC	U16	RO		Return CRC from flash
3100	-	Serial Number	U32	RW		Number or 4-byte string uniquely identifying an mBAT-CAN, given during production.
3101	-	Enable Serial Number write operation	U8	WO	<b>DON'T USE</b>	Writing 5Ah enables one write operation on the Serial Number (Object 3100).
3200		CAN-controller settings and status	Record			
	0	Number of entries	U8	RO	4	
	1	Format error interrupt counters	U32	RO		Byte 0: SERG Byte 1: CERG Byte 2: FERG Byte 3: AERG
*	2	Enable auto-start	U8	RW	0	If =1 go to <i>Operational</i> at startup
*	3	Bus-off max retry counter	U8	RW	2	Counter is decremented every 1s, but if the node reaches this maximum value it abandons re-gaining CAN-bus access
	4	Received message counter	U8	RO		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. ( <i>Bootloader</i> firmware version 1.3 and later supports this feature, otherwise don't use it !)
3301	-	Enable CAN Node Identifier write operation	U32	WO		Writing a number that matches the Serial Number (Object 3100) enables one write operation on the CAN Node Identifier (Object 3300).

<b>Manufacturer-specific Profile Area (BATsCAN) (continued...)</b>						
<b>Index (hex)</b>	<b>Sub Index</b>	<b>Description</b>	<b>Data/Object</b>	<b>Attr</b>	<b>Default</b>	<b>Comment</b>
5000		B-sensor #0 ADC-config	Record			CS5524 24-bit ADC <sup>1</sup>
	0	Number of entries	U8	RO	37	
	1	Number of input channels	U8	RO	7	
*	2	Conversion Word Rate Hall	U8	RW	0	3-bit code <sup>2</sup>
*	3	Input Voltage Range Hall	U8	RW	0	3-bit code <sup>3</sup>
*	4	Unipolar/Bipolar Measurement Mode Hall	U8	RW	0	0 = bipolar, 1 = unipolar
*	5	Conversion Word Rate Temp	U8	RW	0	3-bit code <sup>2</sup>
*	6	Input Voltage Range Temp	U8	RW	5	3-bit code <sup>3</sup>
*	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
	9/29 <sup>4</sup>	Configuration Register	U32	RW		CS5524 Config Register
	10/30 <sup>4</sup>	Offset Register #1	U32	RW		CS5524 physical channel AIN1
	11/31 <sup>4</sup>	Gain Register #1	U32	RW		CS5524 physical channel AIN1
	12/32 <sup>4</sup>	Offset Register #2	U32	RW		CS5524 physical channel AIN2
	13/33 <sup>4</sup>	Gain Register #2	U32	RW		CS5524 physical channel AIN2
	14/34 <sup>4</sup>	Offset Register #3	U32	RW		CS5524 physical channel AIN3
	15/35 <sup>4</sup>	Gain Register #3	U32	RW		CS5524 physical channel AIN3
	16/36 <sup>4</sup>	Offset Register #4	U32	RW		CS5524 physical channel AIN4
	17/37 <sup>4</sup>	Gain Register #4	U32	RW		CS5524 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 (delay for ADC)	U8	RW	10	in $\mu$ s, 10 $\leq$ value $\leq$ 255
	23	ADC recovery from timeout, faulty calib constants, etc. during readout scans, using a reset-and-calibration sequence	Bool	RW	1	TRUE after each power-up or reset (was FALSE, changed to TRUE, in firmware version 1.2.1)
*	24	ADC broadcast enabled	U8	RW	1	Broadcast of commands to B-sensor modules allowed or not
	25-28	---	---	---		<i>reserved</i>

<sup>1</sup> Subindex 2-9 and 22-24 are common to all B-sensor modules ! (If you change them for one B-sensor, you change them for all B-sensors).

<sup>2</sup> **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

<sup>3</sup> **000**: 100 mV,    **001**: 55 mV,    **010**: 25 mV,    **011**: 1 V,    **100**: 5 V,    **101**: 2.5 V

<sup>4</sup> Without ADC selection via DS2405 chip; do a separate selection operation using Object 5B04h; this saves time when reading e.g. all Offset and Gain registers.

Manufacturer-specific Profile Area (BATsCAN) (continued...)						
Index (hex)	Sub Index	Name	Data/Object	Attr	Default	Comment
5001		B-sensor #1 ADC-config	Record			CS5524 24-bit ADC
	<i>etc.</i>	<i>...as above...</i>	<i>...</i>	<i>...</i>	<i>...</i>	
<i>etc</i>	<i>...</i>	<i>...</i>	<i>...</i>	<i>...</i>	<i>...</i>	
507F		B-sensor #127 ADC-config	Record			CS5524 24-bit ADC
	<i>etc.</i>	<i>...as above...</i>	<i>...</i>	<i>...</i>	<i>...</i>	
5100		B-sensor module status	Record			Error status, one bit per B-sensor 0=OKAY, 1=Error or Absent
	0	Number of entries	U8	RO	4	
	1	status B-sensors #0-#31	U32	RO	FFFFFFFFh	Status of string #0
	2	status B-sensors #31-#63	U32	RO	FFFFFFFFh	Status of string #1
	3	status B-sensors #63-#95	U32	RO	FFFFFFFFh	Status of string #2
	4	status B-sensors #96-#127	U32	RO	FFFFFFFFh	Status of string #3
5200		ADC status <u>B-sensor #0</u>	U8	RO	FFh	<sup>1</sup> (see footnote)
5201		ADC status <u>B-sensor #1</u>	U8	RO	FFh	<sup>1</sup>
5202		ADC status <u>B-sensor #2</u>	U8	RO	FFh	<sup>1</sup>
...	...	...	...	...		...
...	...	...	...	...		...
527F		ADC status <u>B-sensor #127</u>	U8	RO	FFh	<sup>1</sup>
5300		ADC-reset-and-calibrate <u>B-sensor #0</u>	U8	WO		Writing any value triggers a reset and calibration sequence on B-sensor #0 using the current ADC settings
5301		ADC-reset-and-calibrate <u>B-sensor #1</u>	U8	WO		reset+calib of B-sensor #1 ADC
...	...	...	...	...		...
...	...	...	...	...		...
537F		ADC-reset-and-calibrate <u>B-sensor #127</u>	U8	WO		reset+calib of B-sensor #127 ADC
5380		ADC-reset-and-calibrate <b>all</b> B-sensors present in current configuration	U8	WO		

<sup>1</sup> B-sensor module ADC status byte/bitmask: **00h**: no error, **01h**: ADC reset error, **02h**: ADC calibration error, **04h**: ADC conversion time-out, **FFh**: ADC absent / not used / not in configuration.

<b>Manufacturer-specific Profile Area (BATsCAN) (continued...)</b>						
<b>Index (hex)</b>	<b>Sub Index</b>	<b>Name</b>	<b>Data/Object</b>	<b>Attr</b>	<b>Default</b>	<b>Comment</b>
5400		ADC readout trigger input	Record			Microcontroller PD0 / INT0 pin
	0	Number of entries	U8	RO	2	
	1	Enabled/disabled	Bool	RW	0	If enabled and the node is Operational a read-out of the B-sensors is started (with data in TPDO4 messages) after an interrupt
	2	Rising or falling edge	Bool	RW	1	1 = on rising edge
5401		ADC readout trigger input	Record			Microcontroller PD1 / INT1 pin
	0	Number of entries	U8	RO	2	
	1	Enabled/disabled	Bool	RW	0	See comment at Object 5400
	2	Rising or falling edge	Bool	RW	1	
5402		ADC readout trigger input	Record			Microcontroller PD2 / INT2 pin
	0	Number of entries	U8	RO	2	
	1	Enabled/disabled	Bool	RW	0	See comment at Object 5400
	2	Rising or falling edge	Bool	RW	1	
5403		ADC readout trigger input	Record			Microcontroller PD3 / INT3 pin
	0	Number of entries	U8	RO	2	
	1	Enabled/disabled	Bool	RW	0	See comment at Object 5400
	2	Rising or falling edge	Bool	RW	1	

<b>Manufacturer-specific Profile Area (BATsCAN) (continued...)</b>						
<b>Index (hex)</b>	<b>Sub Index</b>	<b>Name</b>	<b>Data/Object</b>	<b>Attr</b>	<b>Default</b>	<b>Comment</b>
5500		Read analogue input <u>B-sensor #0</u>	Record			24 bits analogue values (B-sensor #0)
	0	Number of entries	U8	RO	7	
	1	Input 1: Hall-sensor H1	I24	RO		1 <sup>st</sup> analog input:24-bit
	2	Input 2: Hall-sensor H2	I24	RO		2 <sup>nd</sup> " " "
	3	Input 3: Hall-sensor H3	I24	RO		3 <sup>rd</sup> " " "
	4	Input 4: current sense	I24	RO		4 <sup>th</sup> " " "
	5	Input 5: NTC T-sensor	I24	RO		5 <sup>th</sup> " " "
	6	Input 6: 0°C calib input	I24	RO		6 <sup>th</sup> " " "
	7	Input 7: 100°C calib input	I24	RO		7 <sup>th</sup> " " "
5501		Read analogue input <u>B-sensor #1</u>	Record			24 bits analogue values (B-sensor #1)
	0	Number of entries	U8	RO	7	
	1	Input 1	I24	RO		1 <sup>st</sup> analog input:24-bit
	...	...	...	...		...
	7	Input 7	I24	RO		7 <sup>th</sup> " " "
...	...	...	...	...		...
557F		Read analogue input <u>B-sensor #127</u>	Record			24 bits analogue values (B-sensor #127)
	0	Number of entries	U8	RO	7	
	1	Input 1	I24	RO		1 <sup>st</sup> analog input:24-bit
	...	...	...	...		...
	7	Input 7	I24	RO		7 <sup>th</sup> " " "
5600		B-sensor index list				
	0	Total number of B-sensors	U8	RO	<i>n</i>	= number of modules detected when <i>probing</i> (see Object 5B00)
	1	Index of 1 <sup>st</sup> B-sensor	U8	RO		
	...	...	...	...		...
	<i>n</i>	Index of <i>n</i> <sup>th</sup> B-sensor	U8	RO		
5700		Number of B-sensors per string	Array			
	0	Total number of strings	U8	RO	4	
	1	Number of B-sensors on string #1	U8	RO		
	2	Number of B-sensors on string #2	U8	RO		
	3	Number of B-sensors on string #3	U8	RO		
	4	Number of B-sensors on string #4	U8	RO		

<b>Manufacturer-specific Profile Area (BATsCAN) (continued...)</b>						
<b>Index (hex)</b>	<b>Sub Index</b>	<b>Name</b>	<b>Data/Object</b>	<b>Attr</b>	<b>Default</b>	<b>Comment</b>
5800		B-sensor-to-string mapping	Array			This array is kept in EEPROM
*	0	string# with B-sensor #0	U8	RO		value FFh means: this B-sensor not found in any of the strings
*	1	string# with B-sensor #1	U8	RO		"
*	2	string# with B-sensor #2	U8	RO		"
*	...	...	...	...		...
*	...	...	...	...		...
*	127	string# with B-sensor #127	U8	RO		"
5900		B-sensor #0 64-bit ID	Record			
	0	Number of entries	U8	RO	2	
	1	ID data, lower 32 bits	U32	RO	FFFFFFFFh	
	2	ID data, upper 32 bits	U32	RO	FFFFFFFFh	
5901		B-sensor #1 64-bit ID	Record			
	<i>etc.</i>	<i>...as above...</i>	...	...	...	
<i>etc</i>	...	...	...	...		...
597F		B-sensor #127 64-bit ID	Record			
	<i>etc.</i>	<i>...as above...</i>	...	...	...	
5B00	-	Probe strings for B-sensor modules and generate new configuration and store it	U8	RO	0	Triggers a scan for B-sensor modules on all 4 strings; stores the newly found mapping of B-sensor modules on strings (Object 5300) as well as the 64-bit IDs (Objects 4400 to 447F) in EEPROM; resets and calibrates all B-sensor modules found; returns the total number of B-sensor modules found (Object 5100, sub 0); takes roughly about 6 s to complete.
5B03	-	Find and deselect any DS2405-activated B-sensor modules	U8	RO	0	Returns number of deselected modules
5B04	0	Select or deselect B-sensor #0 by toggling its DS2405 output	U8	RO	0	Returns 0 when selected, 0xFF when deselected
	1	Select or deselect B-sensor #1 by toggling its DS2405 output	U8	RO	0	"
	...	...	...	...		...
	127	Select or deselect B-sensor #127 by toggling its DS2405 output	U8	RO	0	"
5B05	*	Probe after every hard reset or power-up	U8	RO	1	If =0 the B-sensor configuration from Object 5B00h is kept unchanged between power-ups

<b>Manufacturer-specific Profile Area (BATsCAN) (continued...)</b>						
<b>Index (hex)</b>	<b>Sub Index</b>	<b>Description</b>	<b>Data/Object</b>	<b>Attr</b>	<b>Default</b>	<b>Comment</b>
5C00	-	Compile Options	U32	RO		Bitmask denoting which compile options were used when the application code was generated
5E00	-	Transfer control to Boot-loader	U8	WO		

## 9 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**

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

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 **BsCAN2** application firmware:

<b>Error Description</b>	<b>Emergency Error Code (byte 0-1; hex)</b>	<b>Manufacturer-specific Error Field (byte 3-7)</b>
CAN communication	8100	Byte 3: 00h Byte 4: total format error count 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	8130	CAN-controller has been reinitialized
RPDO: too few bytes	8210	Byte 3: minimum DLC (Data Length Code)
CRC error	5000	Byte 3: 30h Byte 4: 1 (program FLASH)

...table continues on the next page...

<b>Error Description</b>	<b>Emergency Error Code</b> (byte 0-1; hex)	<b>Manufacturer-specific Error Field</b> (byte 3-7)
EEPROM: write error	5000	Byte 3: 41h Byte 4: Parameter block index <sup>1</sup> 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 <sup>1</sup> Byte 5: Error id (1=CRC, 2=length, 4=infoblock)
B-sensor ADC: conversion timeout	5000	Byte 3: 51h Byte 4: B-sensor index (0..127) Byte 5: ADC channel number (0..7)
B-sensor ADC: reset failed	5000	Byte 3: 52h Byte 4: B-sensor address (0..127) Byte 5: Error id <sup>2</sup>
B-sensor ADC: Hall-sensor calibration failed	5000	Byte 3: 53h Byte 4: B-sensor index (0..127)
B-sensor ADC: T-sensor calibration failed	5000	Byte 3: 54h Byte 4: B-sensor index (0..127)
B-sensor ADC problem(s) during initialization (check OD 1002)	5000	Byte 3: 55h Byte 4: FFh means an invalid or no configuration was present in EEPROM, any other number indicates the number of B-sensor modules that failed to initialize properly
B-sensor ADC: invalid Gain constant detected (current valid setting: 0x41≤high_byte≤0x4B )	5000	Byte 3: 56h Byte 4: B-sensor index (0..127) Byte 5: Gain Reg high byte (of Hall sensor inputs) Byte 6: Gain Reg middle byte (of Hall sensor inputs)
B-sensor ADC: found mismatch with stored index in ADC register	5000	Byte 3: 57h Byte 4: B-sensor index (0..127) Byte 5: index read from ADC register
B-sensor ADC: unexpected ADC (de) selection required on DS2405	5000	Byte 3: 58h Byte 4: B-sensor index (0..127) Byte 5: error counter (if =0 then Byte4=number of deselected DS2405s)
Irregular reset (Watchdog, Brown-out or JTAG)	5000	Byte 3: F0h Byte 4: microcontroller MCUSR register contents <sup>3</sup>
Bootloader: not present	5000	Byte 3: F1h

<sup>1</sup> **0**: PDO communication parameters, **1**: Guarding parameters, **4**: B-sensor ADC configuration, **6**: CAN configuration parameters, **8**: B-sensor string configuration, **FEh**: Calibration constant(s), **FFh**: mBATCAN Serial Number.

<sup>2</sup> **01h**: Reset-Valid bit not set, **02h**: Reset-Valid bit not reset, **04h**: error in Offset Register value, **08h**: error in Gain Register value

<sup>3</sup> AT90CANxx MCUSR register bits: **01h**: Power-On Reset, **02h**: External Reset, **04h**: Brown-Out Reset, **08h**: Watchdog Reset, **10h**: JTAG Reset.

Error Description	Emergency Error Code (byte 0-1; hex)	Manufacturer-specific Error Field (byte 3-7)
Bootloader is now in control <sup>1</sup>	5000	Byte 3: FEh Byte 4: 80h Byte 5: 64h Byte 6: microcontroller MCUSR register contents <sup>2</sup> Byte 7: 00h
Bootloader cannot jump to application: invalid <sup>1</sup>	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 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 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	<i>reserved (=0)</i>
7	manufacturer specific

<sup>1</sup> This Emergency message is generated by the Bootloader program !

<sup>2</sup> AT90CANxx *MCUSR* register bits: **01h**: Power-On Reset, **02h**: External Reset, **04h**: Brown-Out Reset, **08h**: Watchdog Reset, **10h**: JTAG Reset.

## References

- [1] 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>
  
- [2] **DS2405 addressable switch,**  
Datasheet, Dallas Semiconductor Maxim.  
<http://www.maxim-ic.com>
  
- [3] H.Boterenbrood,  
**BATCAN, module for single B-sensor read-out by CAN bus and CANopen,**  
Version 1.2, NIKHEF, Amsterdam, 20 January 2010.  
<http://www.nikhef.nl/pub/departments/ct/po/html/Bsensor/BATCAN.pdf>