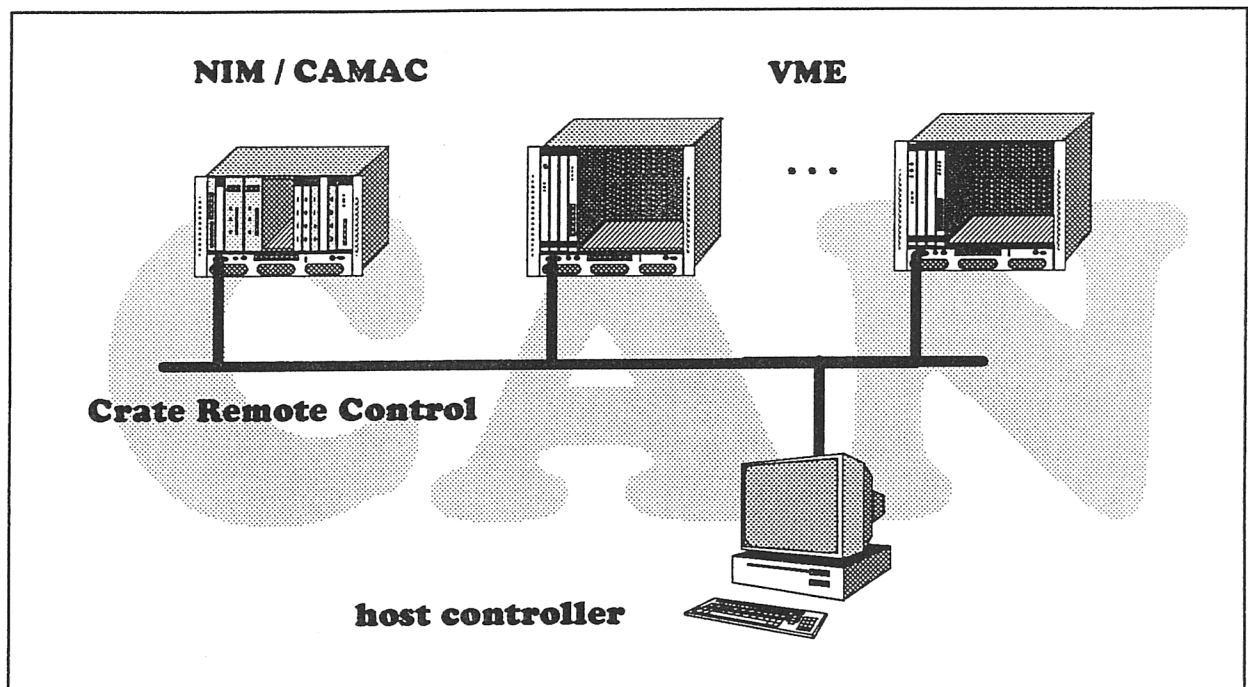


CAN-Bus Interface for W-Ie-Ne-R Crate Remote Control



A. Ruben, A. Köster, W-Ie-Ne-R Plein Baus GmbH,
Müllersbaum 20, D-51399 Burscheid

CAN-BUS Interface for W-IE-Ne-R Crate Remote Control

A. Ruben, A. Köster, W-IE-NE-R Plein & Baus GmbH,

Müllersbaum 20, 51399 Burscheid

25 January 1996

1. W-IE-Ne-R CAN based crate remote control

Due to the enlargement of the electronic set-up in modern experiments the crate remote control as a part of the slow-control system becomes more and more important.

Based on the development of the W-IE-Ne-R μ -processor controlled intelligent fan-tray-units which can be equipped with the W-IE-Ne-R CAN bus interface this remote control can be integrated within an CAN fieldbus system.

In addition to the remote on / off and SYSRES (VME) the user can control and program remotely every crate parameter via the interface as:

- *All voltages,*
- *All current limits,*
- *Over- and Undervoltage trip off points,*
- *Overcurrent trip off points,*
- *Temperature measurements:*
 - *Power supply*
 - *Fan tray air inlet*
 - *air outlet temperature on top of slot 1*
- *Status signals,*
- *Average speed of the fans and display of every single fan speed,*
- *Identification of the crate,*
- *Configuration and adjustment.*

Some of the values given above can only be changed by authorised persons. All commands for configuration, calibration or maintenance functions for system service are disabled (jumper) for the user. ¹

If any error (e.g. fan failure) is detected, a high priority message is transmitted from the can interface itself (without request from the server).

The crate identification number may be in the range from 1 to 126. It is possible to enable a "general call" access (e.g. to switch on some crates with only one can bus command), and to disable the can interface (for troubleshooting).

¹The transfer protocol is identical for all different crates (VME, CAMAC, ...), but the really available commands are dependend of the real target.

2. Introduction to the CAN-Bus

2.1. General Features of CAN

The Controller Area Network (CAN) defined by Bosch in 1985 is an advanced serial multimaster communication protocol. Due to the reliability and technical capability as well as to the available low-price system components CAN is well suited for application in fieldbus system. The most important features of CAN² are:

- unlimited number of nodes (depending on physical layer)
- serial, asynchronous, object-oriented, multi-master communication
- 2032 priorities (message IDs) in standard frame
- max. 8 data bytes per message
- CSMA(CA (collision avoidance) bus access priority controlled (ID) with non-destructive bit-wise arbitration
- wide range of transmission rates (programmable), high speed up to 1.6Mbit/s (577kbit/s information)
- twisted pair cabling, line or star topology
- real-time capability, guaranteed latency time for high priority messages <134µs (1Mbit/s)
- high level of reliability and safety due to integrated error detection (HD=6), handling and confinement, less than 10^{-13} undetected errors per message

2.2. Specification and standardisation of CAN

The CAN specification and standardisation is based on the following ISO reference model for Open Systems Interconnections³,

OSI-Layer 7	Application	specified by system designer, several proposals as CMS(CAL)
OSI-Layer 6	Presentation	empty
OSI-Layer 5	Session	empty
OSI-Layer 4	Transport	empty
OSI-Layer 3	Network	empty
OSI-Layer 2	Data Link	covered by CAN-protocol specs and ISO standard, implemented on CAN-controller ICs
OSI-Layer 1	Physical	covered by ISO standard and partially by CAN protocol

The CAN protocol is defined in the CAN specifications version 2.0 part A (standard frame) and B (extended frame). Two draft have been worked out by the ISO,

- ISO/DIS11898 CAN high-speed; 125kbit/s to 1Mbit/s, max 30 nodes
- ISO/DIS11519 part 1 CAN low-speed; up to 125kbit/s, max. 20 nodes

²CAN Specification Version 2.0, Philips Semiconductors Hamburg, 1991

³ISO 7498 Information Processing System - OSI Basic Reference Model, 1984

2.3. CAN high speed physical layer and transmission medium

Both high-speed and low-speed CAN are using a two-wire differential bus line with common return.

The maximum distance between two nodes is determined by the transmission rate

Max. Distance	Bit Rate	Type
10 m	1.6 Mbit/s	high- speed
40 m	1.0 Mbit/s	
130 m	500 kbit/s	
270 m	250 kbit/s	
530 m	125 kbit/s	
620 m	100 kbit/s	low-speed
1300 m	50 kbit/s	
3300 m	20 kbit/s	
6700 m	10 kbit/s	
10.000 m	5 kbit/s	

In case of high-speed CAN the bus line has to be terminated with 120 Ohm (characteristic impedance)

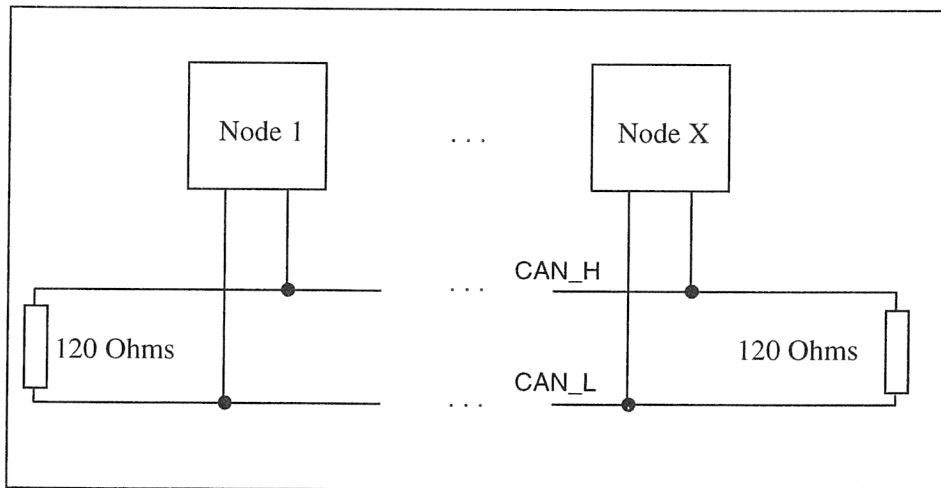


Fig. 1, Can bus line termination

The CAN high-speed bus node levels are:

	recessive	dominant
$V_{CAN_H} - V_{CAN_L}$	-500mV to +500mV, no load	+1.5V to +3.0V (60 Ohm load)

2.4. CAN data frame

To guarantee high flexibility and a (theoretically) unlimited number of node the CAN bus data transfer is organised according to the object-oriented transmission principle. CAN nodes don't have a fixed address, i.e. every message can be detected by every bus node at the same time (broadcast). The decision to process the message or not is done by each node itself with the assistance of an acceptance filter.

The CAN data frame is shown in the following figure⁴,

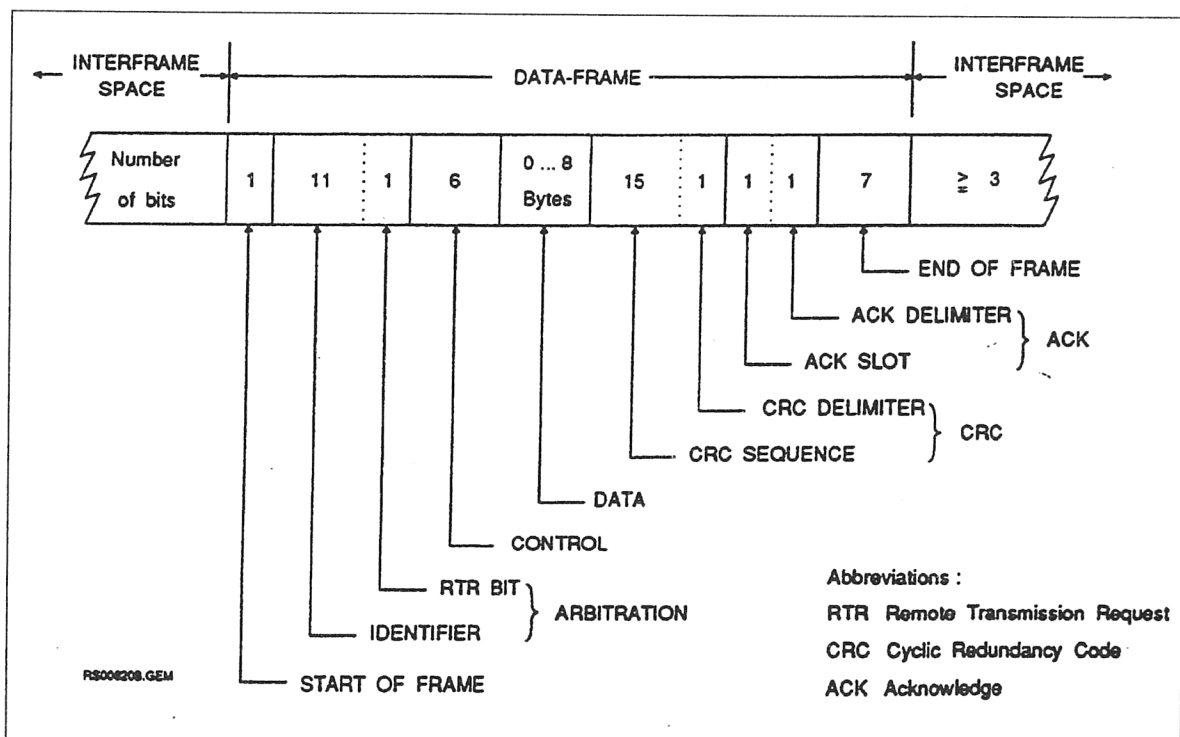


Fig. 2, CAN data frame

⁴Philips Semiconductor, Application Note HKI/AN 92 001,

3. W-Ie-Ne-R CAN Interface Card

The W-Ie-Ne-R CAN-bus Interface for crate remote control is a plug-on card which can be optionally mounted within our processor-controlled fan trays. It is equipped with the PHILIPS micro controller P80C592 including a 80C51 CPU kernel and a CAN Controller (CAN Spec. 2.0A).

The connection to the physical bus is done with an opto-isolated transceiver with controlled rise and fall slope to reduce RFI and allow the use of unshielded cable (PCA82C250). The transceiver allows to connect at least 110 nodes to a high speed (1 Mbaud) bus.

For CAN-bus interfacing the fan trays are equipped on the front panel with a 9-pin male DSUB connector according to the CiA-spec. DS102-1⁵.

4. CAN based Crate Remote Protocol Definition

The following address range definition as well as the protocol (which is based on a subset of the CAN Application Layer (CAL) protocol) is based on basic principles which are conform to the standards used at BESSY II / Control System Division⁶.

4.1. Identifier / Address bit map definition

The **Object-ID** (11 bits) space is divided into an **Node-ID** (7 bits) and a **SubObject-ID** (4 bits). The access of up to 127 crates is favoured due to the driver capability of the used CAN Controller P80C592 (up to 110 nodes). In addition, a second „general call“ address (default 127) may be used to simultaneously switch on or call for data to all connected crates. To avoid **Object-ID** = 0 the **Node-ID** = 0 is forbidden.

The **SubObject-ID** bits refer to 13 different ID values for each crate with the following functions (see for detail next page table):

- Status / Control
- Voltage / current values for power channels 0 - 4
-
- Voltage / current values for power channels 3 - 7
- Fan speed, Temperatures
- Temperatures
- Ident, configuration
- Voltage configuration

The reduction of the **SubObject-ID** space of 4 bits to 13 different ID-function values considers the reserved COB identifier of the CAL protocol as well as the ID's reserved by the P8xC592.

To guarantee nearly same rights to all nodes / crates the **Node-ID** has to be fixed within the 7 lowest significant bits. Due to the 8-bit acceptance code register (ACR) of the applied PHILIPS micro controller P8xC592 the ID-check has to be done by the software within the controller chip, i.e. the acceptance code register can not be used. However, this allows the definition and use of the broadcast-ID for a call of all nodes.

⁵CAN Physical Layer for Industrial Applications, Draft Standard CiA/DS 102-1, CiA 1992

⁶Protokoll für den CAN-Bus-Anschluß der HF-Anlagen-SPS, R. Lange, BESSY II, Control System Division, 1995

Thus the SubObjectID and NodeID definition has the following schema

ID-bit	10	9	8	7	6	5	4	3	2	1	0
ACR- range	ac.7	ac.6	ac.5	ac.4	ac.3	ac.2	ac.1	ac.0	-	-	-
NodeID					n6	n5	n4	n3	n2	n1	n0
SubObjectID	s3	s2	s1	s0							

This ID-bit mapping does not directly correspond to the CAL / CMS priority definition (priority level 1 ... 7) however, the reserved COB identifier (1761 ... 2031) are considered by reducing the SubObjectID to 13 values:

ID-Function	Name	s3	s2	s1	s0	ID-range
Read Status from Crate	IDstat	0	0	0	0	1 ... 127
Write Control Command to Crate	IDctrl	0	0	0	1	129 ... 255
Read voltage/current channel 0+4	IDvc04	0	0	1	0	157 ... 383
Read voltage/current channel 1+5	IDvc15	0	0	1	1	385 ... 511
Read voltage/current channel 2+6	IDvc26	0	1	0	0	513 ... 639
Read voltage/current channel 3+7	IDvc37	0	1	0	1	641 ... 767
Read fan speed	IDfan	0	1	1	0	769 ... 895
Read temperatures	IDtemp	0	1	1	1	897 ... 1023
reserved		1	0	0	0	1025 ... 1151
Crate sends voltage configuration data	IDucfgC	1	0	0	1	1153 ... 1279
Host requests/programs voltage configuration data	IDucfgH	1	0	1	0	1281 ... 1407
Crate sends configuration data	IDcfgC	1	0	1	1	1409 ... 1535
Host requests/programs configuration data	IDcfgH	1	1	0	0	1537 ... 1663

The ID-range 1664 ... 1760 corresponding to the higher part of the CAL CMS priority level 7 is free and can be used for other nodes.

4.2. Crate message exchange principles

4.2.1. Write data to crate

To set values for control (ON/OFF, SYSRES,...), fan speed, voltage channel parameters, ... the host can send a data frame to a single node or to all crates using the broadcast NodeID. Each crate can be enabled/disabled for broadcast calls.

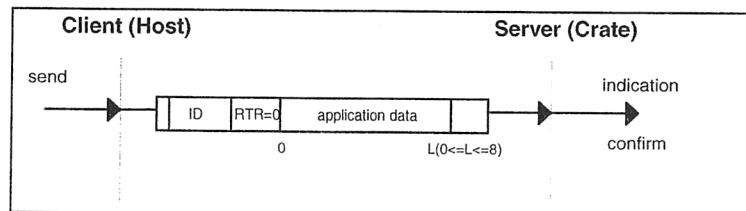


Fig.3, Write data to crate principle

4.2.2. Read crate parameters

The main crate parameters are sent by the crate (within standard data frame) after request from the host which is transmitted via remote frame (RTRbit=1).

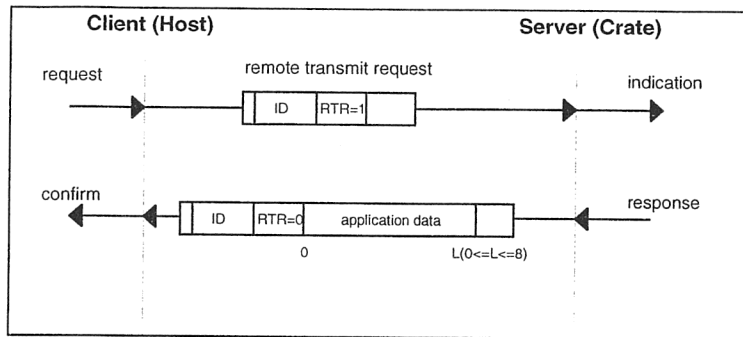


Fig. 5, Standard remote request principle

4.2.3. Indexed Read-Write Access (multiplexed variables)

For the read-out and control of not often required crate parameters (set-up, identifier, calibration, software versions, trip off points, ...) indexed read and write procedures are used which conform to the read-write access of CAL-multiplexed variables. The first data byte i corresponds to the index (≤ 127) or multiplexor whereby, the highest significant bit is used as a specifier for the command type or returned result. The ID's for host- and crate-write for one multiplexed variable are not identical.

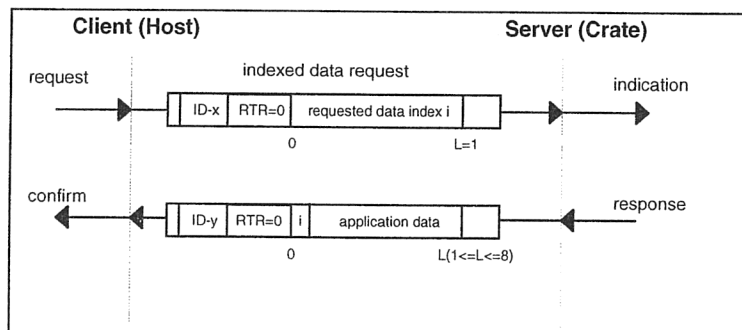


Fig. 6, Indexed data request principle

Read requests may be transmitted by the host at any time without waiting for response. (If there are multiple read requests for the same data, the crate will answer only once.) To avoid data overrun the host is not allowed to send again a write request before receiving the answer (e.g. programming status) from the selected crate.

4.2.4. Failure messages

In case of a crate failure the status is given automatically by this crate to the host. Due to the highest priority of the status-ID within the SubObjectID range this message reaches the host within the shortest time which is possible independently from the crate/node-ID.

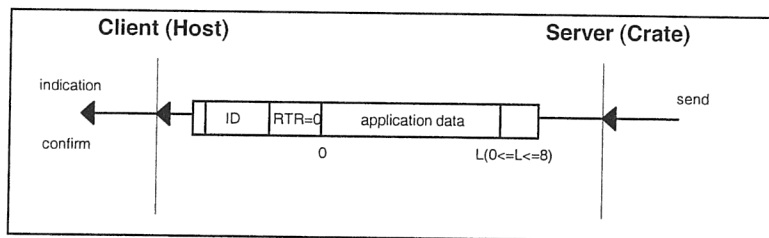


Fig.7, Failure message transfer principle

4.3. Crate commands and SubObjects

- IDstat - Get Crate Status

	RTR	Byte Count	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Host request	1	1-8	-	-	-	-	-	-	-	-
Crate confir	0	same as host request	status byte 0	status byte 1	under-voltage error flags	over-voltage error flags	external temperatur error flags ⁷	over-curernt error flags	ovp error flags	power supply temperatur error flags

Status Byte 0		
Bit	0	1
0	power is off	power is on
1	external power inhibit	no external power inhibit
2	power fail	ac is in limit
3	any error, see other flags for specification	no error condition
4	fans are broken	fans are ok
5	trip off if fans are broken is disabled	trip off if fans are broken is enabled
6	trip off if any error is disabled	trip off if any error is enabled
7	vme bus signal sysfail active (low)	vme bus signal sysfail inactive (high)
Status Byte 1		
Bit	0	1
0	reserved (0)	
1	reserved (0)	
2	reserved (0)	
3	reserved (0)	
4	reserved (0)	
5	flash/eprom data has not changed since last access via can bus	flash/eprom data has changed (e.g. with the manual control of a fan tray)
6	no flash/eprom data checksum error	flash/eprom checksum error, default values are used
7	no write protect (service only)	hardware write protect

⁷ New in CANBUS 1.04, it was reserved for "undercurrent error flags" before, but never implemented

The error flags in byte 3-8 are 0 if ok and 1 if an error condition is valid. There is one bit for each voltage. If the hardware is not able to detect which voltage makes trouble, all bits are set.

If bit 3 of the status byte changes indicating an error, the crate sends the complete status frame (8 bytes) to the host without request.

- **IDctrl - Send Control Command**

	RTR	Byte Count	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Host writes data to crate	0	1, 2	control byte	fan speed	-	-	-	-	-	-

Control Byte		
Bit	0	1
0	disable crate switch (see bit 1)	enable crate switch (see bit 1)
1	switch crate off (only used if bit 0 = 1)	switch crate on (only used if bit 0 = 1)
2	nothing to do	generate vme-sysreset
3		
4		
5		
6	error trip off enable	error trip off disable
7	don't change the fan speed	change the fan speed (new value in byte 2)

- **IDvc04, IDvc15, IDvc26 and IDvc37 - Get measured Voltage and Current of channels 0 ... 7**

	RTR	Byte Count	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Host request	1	1-8	-	-	-	-	-	-	-	-
Crate confirm	0	same as host request	u0 (u1, u2, u3), low byte	u0 (u1, u2, u3), high byte	i0 (i1, i2, i3), low byte	i0 (i1, i2, i3), high byte	u4 (u5, u6, u7), low byte	u4 (u5, u6, u7), high byte	i4 (i5, i6, i7), low byte	i4 (i5, i6, i7), high byte

u, i: 16 bit signed binary data (Voltage or current).
 You must get the exponent with the „Ucfg“-Command. Only two different exponents are used for each channel: one for the voltages and one for the currents. During initialisation, you must request some voltage data (e.g. „Output Voltage Settings“ and some current data (e.g. Current Limit Settings“) with the Ucfg command to get this two exponents.
 If the exponents are both -2, the voltage range is between -327.68V and +327.67V and the current range is between -327.68A and +327.67A)

- **IDfan - Get Fan speed**

	RTR	Byte Count	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Host request	1	1-8	-	-	-	-	-	-	-	-
Crate confirm	0	same as host request	middle fan speed	nominal fan speed	fan 1 speed	fan 2 speed	fan 3 speed	fan 4 speed	fan 5 speed	fan 6 speed

The fan speed is defined as „turns per second“.
 Not existing fans are 255.

- ID_Temp - Get Temperatures

	RTR	Byte Count	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Host request	1	1-8	-	-	-	-	-	-	-	-
Crate confirm	0	same as host request	ext/bin temp 1 (if not existing: fan air inlet temp.)	ext./bin temp 2 (or power supply air temp.)	ext/bin temp 3	ext/bin temp 4	ext/bin temp 5	ext/bin temp 6	ext/bin temp 7	ext/bin temp 8

The temperature range is -128 ... +127 °C.

Not supported temperatures are -128.

- IDucfgC, IDucfgH - Get / Set Voltage Configuration Data

	ID	Byte Count	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Output Voltage Settings										
Host request	UcfcH	1	128+ui+0	-	-	-	-	-	-	-
Crate confirm (ok)	UcfcC	8	ui+0	value (low)	value (high)	min. value (low)	min. value (high)	max. value (low)	max. value (high)	exp.
Crate confirm (fail)	UcfcC	2	ui+0	Status	-	-	-	-	-	-
Host writes data to crate	UcfcH	3, 5, 7, 8	ui+0	value (low)	value (high)	min. * value (low)	min. * value (high)	max. * value (low)	max. * value (high)	exp. *
Crate confirm	UcfcC	2	ui+0	Status	-	-	-	-	-	-
Current Limit Settings										
Host request	UcfcH	1	128+ui+1	-	-	-	-	-	-	-
Crate confirm (ok)	UcfcC	8	ui+1	value (low)	value (high)	min. value (low)	min. value (high)	max. value (low)	max. value (high)	exp.
Crate confirm (fail)	UcfcC	2	ui+1	Status	-	-	-	-	-	-
Host writes data to crate	UcfcH	3, 5, 7, 8	ui+1	value (low)	value (high)	min. * value (low)	min. * value (high)	max. * value (low)	max. * value (high)	exp. *
Crate confirm	UcfcC	2	ui+1	Status	-	-	-	-	-	-
Undervoltage Compare Settings										
Host request	UcfcH	1	128+ui+2	-	-	-	-	-	-	-
Crate confirm (ok)	UcfcC	8	ui+2	value (low)	value (high)	min. value (low)	min. value (high)	max. value (low)	max. value (high)	exp.
Crate confirm (fail)	UcfcC	2	ui+2	Status	-	-	-	-	-	-
Host writes data to crate	UcfcH	3, 5, 7, 8	ui+2	value (low)	value (high)	min. * value (low)	min. * value (high)	max. * value (low)	max. * value (high)	exp. *
Crate confirm	UcfcC	2	ui+2	Status	-	-	-	-	-	-
Overvoltage Compare Settings										
Host request	UcfcH	1	128+ui+3	-	-	-	-	-	-	-
Crate confirm (ok)	UcfcC	8	ui+3	value (low)	value (high)	min. value (low)	min. value (high)	max. value (low)	max. value (high)	exp.
Crate confirm (fail)	UcfcC	2	ui+3	Status	-	-	-	-	-	-
Host writes data to crate	UcfcH	3, 5, 7, 8	ui+3	value (low)	value (high)	min. * value (low)	min. * value (high)	max. * value (low)	max. * value (high)	exp. *
Crate confirm	UcfcC	2	ui+3	Status	-	-	-	-	-	-
minimum Current Compare Settings										
Host request	UcfcH	1	128+ui+4	-	-	-	-	-	-	-
Crate confirm (ok)	UcfcC	8	ui+4	value (low)	value (high)	min. value (low)	min. value (high)	max. value (low)	max. value (high)	exp.
Crate confirm (fail)	UcfcC	2	ui+4	Status	-	-	-	-	-	-

	ID	Byte Count	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Host writes data to crate	UcfgH	3, 5, 7, 8	ui+4	value (low)	value (high)	min. * value (low)	min. * value (high)	max. * value (low)	max. * value (high)	exp. *
Crate confirm	UcfgC	2	ui+4	Status	-	-	-	-	-	-
Overcurrent Compare Settings										
Host request	UcfgH	1	128+ui+5	-	-	-	-	-	-	-
Crate confirm (ok)	UcfgC	8	ui+5	value (low)	value (high)	min. value (low)	min. value (high)	max. value (low)	max. value (high)	exp.
Crate confirm (fail)	UcfgC	2	ui+4	Status	-	-	-	-	-	-
Host writes data to crate	UcfgH	3, 5, 7, 8	ui+5	value (low)	value (high)	min. * value (low)	min. * value (high)	max. * value (low)	max. * value (high)	exp. *
Crate confirm	UcfgC	2	ui+5	Status	-	-	-	-	-	-
Overvoltage Protection										
Host request	UcfgH	1	128+ui+6	-	-	-	-	-	-	-
Crate confirm (ok)	UcfgC	8	ui+6	value (low)	value (high)	min. value (low)	min. value (high)	max. value (low)	max. value (high)	exp.
Crate confirm (fail)	UcfgC	2	ui+6	Status	-	-	-	-	-	-
Host writes data to crate	UcfgH	3, 5, 7, 8	ui+6	value (low)	value (high)	min. * value (low)	min. * value (high)	max. * value (low)	max. * value (high)	exp. *
Crate confirm	UcfgC	2	ui+6	Status	-	-	-	-	-	-
Temperature Warning (CANBUS 1.01) ⁷										
Host request	UcfgH	1	128+ui+7	-	-	-	-	-	-	-
Crate confirm (ok)	UcfgC	8	ui+7	value (low)	value (high)	min. value (low)	min. value (high)	max. value (low)	max. value (high)	exp.
Crate confirm (fail)	UcfgC	2	ui+7	Status	-	-	-	-	-	-
Host writes data to crate	UcfgH	3, 5, 7, 8	ui+7	value (low)	value (high)	min. * value (low)	min. * value (high)	max. * value (low)	max. * value (high)	exp. *
Crate confirm	UcfgC	2	ui+7	Status	-	-	-	-	-	-
Temperature Limit (CANBUS 1.01) ⁷										
Host request	UcfgH	1	128+ui+8	-	-	-	-	-	-	-
Crate confirm (ok)	UcfgC	8	ui+8	value (low)	value (high)	min. value (low)	min. value (high)	max. value (low)	max. value (high)	exp.
Crate confirm (fail)	UcfgC	2	ui+8	Status	-	-	-	-	-	-
Host writes data to crate	UcfgH	3, 5, 7, 8	ui+8	value (low)	value (high)	min. * value (low)	min. * value (high)	max. * value (low)	max. * value (high)	exp. *
Crate confirm	UcfgC	2	ui+8	Status	-	-	-	-	-	-
Output Voltage fine adjustment (CANBUS 1.03)										
Host request	UcfgH	1	128+ui+9	-	-	-	-	-	-	-
Crate confirm (ok)	UcfgC	8	ui+9	value (low)	value (high)	min. value (low)	min. value (high)	max. value (low)	max. value (high)	exp.
Crate confirm (fail)	UcfgC	2	ui+9	Status	-	-	-	-	-	-
Host writes data to crate	UcfgH	3, 5, 7, 8	ui+9	value (low)	value (high)	min. * value (low)	min. * value (high)	max. * value (low)	max. * value (high)	exp. *
Crate confirm	UcfgC	2	ui+9	Status	-	-	-	-	-	-

ui: channel number * 16 (0: U/I0, 16: U/I1, 32: U/I2, ..., 112: U/I7)

Value: 16 bit signed binary data (Voltage or current)

Exp.: 8 bit signed exponent of the values. (If the exponent is -2, the value range is between -327.68 and +327.67.) Only two different exponents are allowed for each channel: one for all voltages and one for all currents.

⁷ Temperature warning and limit allways for BIN/external temperature sensors.

- Status: 0: ok
- 1: trying to program „write protected“ data without permission
- 2: not allowed value (min. value > max. value, min. value > value, max. value < value)
- 3: undefined command
- 4: command is not supported by the existing hardware
- 252: not allowed byte count
- 253: data overrun (a new host write request is received before the crate confirm of the previous write request is transmitted). The new data will be ignored by the crate. If there are multiple data overruns, only the first error will be answered; all errors occurring between the first error and the time the error message is transmitted will be ignored quiet)
- 254: hardware error (eeprom checksum not ok)
- 255: hardware error (unable to access the eeprom data)

*

read-only values

CANBUS X.XX Function is only supportet in CANBUS Software X.XX or higher

- IDcfgC, IDcfgH - Get / Set Configuration and Version Data

	ID	Byte Count	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
CAN Crate Control Software Version										
Host request	cfgH	1	128+0	-	-	-	-	-	-	-
Crate confirm	cfgC	8	0	'C'	'A'	'N'	x	'.'	x	x
Fan Software Version										
Host request	cfgH	1	128+1	-	-	-	-	-	-	-
Crate confirm (ok)	cfgC	8	1	ID0	ID1	ID2	ID3	ID4	ID5	ID6
Crate confirm (fail)	cfgC	2	1	Status	-	-	-	-	-	-
Host request	cfgH	1	128+2	-	-	-	-	-	-	-
Crate confirm (ok)	cfwR	8	2	ID7	ID8	ID9	ID10	ID11	ID12	ID13
Crate confirm (fail)	cfgC	2	2	Status	-	-	-	-	-	-
Power Supply Software Version										
Host request	cfgH	1	128+3	-	-	-	-	-	-	-
Crate confirm (ok)	cfgC	8	3	ID0	ID1	ID2	ID3	ID4	ID5	ID6
Crate confirm (fail)	cfgC	2	3	Status	-	-	-	-	-	-
Host request	cfgH	1	128+4	-	-	-	-	-	-	-
Crate confirm (ok)	cfgC	8	4	ID7	ID8	ID9	ID10	ID11	ID12	ID13
Crate confirm (fail)	cfgC	2	4	Status	-	-	-	-	-	-
Fan Operating Time (CANBUS 1.02)										
Host request	cfgH	1	128+5	-	-	-	-	-	-	-
Crate confirm	cfgC	4	5	Minutes (low)	Minutes (middle)	Minutes (high)	-	-	-	-
Power Supply Operating Time (CANBUS 1.02)										
Host request	cfgH	1	128+6	-	-	-	-	-	-	-
Crate confirm	cfgC	4	6	Minutes (low)	Minutes (middle)	Minutes (high)	-	-	-	-
Fan ID String										
Host request	cfgH	1	128+8	-	-	-	-	-	-	-
Crate confirm (ok)	cfgC	8	8	ID0	ID1	ID2	ID3	ID4	ID5	ID6
Crate confirm (fail)	cfgC	2	8	Status	-	-	-	-	-	-
Host request	cfgH	1	128+9	-	-	-	-	-	-	-
Crate confirm (ok)	cfgC	8	9	ID7	ID8	ID9	ID10	ID11	ID12	ID13
Crate confirm (fail)	cfgC	2	9	Status	-	-	-	-	-	-
Host request	cfgH	1	128+10	-	-	-	-	-	-	-
Crate confirm (ok)	cfgC	8	10	ID14	ID15	ID16	ID17	ID18	ID19	ID20
Crate confirm (fail)	cfgC	2	10	Status	-	-	-	-	-	-
Host request	cfgH	1	128+11	-	-	-	-	-	-	-
Crate confirm (ok)	cfgC	8	11	ID21	ID22	ID23	ID24	ID25	ID26	ID27
Crate confirm (fail)	cfgC	2	11	Status	-	-	-	-	-	-
Host writes data to crate	cfgH	8	8	ID0	ID1	ID2	ID3	ID4	ID5	ID6
Crate confirm	cfgC	2	8	Status	-	-	-	-	-	-
Host writes data to crate	cfgH	8	9	ID7	ID8	ID9	ID10	ID11	ID12	ID13
Crate confirm	cfgC	2	9	Status	-	-	-	-	-	-
Host writes data to crate	cfgH	8	10	ID14	ID15	ID16	ID17	ID18	ID19	ID20
Crate confirm	cfgC	2	10	Status	-	-	-	-	-	-
Host writes data to crate	cfgH	8	11	ID21	ID22	ID23	ID24	ID25	ID26	ID27
Crate confirm	cfgC	2	11	Status	-	-	-	-	-	-
Power Supply ID String										
Host request	cfgH	1	128+12	-	-	-	-	-	-	-
Crate confirm (ok)	cfgC	8	12	ID0	ID1	ID2	ID3	ID4	ID5	ID6
Crate confirm (fail)	cfgC	2	12	Status	-	-	-	-	-	-
Host request	cfgH	1	128+13	-	-	-	-	-	-	-
Crate confirm (ok)	cfgC	8	13	ID7	ID8	ID9	ID10	ID11	ID12	ID13
Crate confirm (fail)	cfgC	2	13	Status	-	-	-	-	-	-
Host request	cfgH	1	128+14	-	-	-	-	-	-	-
Crate confirm (ok)	cfgC	8	14	ID14	ID15	ID16	ID17	ID18	ID19	ID20
Crate confirm (fail)	cfgC	2	14	Status	-	-	-	-	-	-
Host request	cfgH	1	128+15	-	-	-	-	-	-	-
Crate confirm (ok)	cfgC	8	15	ID21	ID22	ID23	ID24	ID25	ID26	ID27
Crate confirm (fail)	cfgC	2	15	Status	-	-	-	-	-	-

	ID	Byte Count	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Host writes data to crate	cfgH	8	12	ID0	ID1	ID2	ID3	ID4	ID5	ID6
Crate confirm	cfgC	2	12	Status	-	-	-	-	-	-
Host writes data to crate	cfgH	8	13	ID7	ID8	ID9	ID10	ID11	ID12	ID13
Crate confirm	cfgC	2	13	Status	-	-	-	-	-	-
Host writes data to crate	cfgH	8	14	ID14	ID15	ID16	ID17	ID18	ID19	ID20
Crate confirm	cfgC	2	14	Status	-	-	-	-	-	-
Host writes data to crate	cfgH	8	15	ID21	ID22	ID23	ID24	ID25	ID26	ID27
Crate confirm	cfgC	2	15	Status	-	-	-	-	-	-
Test and Configuration Data (The user may not use this functions)										
Host writes data to crate	cfgH	8	127	0	BC(0-17)	D0	D1	D2	D3	D4
Crate confirm	cfgC	2	127	0	-	-	-	-	-	-
Host writes data to crate	cfgH	nothing or 8	127	1	D5	D6	D7	D8	D9	D10
Crate confirm	cfgC	2	127	0	-	-	-	-	-	-
Host writes data to crate	cfgH	nothing or 8	127	2	D11	D12	D13	D14	D15	D16
Crate confirm	cfgC	2-8	127	0	BC	D0	D1	D2	D3	D4
Crate confirm	cfgC	nothing or 2-8	127	1	D5	D6	D7	D8	D9	D10
Crate confirm	cfgC	nothing or 2-5	127	2	D11	D12	D13	D14	D14	D16

ui: channel number * 16 (0: U/I0, 16: U/I1, 32: U/I2, ..., 112: U/I7)

Value: 16 bit signed binary data (Voltage or current)

Exp.: 8 bit signed exponent of the values. (If the exponent is -2, the value range is between -327.68 and +327.67.) Only two different exponents are allowed for each channel: one for all voltages and one for all currents.

Status: 0: ok

1: trying to program „write protected“ data without permission

2: not allowed value (min. value > max. value, min. value > value, max. value < value)

3: undefined command

4: command is not supported by the existing hardware

252: not allowed byte count

253: data overrun (a new host write request is received before the crate confirm of the previous write request is transmitted). The new data will be ignored by the crate. If there are multiple data overruns, only the first error will be answered; all errors occurring between the first error and the time the error message is transmitted will be ignored quiet)

254: hardware error (eeprom checksum not ok)

255: hardware error (unable to access the eeprom data)

*

read-only values

5. Technical data W-IE-NE-R CAN bus interface

CAN controller type: P80C592 (CAN 2.0A protocol)
Physical Layer: differential according to ISO 11898
Transceiver: PCA82C250, opto-isolated, rise and fall slope control
CAN connector: 9-pin DSUB male according to CiA DS 102-1

Pin	Line	Comment
1	-	reserved by CiA
2 (10*)	CAN_L	CAN_L bus line (dominant low)
3 (9*)	GND	Ground
4	-	reserved by CiA
5	-	reserved by CiA
6	-	
7 (11*)	CAN_H	CAN_H bus line (dominant high)
8	-	reserved by CiA (failure signal)
9	-	

* optional connection to 15 pin DSUB female
connector (UEV4020 VME Bins only)

Baudrates:

Max. Distance	Bit Rate	Type
10 m	1.6 Mbit/s	high- speed
40 m	1.0 Mbit/s	
130 m	500 kbit/s	
270 m	250 kit/s	
530 m	125 kbit/s	
620 m	100 kbit/s	low-speed
1300 m	50 kbit/s	
3300 m	20 kbit/s	
6700 m	10 kbit/s	
10.000 m	5kbit/s	

Revision History

10.12.98	Documentation: Better explanation of the ui-exponent. (Kö)
21.12.98	Extension CANBUS 1.01 (Monitoring of user supplied temperature sensors.) (Kö)
31.05.99	Extension CANBUS 1.02 (Operating Time monitoring)
	Documentation: Status Byte 0: VME Sysfail definition was wrong in the manual
28.06.99	Extension of IDstat: Flash/EEPROM bits (Kö)
27.03.00	Monitoring of fan temperature and external temperature sensors.
17.04.00	Extension CANBUS 1.03 (Output Voltage fine adjustment)
23.04.02	Extension CANBUS 1.04 (Get Crate Status: min. current flags never used, now redefined to External Overtemperature Error Flags)