Model Electronic Railway Group

TECHNICAL BULLETIN: **TB CBUS-23**
Issue: **2**
Issue Date: March 2024
Author: Tim Calnun-Pratt [7133] Editor: Dave Harris [2740]

Arduino CAN Shield

The Arduino CAN Shield is a module designed to provide CAN bus (Controller Area Network) capability to 3 types of Arduino® Boards. Whilst it plays no part in the User functions of the parent Arduino, it provides CAN bus connection to enable the Arduino to interact with other CAN-based modules operating the LCB (Layout Control Bus) CBUS® protocol. This promotes the development of Arduino based CBUS system modules for both User control and accessory control.

The basic shield design is attributed to Martin Da Costa [6223] who offered it to the MERG Kit Management Team for translation as kit 110. It provides a platform for 3rd generation CBUS modules. This requires the use of libraries for the Arduino written by Duncan Greenwood [5767] and also supports sketches written with mergCBUS library by Amauri de Oliveira [4490]. Credit is duly acknowledged to these members.

The reader should be familiar with the CBUS system and/or TB CBUS-00 and TB CBUS-10.
This document follows on from the MERG kit 110 - Arduino CAN Shield Build Instructions.

Operations described in this document establish that the shield works as designed following its construction and will make use of the MERG FCU (FLiM Configuration Utility) and Arduino IDE (Integrated Development Environment) installed on a user PC.

Table of Contents

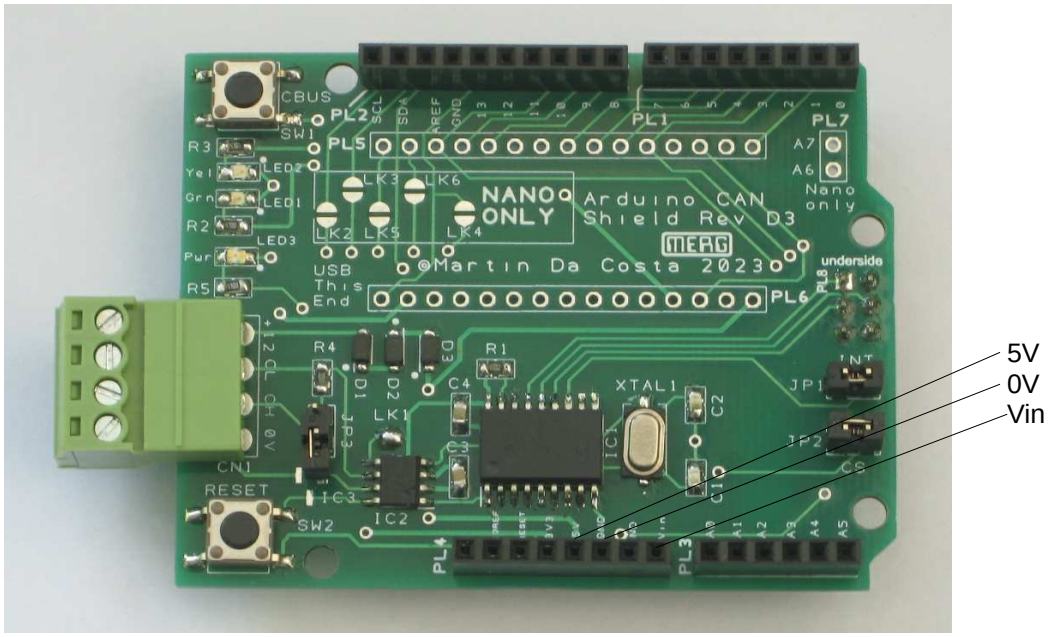


Figure 1 Arduino CAN Shield kit 110 as built

1. CAN Shield Technical Specification

CAN bus Controller chip	Microchip MCP2515, via the SPI bus to the Arduino
CAN bus Controller Firmware	Microchip Proprietary
CAN bus Controller Clock	16 MHz crystal
CAN bus transceiver chip	Microchip MCP2551 or MCP2562
Power input	12V DC via the CN1 CBUS connector. The Arduino regulator provides the shield ICs with 5V.
Connections	3.5mm pitch 4-way pluggable standard CBUS
Diagnostic LEDs	green, yellow and red (power)
PCB	Proprietary etched two layer SMD - Rev D4 – Arduino Shield form factor
Wire Links / Jumpers	5 / 3

2. Requirements

A ‘Arduino CAN Shield’ project has several prerequisites that are required to operate correctly.

- MERG Arduino CAN Shield - Kit 110
- 12V power supply.
- Your Arduino module of choice, from the applicable module list.
- CAN bus / CBUS network
- MERG CANUSB4 module, plus USB cable
- MS Windows PC, with USB port
- The MERG FCU (FLiM Configuration Utility) program installed on your Windows PC
- The Arduino IDE (Integrated Development Environment) program installed on your PC
- Test Arduino sketches and libraries – down loaded from MERG website.

***** Do not fit your Arduino board to the CAN Shield yet *****

2.1. Power Supply

A 12V SMP power block is the preferred power supply type. A 1 Amp rating is sufficient for a small network, or 3 Amp if point motors are connected to any CBUS modules.

2.2. Applicable Arduino modules

This is a standard Arduino format shield that can be built for the Arduino UNO R3 or the Arduino MEGA. As built it can be inserted on top of one of these Arduino to add a CAN bus port. The shield can also optionally be built as a mounting platform for an Arduino NANO. All options are covered in the MERG kit 110 “Build instructions”.

Note that, at the time of writing, an Arduino UNO R4 has been launched which contains a CAN bus port capability but lacks a transceiver that must be added for complete functionality.

***** This shield is not compatible with an Arduino UNO R4. *****

2.3. CAN bus / CBUS network

A CBUS network consists of a series of CAN bus equipped modules that use the CAN bus to allow CBUS® protocol modules to talk to each other. There are interface modules and specific modules to do other things. If you are not sure how to set up a network then consult TB CBUS-10. For information on how CBUS operates there are several documents, TB CBUS-00 or the “CBUS for beginners guide”.

2.3.1. CAN bus / CBUS wiring

For the purposes of shield testing it is necessary that it is connected to the CBUS network at connector CN1 (see below) which is provided in the standard MERG 4 way connector format. This connector also allows application of the necessary 12Vdc power supply. See figure 2.

An LCB (Layout Control Bus) distribution board can be used if preferred, such as Kit 151. CAN signal lines are generally a twisted pair, twisting 2 lengths of wire with a drill is the usual method used to create them.

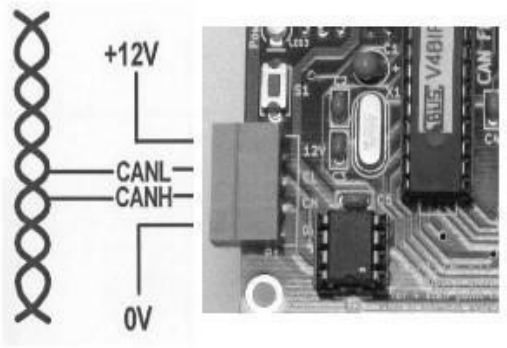


Figure 2 CAN bus connection on CN1

The CBUS comprises 4 wires, 0V, +12V, CANL & CANH. These wires interconnect to all modules.

All wires are polarity sensitive, so CANH must go to CANH and CANL to CANL and +12V to +12V and 0V to 0V. The CANL & CANH wires should ideally be a twisted pair (about 1 or 2 twists per cm) and screening is not necessary.

While it would be usual to wire the bus sequentially round the various modules, it is not essential and individual nodes can be ‘star’ or ‘spur’ connected, if this is more convenient.

The 0V of each module and power block(s) must be interconnected. This forms a common reference line for ALL the modules and the CAN wires.

Note regarding wiring. The +12V is usually red, 0V is usually black, two other different colours are needed to differentiate between CANL & CANH lines. Ensure when making up any wiring that you do not cross anything over, the order shown needs to be maintained on all board connections.

2.3.2. CAN bus termination

A CAN bus network requires that there is a termination resistance between the data lines, CANL & CANH, of 60Ω. Normally this can be provided by two 120Ω resistors in parallel or a single 68Ω resistor will suffice. Note that the shield carries an optional, single and onboard, 120Ω termination resistor R4 enabled by the, insertion of jumper JP3. If an external 68Ω resistor is applied elsewhere in the CBUS network this must not be used.

If the onboard 120Ω termination **is** used, a second 120Ω should be fitted across the CAN signal lines.

2.4. Initial Tests

Once you have constructed the required network (and the Arduino is still not fitted), connect the power block and check that you have the correct voltage and polarity on your shield. Measure Voltage with the meter probe negative on the 0V line. If the 12V input is wrong or reverse polarity, no damage is caused but the fault must be corrected. If anything gets hot, then power off and investigate and fix, before moving on.

Power down the bus and **only now fit your chosen Arduino board** and then re-power the shield.

- The red LED should illuminate on the shield. Switch off if not and investigate.
- Nothing should get hot! Turn power off if hot and investigate.
- You should measure about 2V less than that on the +12V module input, on the **`Vin`** pin of the Arduino.
- You should measure +5V on the 5V line. Turn power off if not and investigate.

2.5. CANUSB4 module

The CBUS network must also be connected, via a MERG CANUSB4 module (kit 80A), to a MS Windows PC on which the FCU will be running (See chapter 2.6).

The USB4 can be powered from the USB cable or from the CBUS +12V power. The preferred method is via USB cable. Jumper JP2 on the CANUSB4 module selects how it gets it power.

Of note is that the CBUS 0V line must be connected to all modules to maintain 0V reference integrity. Use the MS Windows Device Manager to view if the CANUSB4 has connected.

2.6. FCU

The FCU must be installed on a working MS Windows PC. It is used to configure what modules do. The FCU is currently MS Windows-based only and can be downloaded from the MERG website on the link here : https://www.merg.org.uk/merg_wiki/doku.php?id=CBUS_flim:CBUS_flim

Please ensure you use the current released version and read the installation guide if you are unfamiliar with it. The guide for both installation and use of the FCU is available for downloading at the bottom of the main page.

2.7. Arduino IDE

The Arduino IDE is an open source Integrated Development Environment which is fully supported by Arduino and can be downloaded from the install instructions page here.

https://docs.Arduino.cc/software/ide-v1/tutorials/Windows#download-the-Arduino-software-ide

** Be aware that this is IDE v1 (aka Legacy or Classic) and there is also an IDE v2. This TB covers IDE v1 **

The IDE can be installed on Windows, Linux, MacOS or cloud based. But, it can be quite happily installed on the same MS Windows PC as the FCU and both can be running at the same time. However, they will need to use separate COM ports. You can use either two physical ports on the PC, or a port extender. Windows will allocate one COM port number to the FCU and one to the Arduino board.

Once installed it is necessary to install the libraries listed earlier. Refer to the link below and follow the instructions.

There is a search facility and in general searching CBUS will bring up most of what is needed. There is a ZIP file on the main kit page containing the required libraries. Install these by expanding the main file to leave individual zip files, then install these using the IDE install zip file option in the libraries section. They must be installed one at a time. Install CBUS2515 last.

https://docs.Arduino.cc/software/ide-v1/tutorials/Arduino-ide-v1-basics#libraries.

This process may take some time, please be patient and ensure that the libraries are loaded correctly.

2.7.1. Required Arduino libraries

Streaming	C++ stream style output, v5, (http://arduiniana.org/libraries/streaming/)
Bounce2	Debounce of switch inputs
CBUSconfig	Module configuration
CBUS	CBUS Class
cbusdefs	Definition of CBUS codes
CBUSParams	Manage CBUS parameters
CBUSSwitch	Manage CBUS switch
CBUSLED	Manage CBUS indicator yellow/green LEDs
ACAN2515	library to support the MCP2515/25625 CAN controller IC
CBUS2515	CAN controller and CBUS class

3. First Test Sketch - CANshield

The purpose of this initial sketch is to confirm the correct operation of the two CAN chips installed on the shield.

From the main kit page (MERG Kits:110) and download cansheild.zip and extract the zipped files to a known location. Use the Arduino IDE to set up the board and COM port.

https://support.Arduino.cc/hc/en-us/articles/4406856349970-Select-board-and-port-in-Arduino-IDE

Use the IDE file menu to open the sketch you previously saved. Upload it to your board see here: -

https://support.Arduino.cc/hc/en-us/articles/4733418441116-Upload-a-sketch-in-Arduino-IDE#upload

Once uploaded you should see the green LED on the shield illuminate. If you get anything else, use the serial monitor in the IDE to send a ‘z’ command twice within 2 seconds (See chapter 7) to reset the Arduino module and reload the sketch. The Arduino is now termed to be in CBUS SLIM mode.

Ensure you have set up the FCU and it is running and connected to the shield.

Press and hold the CBUS push button (NOT the RESET button) until the green LED extinguishes and the yellow LED starts to flash. You should see a pop-up box in the main FCU window. See figure 3 below.

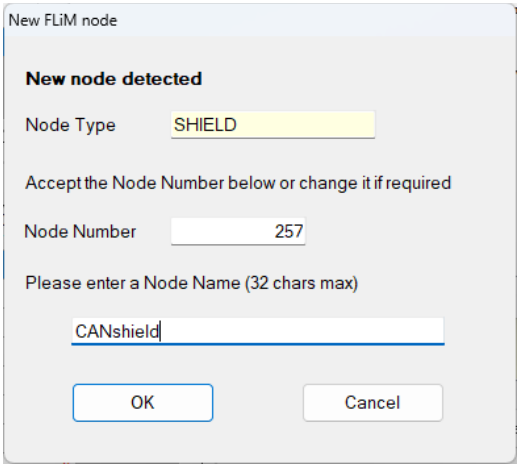


Figure 3 - New FLiM node pop-up

It is essential that this process is completed within 30 seconds. There is a time delay written into the software that will revert the Arduino back to SLiM mode (green LED lit).

If this happens click the cancel button and try again.

The Node number may be different. For now just accept what's there.

Enter a Node name e.g. CANshield and click the OK button.

The yellow LED should now stop flashing and remain lit. The Arduino is said to be in FLiM mode and has been registered with the FCU.

The pop-up should disappear and the screen should display the module as in figure 4.

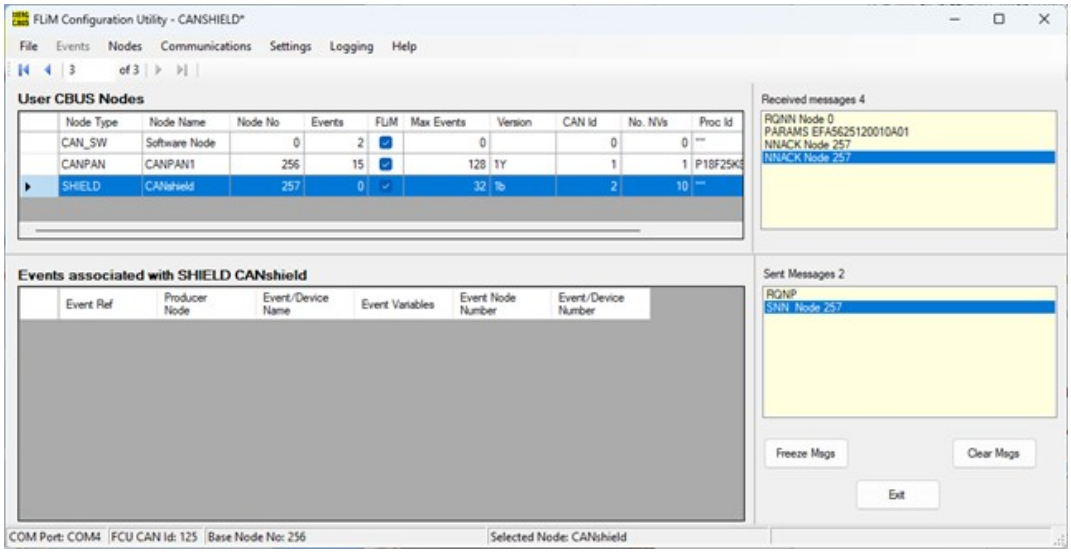


Figure 4 - CBUS nodes display

You should see the SHIELD node type in the Nodes list, with the node name you gave it.

This completes the initial test of the shield components and proves they are functioning correctly.

4. Second Sketch - CAN4in4out_shield

This sketch demonstrates the use of the Arduino attached to the shield to generate events using switches and the ability to respond to events seen on the CBUS network using LEDs.

An Arduino sketch to allocate 4 switches as input pins and 4 outputs to LEDs. This sketch provides a specific example of use that will enable users of the Arduino CAN Shield to gain familiarity with the shield use.

Key Features:

- MERG CBUS interface.
- LED flash rate selectable from event variables.
- Switch function controllable by node variables.
- Modular construction to ease adaptation to your application.
- Derived from the proven CANmINnOUT sketch https://github.com/MartinDaCosta53/CANmINnOUT

The program is written in C++, but you do not need to understand this to use the program. The program includes a library that manages the LED functionality. It can be downloaded from the main Kit 110 page.
You must make a directory called CAN4in4out_Shield. Into this you place the following files:

CAN4in4out_Shield.ino, LEDControl.h and LEDControl.cpp

It should compile and upload as per the previous sketch.

4.1. Sketch CAN4in4out pin assignments

The MCP2515 interface requires five Arduino pins to be allocated. Three of these are fixed in the architecture of the Arduino processor. One pin must be connected to an interrupt capable Arduino pin. On the shield, this is set to pin 2 but a jumper can be removed, and a wire link used to connect to another interrupt pin if desired. This may be appropriate when using a Mega with additional interrupt inputs. See chapter 5 table for the Pin assignments for the differing Arduino modules.

The CAN Controller Chip Select pin on the shield is connected to pin 10. Once again, by removing the relevant jumper, another pin can be wire linked for this function.

If you change the interrupt or chip select pins, make sure that you update the relevant pin allocations in the sketch. See chapter 5 for details.

4.2. Sketch operation

To operate the sketch, download the zip file from the MERG website shield kit page. Expand it to the folder given earlier. Load it into the Arduino IDE, the two required libraries are picked up by the IDE automatically. Load it to your chosen board.

If required issue the double ‘z’ command (See chapter 7) from the IDE to reset the board and then reload, It needs to start in SLIM mode and be registered with the FCU as before.

The Sketch once in FLiM mode and registered with the FCU provides the user the ability to produce events using the attached push buttons that the FCU can then teach to other attached modules on a CBUS network, using standard FCU drag and drop methods.

Similarly, Events from other modules or short events taught to the FCU can be taught to the Arduino based module to operate the LED’s.

5. Arduino Sketch Pin Numbers

Table 1: Arduino Sketch pin numbers

CAN Shield Function	UNO Pin	Nano Pin	Mega Pin	Comments
CBUS Green LED	4	4	4	Fixed
CBUS Yellow LED	7	7	7	Fixed
CBUS Switch	8	8	8	Fixed
CAN Controller Interrupt	2	2	19	Mega will require removal of JP1 and wire link.
CAN Controller Chip Select	10	10	10	Adjustable by removing JP2 and using wire link
SPI SI	11	11	50	Serial In - Connections are automatically configured.
SPI SO	12	12	51	Serial Out - Connections are automatically configured.
SPI SCK	13	13	52	System Clock - Connections are automatically configured.

6. Switch/LED pins

Table 2: Input/Output pins

Switch Pins	A0, A1, A2, A3	0v.
LED Pins	D3, D5, D6, D9	+5v.

7. IDE Serial Monitor command codes

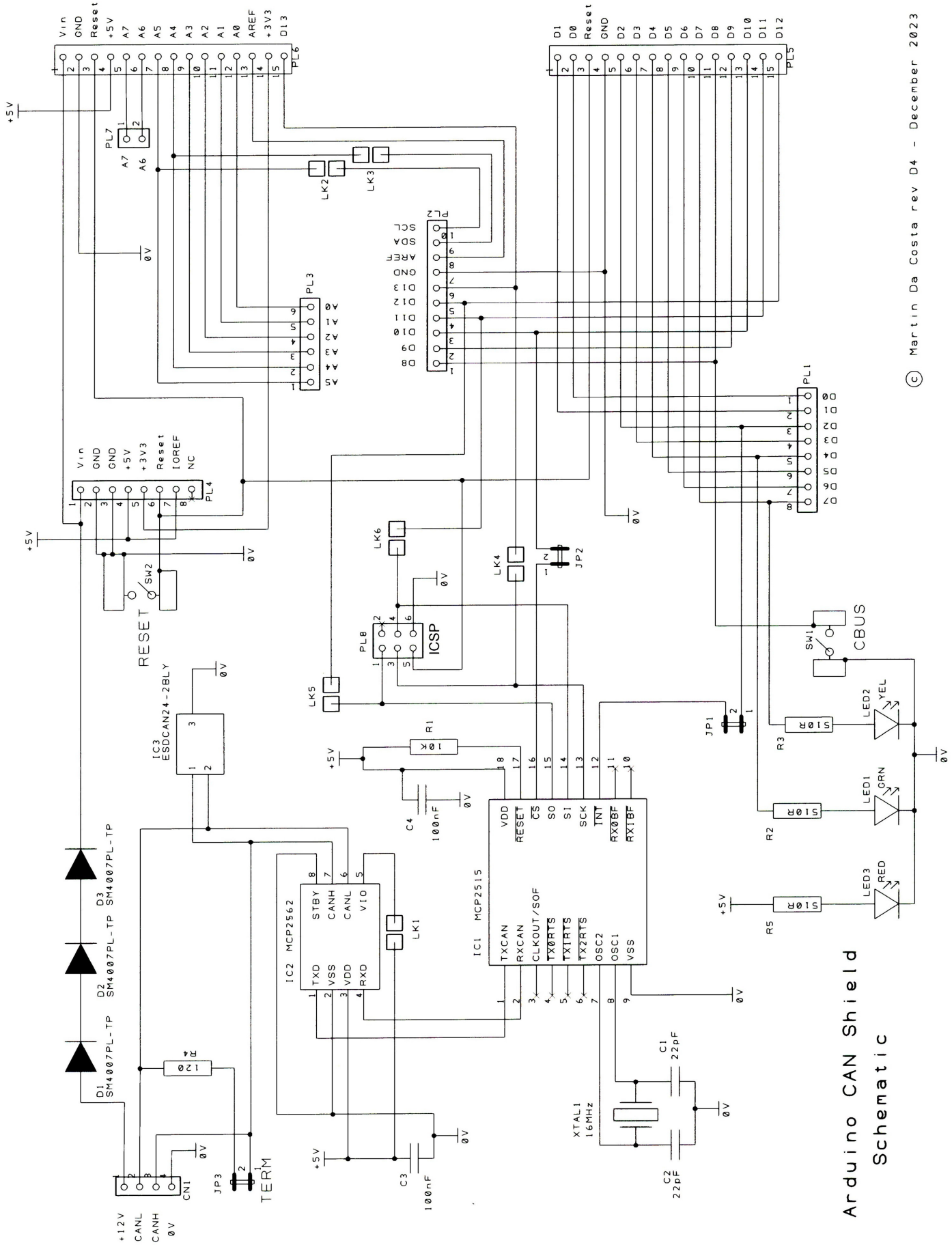
If your Arduino is connected to the Arduino IDE on your computer via the USB port, you can access information by sending a character from the IDE. The function of these characters is as follows:

Table 3: IDE Serial Monitor command codes

Character sent	This character will ...
r	cause the module to renegotiate its CBUS status by requesting a node number. The FCU will respond as it would for any other unrecognised module
z	Reset the module and clear the EEPROM. It should thus be used with care. This character needs to be sent twice within 2 seconds so that its action is confirmed
n	return the node configuration
e	return the learned event table in the EEPROM
v	return the node variables
c	return the CAN bus status
h	return the event hash table
y	Reset the CAN bus and CBUS message processing
*****	Reboot the module

m	return the amount of free memory

8. Schematic


Arduino CAN Shield
Schematic
© Martin Da Costa rev D4 - December 2023
IC1 MCP2515
IC2 MCP2562
IC3 ESDCAN24-2BLY
PL1
PL2
PL3
PL4
PL5
PL6
PL7
PL8
ICSP
RESET
CBUS
TERM
CN1
JP1
JP2
JP3
SW1
SW2
LED1 GRN
LED2 YEL
LED3 RED
D1 SM4007PL-TP
D2 SM4007PL-TP
D3 SM4007PL-TP
XTAL1 16MHz
C1 22pF
C2 22pF
C3 100nF
C4 100nF
R1 10K
R2 510R
R3 510R
R4 120
R5 510R
LK1
LK2
LK3
LK4
LK5
LK6
+5V
0V
+12V
CANL
CANH