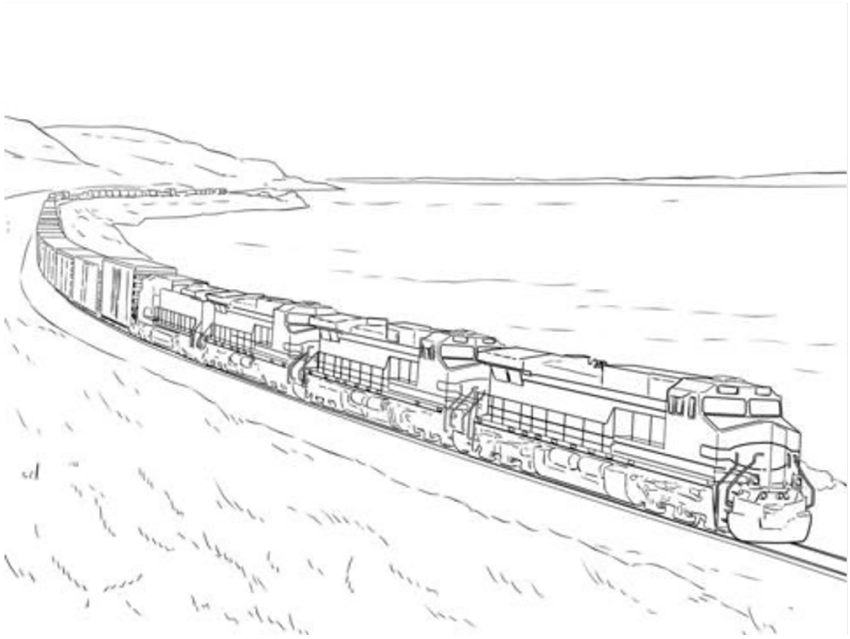


A Layout Control System for Model Railroads

Helmut Fieres December 13, 2024

Contents

1 LCS Hardware Module Design

So far we covered the general concepts, messages, protocols as well as the LCS core library and a glimpse how all of this might be used. Let's take a break from all that concepts and mostly software talk. For the software to run, hardware modules need to be built. Welcome to the next big part of this book. Here, we will talk about the lCS hardware modules. A hardware module conceptually consist of three key parts.

- communication
- controller
- function block(s)

At the center of a hardware module is the **controller**. There is a great variety of controllers and development environments available. When selecting a controller for LCS, we will talk in a minute which one was picked, its is important that there is enough CPU power and equally important a powerful development environment. A console command line interface and interfaces to load the software is also very handy for configuring, monitoring and debugging. The **communication** part implements at a minimum the LCS message bus interface for the messages to transmit between the modules. Finally, the **function block**s implement the hardware module specific capabilities.

This chapter is the first in series of chapters on hardware modules. Instead of presenting complete schematics for each major hardware module, such as the base station, we will go a slightly different route. We will first present the basic components an LCS node might need. Definitively we will need a controller and a CAN bus interface. Some LCS nodes might make use of an extended non-volatile storage, others need plenty of digital outputs. Just like Lego Blocks, all these parts should be combined easily to form the desired LCS hardware module. We will tackle each component one at a time to understand how they work. The later chapters will just combine these basic blocks with minor adaptations and perhaps some very dedicated components for their functionality.

1.1 Selecting the controller

The module designs described in this book initially used the AtMega controller platform along with the Arduino IDE to write the software. There is the Arduino IDE and by now a whole set of different processors. Since it was released, the Atmega controller family and boards such as Arduino UNO, Arduino NANO, Arduino MEGA are in widespread use. The LCS core library program and non-volatile storage requirements do place however a higher demand on the controller capabilities.

Meanwhile, the Raspberry PI Pico `(PICO)` controller joined the club. And it has a lot to offer. The PICO is a dual core controller running at up to 133 Mhz. It features a whopping 16Mbytes of flash and 264 Kbytes of main memory. There are plenty of

IO ports, and functional blocks for UARTS, SPI and I2C interfaces. What makes this controller especially interesting are the PIO state machines that allow for implementing your own I/O protocols. There is CAN bus software library built using these state machines. This way no extra CAN bus controller is needed. The PICO comes with its own software development kit and also an Arduino IDE integration is available.

As time goes by, there will be for sure more capable controller entering the market. However, when you want to complete a project versus chasing the latest controllers, you will need to pick. In our case, the PICO is the controller of choice. Its capabilities match our requirements and will be a good choice for the years to come. nevertheless, the LCS library software should be designed as independent of a particular controller as possible. More on this later.

1.2 The Controller Platform

The following table gives some guidance on the capabilities needed in our designs. This list also applies in general to other controllers.

Table 1.1: Controller Attributes

Attributes	**Notes**
Processor	For a typical module, the PICO offers plenty in terms of CPU power. Since we use a software implementation for the CAN bus, running the software in one core and the CAN bus state machine in the other will well match what the PICO offers.
Memory	Memory depends on the size requirements of the node, port and event maps and the node-specific firmware data demands. A simple module would perhaps get by with 2Kb, a base station could easily use 32Kb or even more.
Program Memory	The LCS library already uses round about 64Kb of code storage. A simple module would get by with 32Kb, a base station could easily use 128Kb and more.
External NVM	Additional NVM storage is allocated in a separate EEPROM or FRAM. The capacity is highly dependent on the module use case. External NVM components typically also require the SPI or I2C interface. Most external EEPROM chips have write cycles of more than a million. At a minimum, a chip size of 32Kb is recommended. The PICO does not offer an internal EEPROM, so an external NVM is always required.
Digital channels	The bulk of control lines is digital and used heavily. For some hardware modules, a subset of the digital pins should also be PWM capable.

Continued on next page

Attributes	Notes
Analog channels	Analog input is typically used for the power module for analog voltage measurements. Otherwise, it is perhaps optional. The PICO allows for only three inputs. If more are desired, an external multiplexer needs to be implemented.
I2C	The I2C interface comes in very handy to connect a large variety of chips. Communication to the external NVM and also to chips that implement functions such as a servo controller will require this bus.
Serial I/O	The serial I/O is used in some hardware modules for implementation of RailCom detectors. The PICO features two hardware UARTS and the option to implement more in software using the PIO state machines.
Console I/O	Serial I/O is used for console I/O. Rather than using dedicated I/O pins and a UART block in the controller, the PICO serial I/O will be implemented via the USB connector.
LEDs, Button and Dip Switches	A hardware module could make use of LEDs to indicate readiness and activity, as well as a set of switches to configure a hardware option. Not really required but certainly useful.
WLAN	WLAN is optional. But there is a PICO version with WLAN capability integrated.

1.3 Hardware Module Schematics

Hardware modules are described to large extent via schematics. The schematics shown in the following chapters are all drawn with the EasyEDA software. It is a great hardware development platform, and you can order PCBs for the final design in one easy step. Following a building block principle, the schematic diagrams will show functional components with many network endpoints where they connect to other building blocks. Each network endpoint is labelled with a name that is unique across all building blocks used in a hardware module schematic drawn. For example, "VCC-3V3" will always refer to the 3.3V power supply line. If two building blocks have an endpoint with the same name, the endpoints will be connected on all building block schematics in the final hardware module design.

A general word to the building blocks. They serve as examples of how the individual parts could be implemented and help to understand how each part works. Parts of the library software assume the presence of these blocks and how they basically work. Although the library has been written with as much as possible independence of the hardware, the final adaption of timers, serial lines, I/O pins and so on is required needs to be considered. Throughout the next chapters, you will find comments on what is perhaps generic and what would require some adaption if moving to another processor family.

1.4 Controller and Extension Board

Each node in the layout control system is a node and hence there is a controller for running the node firmware. Without a question, there will be many different nodes and as time goes by perhaps even a new controller families. However, each node would need at least some form of power supply, the CAN bus interface and depending on the storage demands and controller family, an external NVM. On top there is the node specific hardware. One approach is to design a board for each dedicated purpose. This board would include all the common portion for a LCS node and the hardware module specific portion. Another approach is to design a node controller board with extension boards that can be connected to it. In the remainder of this chapter, we will describe the main controller and extension concept. However, it is also perfectly all-right to design a hardware component with all the components integrated on one board. For a complex node such as the base station, this is a very reasonable solution. The building blocks shown in this chapter thus also form the basis for a more monolithic hardware module design. But first, let's look at the physical dimension of our boards.

picture

All boards will have a form factor of 10cm wide and 8, 12, and 16cm long. In particular, the 10x16cm board should be very familiar as the "Euro PCB" format. The main controller board has on the left side the connectors for the LCS bus and the power input. On the right side, there are two connectors toward an extension board. As described before, there are two types of extension boards. The usage of the individual connector pins are described in the upcoming chapter. To ease the hardware schematic development and ensure that all boards fit together, the PCB boards along with their connectors are available as symbols and PCB footprints in the EasyEDA library.

1.5 LCS Bus connector

Every hardware module needs the LCS bus interface to connect to the bus. Some modules may also draw power from this bus. The modules use an RJ45 connector for connecting to the bus. The bus signals can be grouped in several categories. The CAN bus differential lines represent the CAN bus. The VS line is intended for hardware modules with very little power consumptions so that they can directly be powered by the bus. The DCC signal lines are an exact copy of the DCC signal that would go to a track sent out by the DCC signal generating base station. The signal is intended to be routed from the base station to booster nodes, but also to hardware modules that analyze the DCC signal for some action. Finally there is the STOP signal line. This is a wired OR line that allows a simple button along the layout with access to this line to issue a STOP signal. The base station or any nodes interested in the signal can monitor this line. There are the following signal lines.

Table 1.2: Bus Connector Pins

Pin	Name	Purpose
1	DCC-Sig-1	The DCC signal labelled "+"
2	DCC-Sig-2	The DCC signal labelled "-"
3	GND	Common ground
4	RSV	reserved for future extensions.
5	RSV	reserved for future extensions.
6	PWR	The bus supplied 12V power line. This line is intended for devices with very little power consumption to get their power from. Any other module should connect to its own power supply line.
7	CAN-L	Line L of the differential CAN bus signal.
8	CAN-H	Line H of the differential CAN bus signal.

1.6 LCSNodes Extension Board Connector

For interchangeability of extensions, there is a standardized **extension board connector** between controller and extensions. Extension boards come in two flavors. The first will have the extension connector on both sides of the board. Main controller boards and extension board will have a female connector on the right hand side. The first flavor extension board will have a matching connector on the left side. This way main controller and extension boards can be placed next to each other, just like a train. The second type of extension boards only have the connectors on the right hand side. They are intended for a backplane style layout where main controller and extension boards are plugged next to each other.

The overall concept is very similar to the the shield concept found in the Arduino or Raspberry PI universe, except that we can stack boards, as well as placing them next to each other.

The I2C interface will be the main communication method between the boards. Nevertheless, a rather rich functionality set from the controller should be available to the extension board for flexibility. There should be ports for digital input and output, analog input, PWM outputs, serial outputs and so on. The following table shows the connector pin assignments for the communication between a main controller board and extension boards. All boards will have a 40-pin connector organized as 2 rows of 20 pins.

Table 1.3: Controller Attributes

Pin	Name	Pin	Name	Purpose
1	**GND**	2	**GND**	Common ground pins.

Continued on next page

Pin	Name	Pin	Name	Purpose
3	**DCC-1**	4	**DCC-2**	The DCC "+" and "-" signal as generated by the DCC Signal Generator. These pins are typically driven by the base station generating the layout DCC signal.
5	**GND**	6	**GND**	Common ground pins.
7	**RST**	8	**EXT**	RST is reset line. Active Low. EXT is the external interrupt line which be raised from an extension board. Active low.
9	**ADC-0**	10	**ADC-1**	Analog input pins. The input is not protected. The analog voltage range is 0 to VCC.
11	**GND**	12	**GND**	Common ground pins.
13	**DIO-0**	14	**DIO-1**	Plain digital Pins, input or output. The pins are protected.
15	**DIO-2**	16	**DIO-3**	Plain digital Pins, input or output. The pins are protected.
17	**DIO-4**	18	**DIO-5**	Plain digital Pins, input or output. The pins are protected.
19	**DIO-6**	20	**DIO-7**	Plain digital Pins, input or output. The pins are protected.
21	**GND**	22	**GND**	Common ground pins.
23	**BI-0**	24	**BI-1**	Bus Address input lines. Up to four extension boards can be connected, the BI pins are used to determine the I2C address on the I2C extension bus.
25	**BO-0**	26	**BO-1**	Bus Address output lines. The BO lines are computed form the BI lines. If for example BI is 1:0 the BO lines will become 1:1. The starting output pins values are 1:1.
27	**SCL**	28	**SDA**	I2C extension bus channel. The lines are protected with a serial resistor and there is a pull-up resistor to VCC.
29	**GND**	30	**GND**	Common ground pins.
31	**VCC**	32	**VCC**	VCC 5V supply to extension boards.
33	**GND**	34	**GND**	Common ground pins.

Continued on next page

Pin	Name	Pin	Name	Purpose
35	**VS**	36	**VS**	Board Input voltage forward. These connector pins are primarily used by extension boards that need the high power input. Examples are H-Bridges on such a board or boards that have their power supply circuitry.
37	**VS**	38	**VS**	Board Input voltage forward.
39	**GND**	40	**GND**	Common ground pins.

The connectors on the boards are female connectors.

Since the connector chosen is a 2x20 connector, the signal pin numbers shown above change their row numbering, depending whether it is output or input connector. When looking at the schematics and the connector layout on the PCB, the signals on the output connector have pin 1 to 10 on the leftmost row, and on the rightmost row on the input connector, such that pin 1 of the output connector connects to pin 1 of the input connector and so on. The same is true for pins 11 to 20.

There are EasyEDA symbols that offer the connector pins with you going through these details. The appendix contains EasyEDA symbols for the most common board dimensions with the connectors placed in the right location. A new projects can just start with this EasyEDA symbol. There is also the option of cascading extensions. An extension connector could therefore be on both ends of the board and pass the GND, VCC, Reset, DCC and I2C lines from board to board. For this type of board, EasyEDA symbols are also available.

A key question is how many controller pins are available to an extension board. Most of the extension boards would just need the I2C bus. However, if there is a rather complex extension board, such as a block controller shown in one of the next chapters, the IO pins needed from the controller board to the extension are many and quickly reach the limit of the extension connector. Why not place a connector with more pins on the boards ? First, a different controller may not have that many IO pins and there would be no easy mix and match between main and extension boards. Second, the majority of extension boards are rather encapsulated and most often just need the I2C bus to communicate. To find a middle ground, the 20-pin connector along with the pin capabilities outlined was chosen.

For more complex extension boards, it is perhaps the better idea to combine a main board with an extension board to one monolithic board and still keep the extension connector for other not so complex boards to attach. As a convention, only the first extension board will benefit from all signals coming from the main controller board. All follow on extension boards will only get the DCC signals, the reset line, the I2C signal and the power lines.

1.7 Track Power Connectors

In addition to the extension board connector, there is the **track power connector**. This connector is only used by the base station, block controller and associated extensions. Its purpose is to pass the track power signals from the H-bridges on the block controller (or booster) board to the extension boards. This connector is described in more detail in the base station and block controller chapter.

Table 1.4: Controller Attributes

Pin	Name	Pin	Name	Purpose
1	**DCC-SIG-B0**	2	**DCC-SIG-B0**	Bridge-0 DCC Signal "+" and "-".
3	**DCC-SIG-B1**	4	**DCC-SIG-B1**	Bridge-1 DCC Signal "+" and "-".
5	**DCC-SIG-B2**	6	**DCC-SIG-B2**	Bridge-2 DCC Signal "+" and "-".
7	**DCC-SIG-B3**	8	**DCC-SIG-B2**	Bridge-3 DCC Signal "+" and "-".

When using all four bridge signal output pairs, each each output pair is rated up to 3Amps. For high power bridges with up to 6Amps, two pairs can be combined and the number of bridges signals passed on is two.

1.8 Summary

This chapter introduced the basic architecture of a hardware modules, it connectors and board layout. A key concept is the idea of a common component, the main controller, and extension that can be connected. Nevertheless, there are good cases for combining a main controller and the extension hardware into one monolithic board. But in any case, the connectors and their purposes stay the same from board to board. While the main controller boards always have the LCS bus and power input on the left side, the extension connector and tack line connector on the right, extension boards come in two flavors.

The first extension board type has male connectors for track line and extension lines on the left side of the board while the second type has not. Both types have female track line and extension line connectors on their right. The first type can just be plugged into the main controller type boards, additional extension boards are simply plugged into the previous extension board. The second extension type is intended for a backplane type design where main controller boards as well as up to four extension board types are plugged into a backplane board. Throughout the chapter to come, you will see how easy boards can be combined using the two connectors lanes and standards behind them.

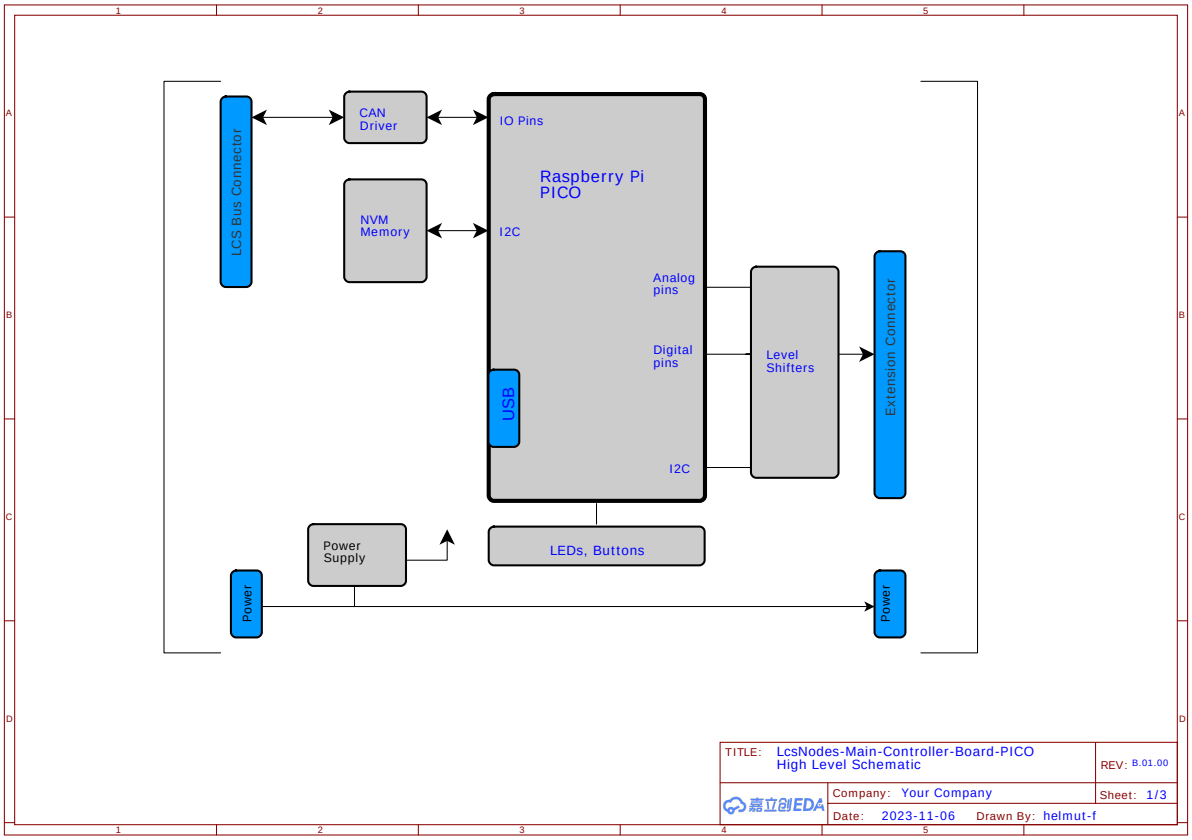
Ready for the first hardware work ? All aboard, the train leaves for the next chapter.

A Tests

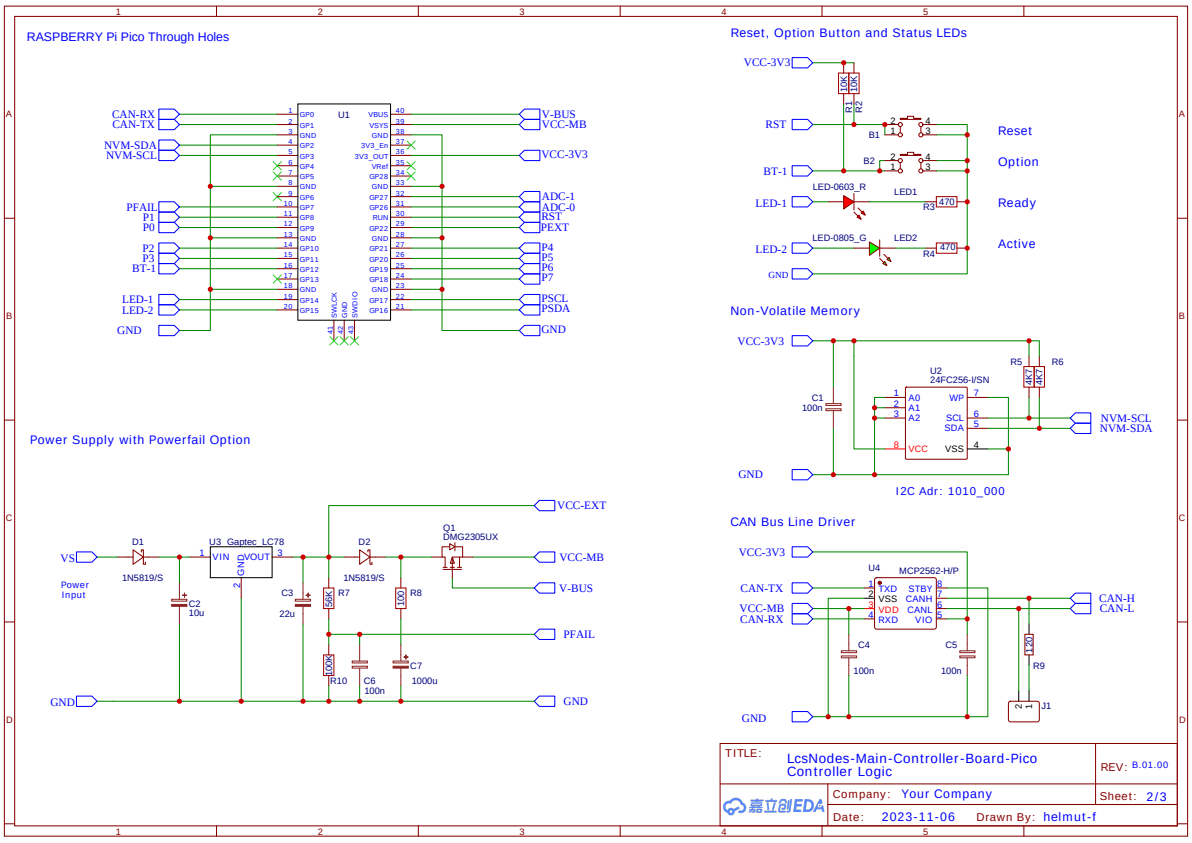
A.1 Schematics

float barrier command to ensure that text stays close to the picture but no text from after the picture.

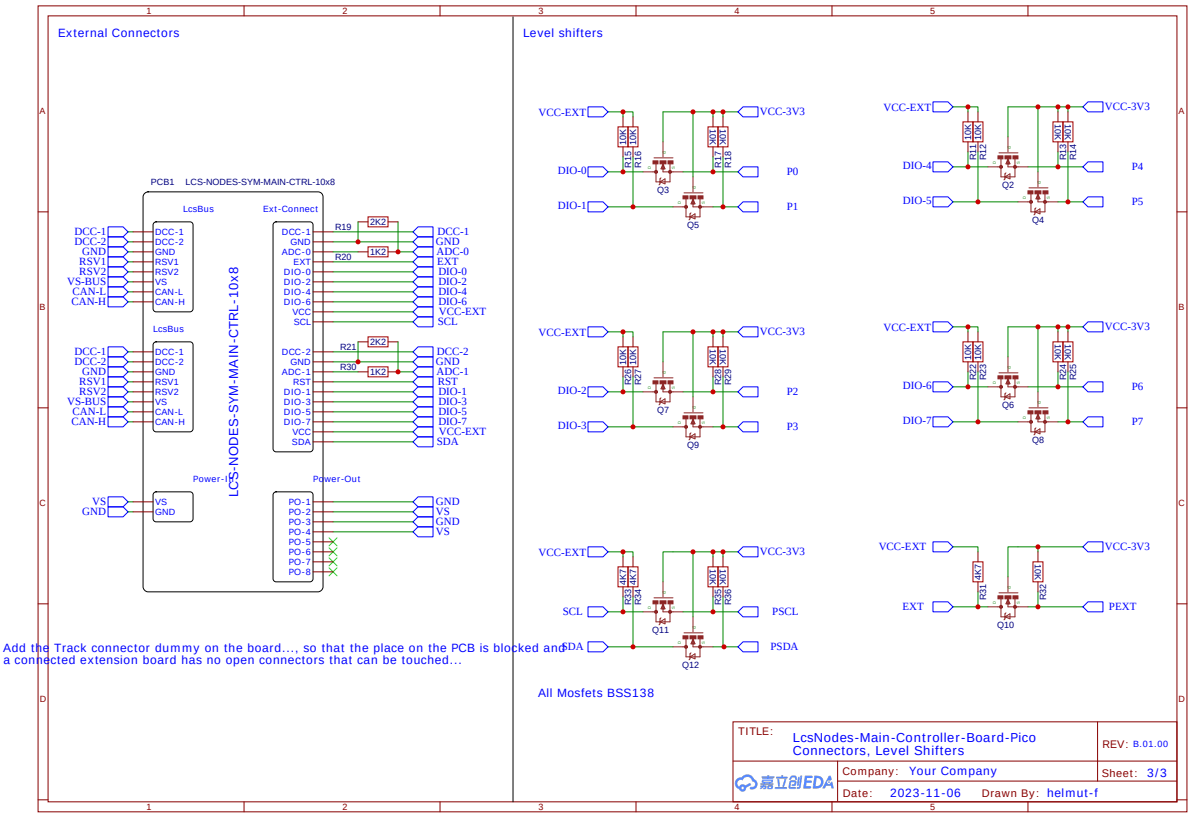
A.1.1 part 1



A.1.2 part 2


RASPBERRY Pi Pico Through Holes
CAN-RX
CAN-TX
NVM-SDA
NVM-SCL
PFAIL
P1
P0
P2
P3
BT-1
LED-1
LED-2
GND
U1
V-BUS
VCC-MB
VCC-3V3
ADC-1
ADC-0
RST
PEXT
P4
P5
P6
P7
PSCL
PSDA
GND
Reset, Option Button and Status LEDs
VCC-3V3
10K
R1
R2
RST
B1
Reset
BT-1
B2
Option
LED-1
LED-0603_R
LED1
R3
470
Ready
LED-2
LED-0805_G
LED2
R4
470
Active
GND
Non-Volatile Memory
VCC-3V3
C1
100n
U2
24FC256-I/SN
A0
A1
A2
WP
SCL
SDA
VCC
VSS
R5
R6
4K7
NVM-SCL
NVM-SDA
GND
I2C Adr: 1010_000
Power Supply with Powerfail Option
VS
D1
1N5819/S
Power Input
U3_Gaptec_LC78
VIN
VOUT
GND
C2
10u
C3
22u
D2
1N5819/S
Q1
DMG2305UX
VCC-EXT
VCC-MB
V-BUS
PFAIL
R7
56K
R8
100
R10
100K
C6
100n
C7
1000u
GND
CAN Bus Line Driver
VCC-3V3
U4
MCP2562-H/P
CAN-TX
VCC-MB
CAN-RX
TXD
VSS
VDD
RXD
STBY
CANH
CANL
VIO
CAN-H
CAN-L
C4
100n
C5
100n
120
R9
J1
GND
TITLE:
LcsNodes-Main-Controller-Board-Pico Controller Logic
REV: B.01.00
Company: Your Company
Sheet: 2/3
Date: 2023-11-06
Drawn By: helmut-f

A.1.3 part 3



A.2 Lists

A.2.1 A simple list

- First bullet point
- Second bullet point
- Third bullet point

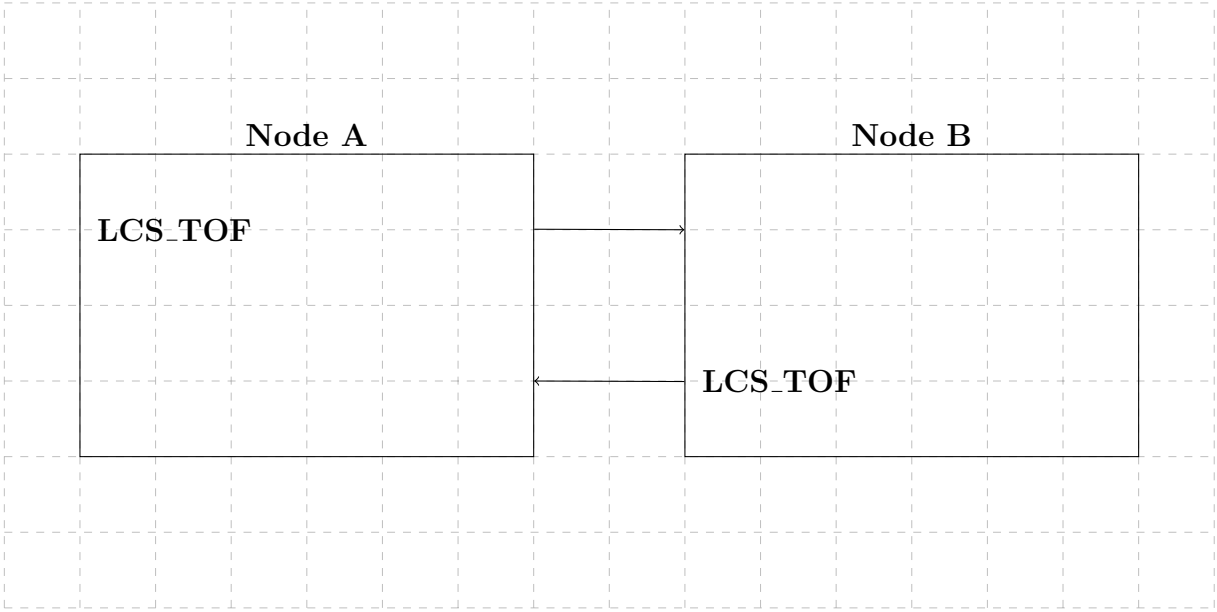
A.2.2 An instruction word layout

A little test for an instruction word layout ... will be a bit fiddling work ...

1	3	6
Test		

A.3 Protocol boxes

A bit cumbersome and we would need to have text at defined locations. Perhaps keep the simple table in the protocol chapter.



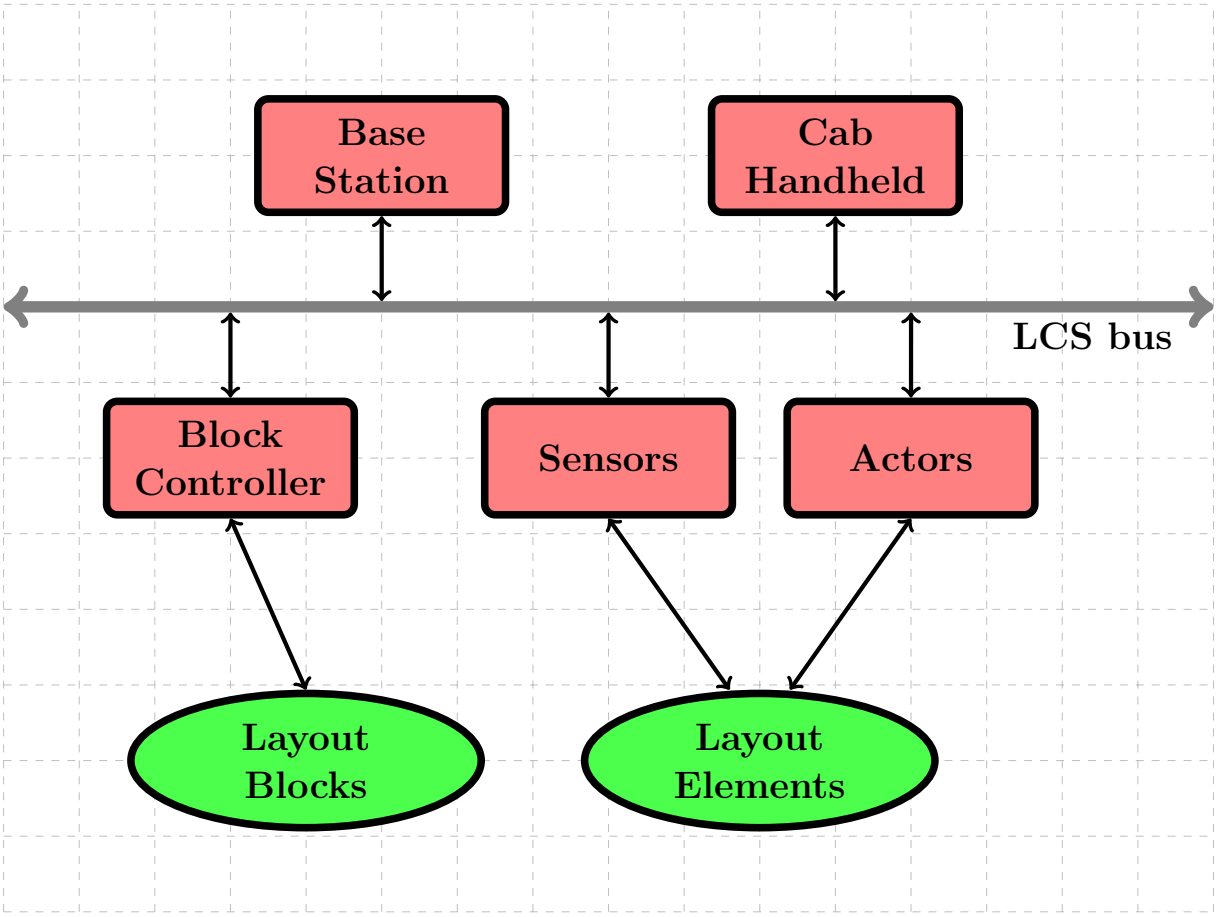
A.4 Split rectangle

We would need the split rectangle for the runtime area maps....

Hugo
Berta
Carla

A.5 Using tikzstyle

Another test with tikzstyle. It still is a lot of work to even make simple pictures look nice :-)


Base Station
Cab Handheld
LCS bus
Block Controller
Sensors
Actors
Layout Blocks
Layout Elements