Handbook screen CAN 2×16 characters

Trilys

trilys.com

contact@trilys.com

1115, Rue René Descartes
13857 Aix-En-Provence
France

Last edition: 2018-03-30

Table of contents

I - Hardware

This section is about the hardware components of the screen 2×16 (2 lines / 16 columns) from Trilys.

I.1 - Screen features

The screen is composed of two lines (or four for the version 4×16) and sixteen columns.

Every characters is made of 5×8 pixels.

Figure 1 - Characters of the screen

Contrast and brightness are settable with CAN commands.

The refresh rate of the screen is about 100ms.

The total consumption is about 28mA with maximum brightness and under 9mA with backlight turned off.

I.2 - Buttons

I.2.1 - Three buttons version

Figure 2 - Screen with 3 buttons

The identification of these three buttons is described later in this datasheet.

I.2.2 - Six buttons version

Figure 3 - Screen with 6 buttons

As the previous comment, the identification of these six buttons is described later in this datasheet.

I.3 - Connectors

On the back side of the screen, there are two RJ45 connectors. Each connectors are connected to the same signals, so there is no difference between them. With this configuration you can use more devices on the same CANBus.

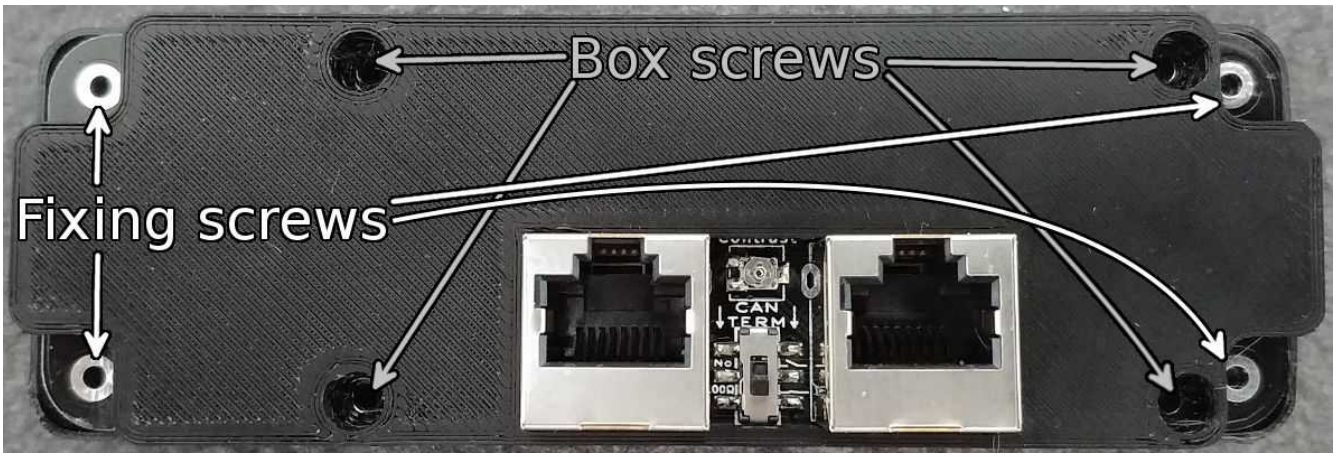


Figure 4 - Back of the screen

Any straight RJ45 (not crossover) connector can be used to communicate with Trilys devices.

The RJ45 pinout is:

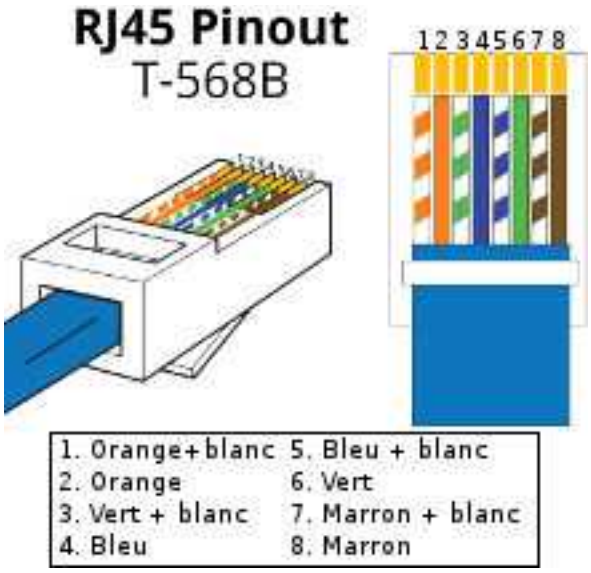


Figure 5 - RJ45

Same colors wires are twisted pair.

N°	Name	Description
1	NC	Not connected.
2	NC	Not connected.
3	NC	Not connected.
4	+5V	Power supply input.
5	GND	Ground.
6	NC	Not connected.
7	CAN_L	CANBus low.
8	CAN_H	CANBus high.

Warning: Do not connect this RJ45 with any other RJ45 port! Signals must be compatibles before trying any connection. IE: Your computer's Ethernet is not compatible.

I.4 - CANBus terminator switch

Every CANBus needs a terminal resistor. If the screen is a termination, you need to enable its terminal resistor.

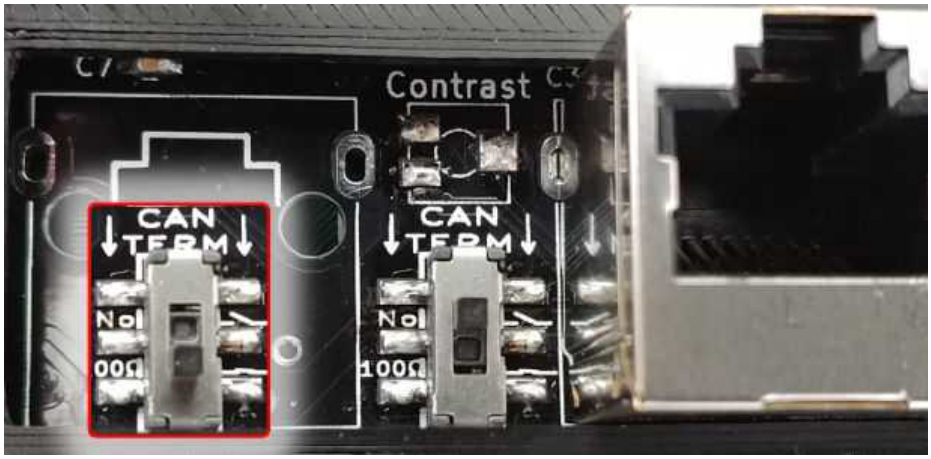


Figure 6 - Enable or disable the terminal resistor

If the switch under the "CAN TERM" mark is pushed to the top (cf Figure 6, no red frame), the termination is disabled. On the contrary, if the switch is pushed down (in red frame), the termination is enabled.

So you must enable the termination (switch pushed down) if the screen is the termination.

II - CANBus Communication

The screen CAN 2×16 uses some functions from CANOpen.

II.1 - CANOpen Trilys

The seven last bits of the CAN address is the identification number, called "*node ID*".

Example: the address "0x190" (=0x180+0x10) is sent by node ID 0x10 and the message is a TPDO "T*ransmit Process Data Object #1"* (0x180). In this case, this is a click on a button.

II.2 - CANOpen settings

You can change some settings saved into the eeprom of the screen. **If you want to factory reset the screen, you need to push the "Up" and "Down" buttons during the bootup. The screen should write "Restauration effectuee" which means "Factory reset".**

Setting any settings uses the CANOpen method. You need to send the message to the address 0x600+NODE_ID (default is 0x610). The message is made of four groups which are command, address, sub-address and message.

- The first byte is the command:
  - Read request: 0x40. Response: 0x80 in case of error, 0x4F if the message is 1 byte, 0x4B if it is 2 bytes, 0x43 if it is 4 bytes.
  - Write request: 0x2F if the message is 1 byte, 0x2B if it is 2 bytes, 0x23 if it is 4 bytes. Response: 0x80 in case of error, 0x60 if message is saved.
- The second and third bytes are the address (little-endian format - the less significant byte is sent first).
- The fourth byte is the sub-address.
- The four lasts bytes are the message (little-endian format - the less significant byte is sent first).

Summary of commands:

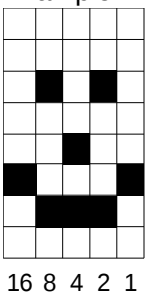
Access to data			
Command	1 Octet	2 Octets	4 Octets
Write request	0x2F	0x2B	0x23
Write response	0x60	0x60	0x60
Read request	0x40	0x40	0x40
Read response	0x4F	0x4B	0x43
Error response	0x80	0x80	0x80

Example: To send the value 100 (0x64, size of 1 byte) to the address 0x4600 and sub-address 0, of the node ID 0x10, you need to send 0x2F0046006 4 to the address 0x610. The screen should respond 0x60004600 64 from the address 0x590 if it accepts this message, or 0x80004600.. if it refuses.

Summary of addresses:

Data address			
Address	Sub-address	Description	Interval
0x1005	0	Sync ID: Send a TPDO2 on each reception.	[0;0xff]
0x1017	0	Period for *heartbeat* timer in seconds.	[1; 255]
0x4600	0	Brightness of the screen	[0; 100]
	1	Contrast of the screen	[0; 100]
	2	Delay before turn off backlight with no activities. (0=Off) in seconds.	[0; 65535]
	3	1 for reboot screen. (2=Force reboot without saving).	[0; 2]
0x4601	0+8*num	Special character #num, line 0 (top).	[0; 31]
	1+8*num	Special character #num, line 1.	[0; 31]
	2+8*num	Special character #num, line 2.	[0; 31]
	3+8*num	Special character #num, line 3.	[0; 31]
	4+8*num	Special character #num, line 4.	[0; 31]
	5+8*num	Special character #num, line 5.	[0; 31]
	6+8*num	Special character #num, line 6.	[0; 31]
	7+8*num	Special character #num, line 7 (bottom).	[0; 31]
0x4602	0	Status of button: up (1=pushed, 0=released)	[0; 1]
	..	Status of button: right, down, left, enter (1=pushed, 0=released)	[0; 1]
	5	Status of button cancel (1=pushed, 0=released)	[0; 1]
	8	The 6 previous bits in one byte.	[0; 0x3F]
0x5800	0	Enable TPDO1 (=1).	[0; 1]
0x5801	0	Node ID used as CANOpen.	[1; 0x7E]
	1	CAN Speed: 1=1kk,2=500k,3=250k,4=125k,6=50k,7=20k,8=10k.	[1; 8]

Example: We are going to save this smiley in the screen.



Each character is 40 pixels (8 lines and 5 columns).

Each lines is coded with 5 bits.

The line "16 8 4 2 1" is the value of the black pixel, corresponding to its column.

The values on the right is the result of the line.

So saving {0, 0, 10 (0xA), 0, 4, 17 (0x11), 14 (0xE), 0} would print a smiley.

To do that, we need to send the next messages to the address 0x600+NODE_ID:

0x2F01460000, 0x2F01460100, 0x2F0146020A, 0x2F01460300, 0x2F01460404, 0x2F01460511, 0x2F0146060E, 0x2F01460700.

The character is refreshed on every save of the last line (here 0x2F01460700). If the character is already printed on the screen, it would be updated. So it is not possible to print more than height specials characters at the same time.

Now, to print this character at top left (node ID 0x10), we need to send 0x01010E00 to the address 0x510 (more details on II.4.2.Special modification on the screen).

II.2.1 - CAN Identification number (CAN ID) x5801:00

The default node ID is 0x10. The ID can be any hexadecimal value between 1 and 0x7E. The CANOpen address of the ID is 0x5801, sub-address 0.

To change it, we need to send: "0x2F015800 *NEW_ID"*, to the address 0x600+OLD_*ID*.

For example, current address is 0x10, and we want to change it to 0x42, we need to send: "0x2F01580042" to the address 0x610. Then reboot with 0x2F00460301 to the old address, 0x610.

II.2.2 - CAN speed x5801:01

The default speed is 500kbps. To change it, we need to pick-up the right value in the next table:

Speed (kbps)	1000	500	250	125	50	20	10
CAN value to send	1	2	3	4	6	7	8

Example: To change the speed to 250kbps: we need to send 0x2F01580103 then reboot (0x2F00460301). After the reboot, the speed would be 250kbps.

II.3 - Transmission data

The screen sends data to the addresses 0x180+ID (TPDO1), 0x280+ID (TPDO2) and responds to SDO 0x580+ID.

II.3.1 - Heartbeat

With factory settings, every 3 seconds the screen sends heartbeats to show its status. They are at the address 0x700+ID. There are three different status:

- 0x00: Booting.
- 0x7F: End of boot. Screen is ready.
- 0x05: Every X seconds if the screen is ready.

II.3.2 - TPDO1 0x180+ID *Transmit Process Data Objects*

When any button is pressed, the screen send one TPDO (*Transmit Process Data Objects*) to the address 0x180+ID. These TPDO are 2 bytes:

- Byte 1: Value of the button, pushed or released. The most significant bit is the status of the button (1=pushed, 0=released). The seven less significant bits are the identification of the button.
- Byte 2: Value of the buttons pushed for more than 700ms.

The bits are (less significant to most significant): 1: Up, 2: Right, 3: Down, 4: Left, 5: Enter, 6: Cancel and 8: Status of transition (0: falling edge, 1: rising edge).

- 0x8100: Up pushed.
- 0x0100: Up released.
- 0x8100: Up pushed.
- 0x8101: Long press on Up (700ms later).
- 0x8101: Still long press on Up (200ms later).
- 0x0100: Up released.
- 0x8400: Down pushed.
- 0x0400: Down released.

Several buttons can be pushed at the same time.

II.3.3 - TPDO2 0x280+ID

When a *Sync ID* (0x1005) is received, the screen sends the status of all buttons. The bits are (less significant to most significant): 1: Up, 2: Right, 3: Down, 4: Left, 5: Enter, 6: Cancel:

- 0x01: Up pushed.
- 0x05: Up and down pushed.
- 0x04: Down pushed

The screen waits 5×NODE_ID after receiving a sync ID and before sending TPDO2.

II.4 - *Receive Process Data Objects*

The screen use some CANOpen RPDO. But the address 0x100, usually used for *timestamp* in CANOpen, are replaced with another RPDO.

So the next addresses 0x100+ID, 0x200+ID, 0x300+ID, 0x400+ID, 0x500+ID are RPDO.

To print any character, please refer to the ASCII table below:

HEX	Char	HEX	Char	HEX	Char	HEX	Char	HEX	Char	HEX	Char	HEX	Char
0	Spec0	28	(	38	8	48	H	58	X	68	h	78	x
1	Spec1	29	)	39	9	49	I	59	Y	69	i	79	y
2	Spec2	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
3	Spec3	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
4	Spec4	2C	,	3C	<	4C	L	5C	¥	6C	l	7C	|
5	Spec5	2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
6	Spec6	2E	.	3E	>	4E	N	5E	^	6E	n	7E	→
7	Spec7	2F	/	3F	?	4F	O	5F	_	6F	o	7F	←
20	[Space]	30	0	40	@	50	P	60	`	70	p		
21	!	31	1	41	A	51	Q	61	a	71	q		
22	"	32	2	42	B	52	R	62	b	72	r		
23	#	33	3	43	C	53	S	63	c	73	s		
24	$	34	4	44	D	54	T	64	d	74	t		
25	%	35	5	45	E	55	U	65	e	75	u		
26	&	36	6	46	F	56	V	66	f	76	v		
27	'	37	7	47	G	57	W	67	g	77	w		

Figure 7 - ASCII table of the screen

The eight first characters (Spec0 → Spec7) are the custom characters at the address 0x4601.

The number of bytes in one CAN message is 8 maximum, so there is 5 addresses to print any character anywhere on the screen.

To remove a character, you can send a space (0x20).

II.4.1 - Fast print on screen

The screen is split into four sections, where each section has its own CAN address:

- 0x100+ID: Screen top left.
- 0x200+ID: Screen top right.
- 0x300+ID: Screen bottom left.
- 0x400+ID: Screen bottom right.

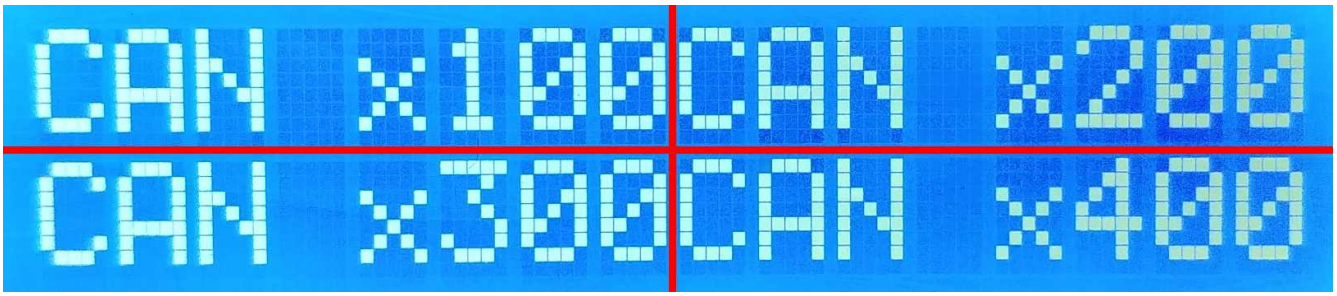


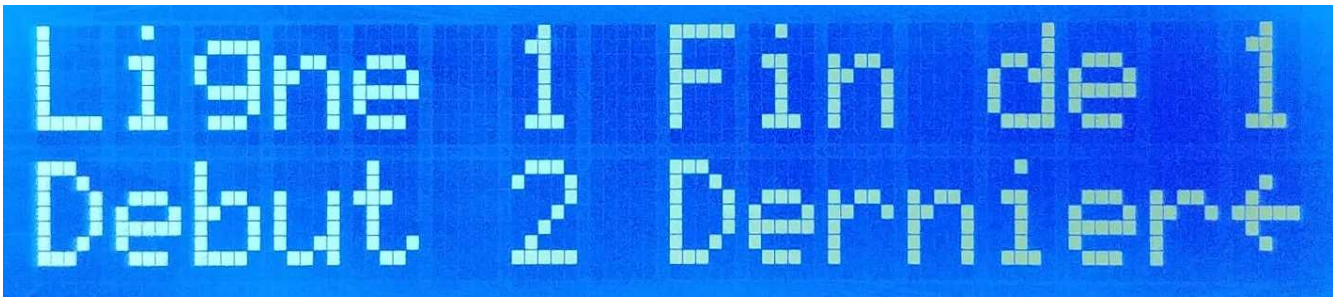
Figure 8 - CAN address of screen section

To get the previous screen, we need to send 0x43414e2078313030 to 0x100+ID, then 0x43414e2078323030 to 0x200+ID, then 0x43414e2078333030 to 0x300+ID and 0x43414e2078343030 to 0x400+ID.

Example:

- To print "Ligne 1", on top left, send 0x4C69676E652031 (cf ASCII Table) to 0x100+ID.
- To print "Fin de 1", on top right, send 0x46696E2064652031 (cf ASCII Table) to 0x200+ID.
- To print "Debut 2", on bottom left, send 0x44656275742032 (cf ASCII Table) to 0x300+ID.
- To print "Dernier←", on bottom right, send 0x4465726E6965727F (cf ASCII Table) to 0x400+ID.

The four previous messages will print:



II.4.2 - Special modification on the screen: 0x500+ID

With the last RPDO (0x500+ID), we can print a lot of data anywhere on the screen.

The eight bytes of the message are defined in the next table:

#Byte	Description	Interval
1	Number of line (1=Top, 2=Bottom)	[1; 2]
2	Number of column (1=Left, 16 (0x10) =Right)	[1; 16]
3	Type of data *	[0; 14]
4, 5, 6, 7, 8	Data	[0; 0xffffffffff]

* Types of data are listed in the next table:

Byte 3	Description
1	Data ASCII: 1 character at byte #4.
2	Data ASCII: 2 character at bytes 4 and 5.
3	Data ASCII: 3 character at bytes 4, 5 and 6.
4	Data ASCII: 4 character at bytes 4, 5, 6 and 7.
5	Data ASCII: 5 character at bytes 4, 5, 6, 7 and 8.
6	Data unsigned 8 bits (uint8) at byte 4.
7	Data signed 8 bits (int8) at byte 4.
8	Data hexadecimal at byte 4.
9	Data hexadecimal (0 padding, 2 characters) at byte 4.
10 (0x0A)	Data unsigned 16 bits (uint16) at bytes 4 and 5 ([4]+256*[5]).
11 (0x0B)	Data signed 16 bits (int16) at bytes 4 and 5 ([4]+256*[5]).
12 (0x0C)	Data unsigned 32 bits (uint32) at bytes 4, 5, 6 and 7 ([4]+256*[5]+256^2*[6]+256^3*[7]).
13 (0x0D)	Data signed 32 bits (int32) at bytes 4, 5, 6 and 7 ([4]+256*[5]+256^2*[6]+256^3*[7]).
14 (0x0E)	Print the special character from 0 to 7.

Example:

- To print a simple character (an arrow) on bottom right, send: 0x0210017F.
- Same command for 2, 3, 4 and 5 characters.
- To print a signed 8 bits (123=0x7b) on top left, send: 0x0110067B.
- To print one hexadecimal byte with 0 padding (0x2) on top and middle, send: 0x01070942.
- To print a signed 16 bits (-12345=0xCFC7) on bottom and middle, send: 0x02060BC7CF.
- To print the first special character on bottom left, send: 0x02010E00.