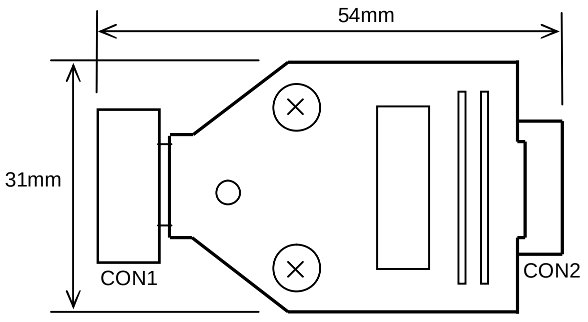
CAN Bus Controller *Preliminary* EMCBK33A



- **CAN Bus controller for TFT and VFD Modules**
- **Supports CAN 2.0a / b protocol at speeds of up to 1Mbit/s**
- **Asynchronous and synchronous (SPI) interfaces**
- **Binary and text modes (text mode in v2 only)**
- **User configurable with settings stored in EEPROM**
- **5V / 3V logic options**
- **Reset out option**
- **Status LED**

This compact CAN bus controller module has been designed to interface to our TFT and VFD modules. It can be used in other CAN bus applications where a CMOS asynchronous or synchronous (SPI) serial interface (either 5V or 3V) is required. In binary mode all data received from the CAN bus with an ID matching the configured ID is sent (via a 256 byte receive buffer) to the the async / SPI interface. All data received from the asynchronous interface is transmitted (via a 256 byte transmit buffer) on the CAN bus with the configured ID and type. Settings such as CAN ID, CAN bit rate, asynchronous baud rate, SPI mode can be set via the CAN bus or the asynchronous interface. In text mode the async / SPI communication is in the form of ASCII text packets.

ELECTRICAL SPECIFICATION (V_{logic} = 3V or 5V, default = 3V)

Parameter	Symbol	Value	Condition
Power Supply Voltage	VCC	5.0V +/- 5%	GND=0V
Power Supply Current	ICC	100 mA (typ)	VCC=5V
Logic High Input	VIH	0.6xV_{logic} min. / V_{logic}+0.5V max.	VCC=5V
Logic Low Input	VIL	-0.5V min. / 0.2xV_{logic} max.	VCC=5V
Logic High Output	VOH	0.8xV_{logic} min.	IOH=-10µA
Logic Low Output	VOL	0.2xV_{logic} max.	IOL=4mA
CAN Low Output	CVOL	0.5V min. / 1.25V max.	Dominant
CAN High Output	CVOH	2.45V min. / V_{logic} max.	Dominant
CAN Low Output	CVOL	2.3V typ.	Recessive
CAN High Output	CVOH	2.3V typ.	Recessive
CAN Input Differential (High)	CVIDH	500mV min.	VCC=5V
CAN Input Differential (Low)	CVIDL	900mV min.	VCC=5V
CAN Input Hysteresis	CVIhys	100mV typ.	VCC=5V
CAN Input resistance	CR_I	35KΩ typ.	VCC=5V

CON1

Pin	Signal
1	VCC
2	SCK
3	TXD
4	MOSI
5	GND
6	MISO
7	RXD
8	RESET
9	MB
10	HB

CON2

Pin	Signal
1	NC
2	CAN-L
3	GND
4	NC
5	NC
6	GND
7	CAN-H
8	NC
9	VCC

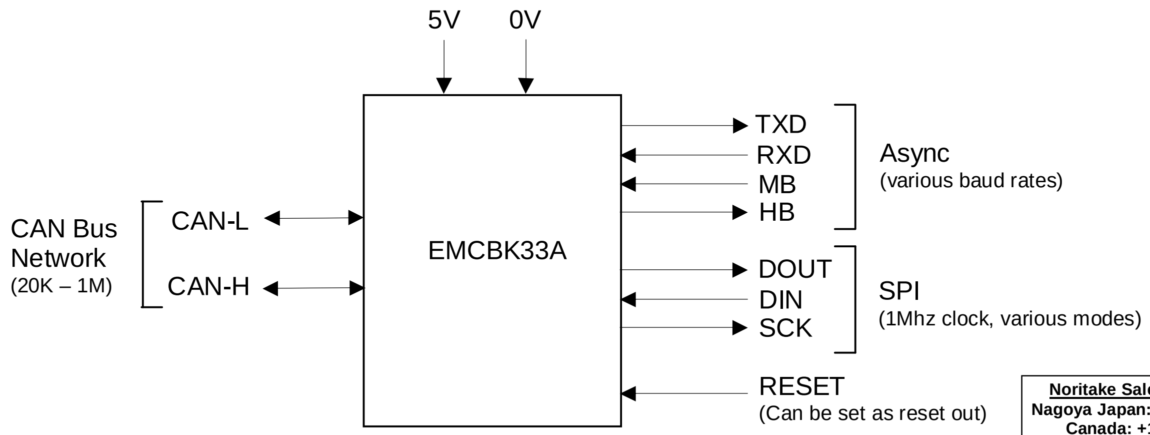
ENVIRONMENTAL SPECIFICATION

Parameter	Value
Operating Temperature	–40°C to +85°C
Storage Temperature	–40°C to +85°C
Operating Humidity	20 to 85% RH @ 25°C non condensing

Status LED functionality

Colour	Controller state
Green	Idle / ready
Blue	Data transfer (CAN -> Async/SPI)
Magenta	Data transfer (Async -> CAN)
Yellow	Configuration data received
Cyan	CAN bus passive error state
Red	CAN bus receive / transmit error
Off	CAN bus off error state

FUNCTIONAL DIAGRAM



5V
0V
EMCBK33A
CAN Bus Network (20K – 1M)
CAN-L
CAN-H
TXD
RXD
MB
HB
Async (various baud rates)
DOUT
DIN
SCK
SPI (1Mhz clock, various modes)
RESET
(Can be set as reset out)

Noritake Sales Office Tel Nos
Nagoya Japan: +81 (0)52-561-9867
Canada: +1-416-291-2946
Chicago USA: +1-847-439-9020
Munchen (D): +49 (0)89-3214-290
Itron UK: +44 (0)1493 601144
Rest Europe: +49 (0)61-0520-9220
www.noritake-itron.com

Subject to change without notice.
Doc Ref: 42420 Iss:2 10Jun11

Operating modes

Communication with the EMCBK33A can be in binary or text format. The factory default mode is binary (this is the only mode available in the v1 product). When the operating mode is changed the mode is stored in EEPROM.
In binary mode every CAN data message received with the configured receive ID is processed and the data bytes extracted from the packet and sent in raw binary format to the target device. Each byte received from the target device is sent on the CAN bus with the configured transmit ID as a single 1 byte packet. Although an ID mask can be set to allow the reception of messages from a range of CAN ID's the CAN ID itself is not communicated to the target.
Text mode allows more flexibility and maintains all information about the CAN packet when transferring to / from the target device. A disadvantage of text mode is more bytes are transferred on the target side as messages are sent as ASCII text (ASCII HEX for ID and data values). Depending on the CAN bitrate and async baud rate this can lead to a limited throughput of data. Using receive ID filtering with appropriate mask values can help by limiting the range of ID's accepted. Text mode is preferred when using the EMCBK33A with the iSMART TFT modules because processing of the incoming data on the async port AS1 can be triggered on <CR> and the preceding text packet analyzed using a series of CALC commands to determine packet type, ID, length, payload data, etc. Transmission is also more flexible as a CAN message can be sent with any ID on the fly using the same, easy to read text format.

Binary mode

Configuration

Default settings are CAN ID = 155h, both 11 bit and 29 bit ID CAN messages received, 11 bit CAN messages transmitted, CAN bit rate of 1Mbit/s, asynchronous baud rate = 38400, SPI mode 0 (fixed 1Mhz clock speed). When a configuration command is received the new configuration settings are stored in EEPROM and the controller performs an internal reset and the new settings are used immediately. Only a small selection of settings are available if configured using the internal jumper links.

Changing the configuration via the CAN bus

Setup via the CAN bus is achieved by sending a single CAN message with ID=0h (either 11 or 29 bit ID accepted). The data length of all CAN configuration messages must be 8 bytes (unused parameter bytes at the end of the packet must be sent but can be any value). The format of the 8 data bytes is as follows :-

Byte 1	*Byte 2*	*Byte 3*	*Byte 4*	*Byte 5*	*Byte 6*	*Byte 7*	*Byte 8*
'C'	'f'	type	param1	param2	param3	param4	param5

If a message is received via CAN with an ID of 0h that does not match this format no further messages with ID of 0h are processed until the controller is powered OFF / ON.

Changing the configuration via the asynchronous interface

Setup via the asynchronous interface is achieved by sending a configuration sequence at the currently selected baud rate. Unlike setting via the CAN bus the packet length depends on the configuration type.

Byte 1	*Byte 2*	*Byte 3*	*Byte 4*	*Byte 5 – Byte n*	
1Bh	'C'	'f'	type	param1	param*n*

Configuration commands – [1Bh] is sent first when sending the configuration via the asynchronous interface.

'A' **Set Async baud rate**

Set the asynchronous baud rate

[1Bh] 43h 66h 41h baud

Baud -

4800	00h
9600	01h
19200	02h
38400	03h
57600	04h
115200	05h

'B' **Set CAN bit rate**

Set the CAN bit rate from a range of common values. Different bit rates are possible by using the custom option.

[1Bh] 43h 66h 42h rate
[1Bh] 43h 66h 42h rate [b1] [b2] [b3]

Rate -

20K	00h
50K	01h
100K	02h
125K	03h
200K	04h
250K	05h
500K	06h
1M	07h
Custom	FFh (uses b1, b2 and b3 to create non-standard bit rate – contact us for details)

‘C’ Set CAN receive / transmit modes and select either Async / SPI mode

Set either asynchronous or SPI mode. Set whether 11 bit or 29 bit ID will be used for CAN transmit. Set the CAN receiver to accept 11 bit, 29 bit or both formats.

[1Bh] 43h 66h 43h mode

Mode -

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
A/S	-	-	-	-	CANTX	CANRX2	CANRX1

A/S	0=Async, 1=SPI
CANTX	0=11 bit CAN transmit mode, 1=29 bit CAN transmit mode
CANRX	01=11 bit CAN receive mode, 10=29 bit CAN receive mode, 11=both 11 and 29 bit CAN receive mode

‘Y’ Enter text mode – controller is reset after this command

After this command is sent the controller operates in text mode.

‘Z’ Reset controller to factory defaults

‘R’ Set CAN receive ID (11 bit)

Specify the 11 bit ID to use for CAN receive.

[1Bh] 43h 66h 52h 00h [b1] [b2]

b1	11 bit ID (bits 10-8)
b2	11 bit ID (bits 7-0)

‘R’ Set CAN receive ID (29 bit)

Specify the 29 bit ID to use for CAN receive.

[1Bh] 43h 66h 52h 01h [b1] [b2] [b3] [b4]

b1	29 bit ID (bits 28-24)
b2	29 bit ID (bits 23-16)
b3	29 bit ID (bits 15-8)
b4	29 bit ID (bits 7-0)

‘T’ Set CAN transmit ID (11 bit)

Specify the 11 bit ID to use for CAN transmit.

[1Bh] 43h 66h 54h 00h [b1] [b2]

b1	11 bit ID (bits 10-8)
b2	11 bit ID (bits 7-0)

‘T’ Set CAN transmit ID (29 bit)

Specify the 29 bit ID to use for CAN transmit.

[1Bh] 43h 66h 54h 01h [b1] [b2] [b3] [b4]

b1	29 bit ID (bits 28-24)
b2	29 bit ID (bits 23-16)
b3	29 bit ID (bits 15-8)
b4	29 bit ID (bits 7-0)

‘S’ Set SPI mode

Sets SPI parameters. SPI is currently operational in transmit only (EMCBK33A -> Module). Data order, polarity and phase can be set. Clock frequency is fixed at 1Mhz.

[1Bh] 43h 66h 53h mode

Mode

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	-	-	DORD	POL	PHASE

DORD	0=MSB sent first, 1=LSB sent first
POL	0=clock idle low, 1=clock idle high
PHASE	0=leading edge clock, 1=trailing edge clock

Example setup

The controller is to be used on a CAN 2.0a (11 bit ID, receive and transmit) bus network operating at 50K. The controller will be set to respond to a CAN ID of 723h. It will connect to a VFD module via the asynchronous interface with a baud rate of 19200. The following configuration commands should be sent to setup the controller :-

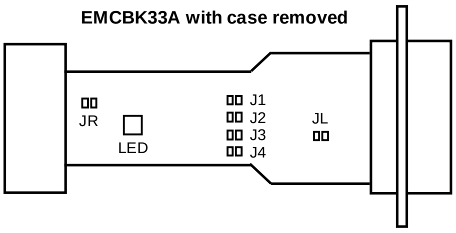
Setup via CAN bus	
43h 66h 41h 02h 00h 00h 00h 00h	Set async baud rate
43h 66h 42h 01h 00h 00h 00h 00h	Set CAN bit rate
43h 66h 43h 01h 00h 00h 00h 00h	Set modes
43h 66h 52h 00h 07h 23h 00h 00h	Set 11 bit CAN receive ID
43h 66h 54h 00h 07h 23h 00h 00h	Set 11 bit CAN transmit ID

Setup via the asynchronous interface	
1Bh 43h 66h 41h 02h	Set async baud rate
1Bh 43h 66h 42h 01h	Set CAN bit rate
1Bh 43h 66h 43h 01h	Set modes
1Bh 43h 66h 52h 00h 07h 23h	Set 11 bit CAN receive ID
1Bh 43h 66h 54h 00h 07h 23h	Set 11 bit CAN transmit ID

Changing the configuration via the internal jumper links

Only a small selection of possible settings are available if configured using the internal jumper links J1 – J4.

EMCBK33A with case removed



INTERNAL JUMPER LINK SETTINGS

Jumper	Connect	Function
JL	Open	3V signal level*
	Link	5V signal level
JR	Open	Reset out (future option)
	Link	Reset in*
J1	Open	Settings from EEPROM*
	Link	Settings from J2,J3,J4
J2	Open	19200 baud
	Link	115200 baud
J3	Open	2AAh CAN ID
	Link	3BCh CAN ID
J4	Open	100K CAN speed
	Link	250K CAN speed

* = default link setting

<u>Text mode</u> (v2 only)

Overview

In text mode all communication between the EMCBK33A v2 and the target device is in the form of ASCII text packets.

Commands to controller

O<CR> Open the CAN channel (LED bright cyan) (stored in EEPROM, default = closed)

CAN messages can be sent and received. The bitrate must be set up before the CAN channel is opened.

C<CR> Close the CAN channel (LED dim cyan) (stored in EEPROM, default = closed)

No CAN messages can be sent or received.

Br<CR> Set standard CAN bit rate (stored in EEPROM, default = 1M)

r=0	20K
r=1	50K
r=2	100K
r=3	125K
r=4	200K
r=5	250K
r=6	500K
r=7	1M

This command should be sent while the CAN channel is closed.

bxxyyzz<CR> Sets custom CAN bit rate (stored in EEPROM)

Register 1 xx	xddddddx	d=divider
Register 2 yy	xssxpppx	ss=SJW, p=propogation
Register 3 zz	xhhhHHHx	h=phase2, H=phase1

Bitrate (bps) = 16000000 / [(divider+1) x (1+ (p+1) + (h+1) + (H+1))]

Example – 88.8Kbps

Divider (to derive TQ from master clock) = 8

xx	x001000x	
yy	x10x111x	
zz	x100101x	**b**104E4A<CR>

This command should be sent while the CAN channel is closed.

txxxl[data]<CR>	Send a standard 11 bit data CAN message with ID of xxx, length l and optional data Only valid if CAN channel is open. Example t100410203040<CR> sends an 11 bit packet with ID of 100h and 4 data bytes 10h, 20h, 30h, 40h
Txxxxxxxxl[data]<CR>	Send an extended 29 bit data CAN message with ID of xxxxxxxx, length l and optional data Only valid if CAN channel is open. Example T000012FE1F5<CR> sends a 29 bit packet with ID of 000012FEh and 1 data bytes F5h
rxxxl<CR>	Send a standard 11 bit remote CAN message with ID of xxx and length l Only valid if CAN channel is open. Example r7FE3<CR> sends an 11 bit RTR packet with ID of 7FEh with length of 3
Rxxxxxxxxl<CR>	Send an extended 29 bit remote CAN message with ID of xxxxxxxx and length l Only valid if CAN channel is open. Example r100000FE2<CR> sends a 29 bit RTR packet with ID of 100000FEh with length of 2
ixxx<CR>	Set standard 11 bit receive ID (stored in EEPROM, default = 000) Example i7FF<CR> sets 11 bit receive ID to 7FFh (see corresponding mask command for message filtering)
mxxx<CR>	Set standard 11 bit receive MASK (stored in EEPROM, default = 000). Use in conjunction with receive ID to set up receive filtering.
Ixxxxxxxx<CR>	Set extended 29 bit receive ID (stored in EEPROM, default = 00000000)
Mxxxxxxxx<CR>	Set extended 29 bit receive MASK (stored in EEPROM, default = 00000000)
Xn<CR>	Enable / disable timestamp info on RX packets (0=disable, 1=enable) (stored in EEPROM, default = disabled)
Un<CR>	Sets async baud rate (stored in EEPROM, default = 38400) 0=4800 1=9600 2=19200 3=38400 4=57600 5=115200
Q<CR>	Return version information and current configuration
F<CR>	Return CAN status flags in the form Fxx<CR> bit 7 - bit 6 timestamp enabled bit 5 CAN enabled bit 4 CAN initiated bit 3 - bit 2 - bit 1 bus off error bit 0 passive error
Y<CR>	Enter binary mode – controller is reset after this command
Z<CR>	Reset controller to factory defaults
Lrrggbb<CR>	LED test - sets colour to RGB HEX values

Format of commands received

dxxxl[data][tttt]<CR>	Standard data CAN message received with ID, length, optional data and timestamp data if enabled
Dxxxxxxxxl[data][tttt]<CR>	Extended data CAN message received with ID, length, optional data and timestamp data if enabled
rxxxl[tttt]<CR>	Standard remote CAN message received with ID, length and timestamp data if enabled
Rxxxxxxxxl[tttt]<CR>	Extended remote CAN message received with ID, length and timestamp data if enabled

Example setup

For use in an 250Kbps 11 bit system. Filter so only packets with ID's between 170h and 17Fh are received. Enable receive timestamp.

B5<CR>	Set bitrate
i170<CR>	Set ID
m7F0<CR>	Set mask
X1<CR>	Enable timestamp
O<CR>	Open CAN channel