Modernizing the FASTBUS readout system of the CERN NA49/NA61 experiment ToF detector using Raspberry Pi

Gábor Csaba Lucza, email: luczgab03@gmail.com [1]
Tibor Benke, email: benketib@student.elte.hu [2]
Dominik Márk Lévai, email: levaidominik@student.elte.hu [2]
Márk Tamás Baross, email: baross.mark@kvk.uni-obuda.hu [1]

[1]Kandó Kálmán Faculty of Electrical Engineering, Óbuda University, Hungary, 1034 Budapest, Bécsi út 96/B
[2]Institute of Physics and Astronomy, Eötvös Loránd University, Hungary, 1117 Budapest, Pázmány Péter sétány 1/A

Abstract: The FASTBUS [1] crate is a powerful modular data transfer system which uses ECL (Emitter Coupled Logic) electrical standards to achieve fast parallel communications with a 32 bits wide data line. Although FASTBUS crates were largely replaced by modern serial solutions [2][3], they are still in widespread use in legacy systems such as in older particle detectors. In this study we present a modernized FASTBUS readout system designed in collaboration with the Institute of Physics and Astronomy at Eötvös Loránd University for use with the ToF (time of flight) detector[4] formerly used in the CERN NA49/NA61 experiments and now being recommissioned for educational and research purposes. Our goal was to design a low-cost system capable of reading out up to 744 data channels used in the detector. For these purposes we have decided to use a Raspberry Pi 3B microcomputer connected to the FASTBUS crate via a custom control card. The control card translates between ECL and TTL signals and extends the number of general purpose I/Os of the Raspberry Pi, so it can use almost all FASTBUS operations.

Keywords: FASTBUS, readout system.

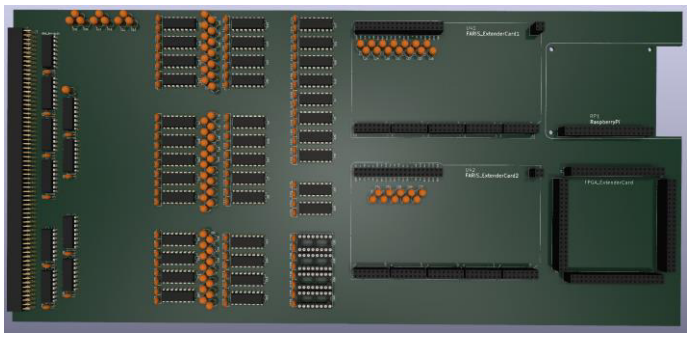

Figure 1: The main board of the FASTBUS control card. There are pins to potentially replace the Raspberry with an FPGA in the future.

1 Introduction

In 2020 ELTE acquired two older particle detector walls previously used in CERN's NA49/NA61 experiments, which were previously scheduled for disposal. The detector walls also known as "Buda Walls" were built around 1996 by KFKI in collaboration with ATOMKI. The walls were used in a ToF (time of flight) setup to identify the particles created in the experiments.

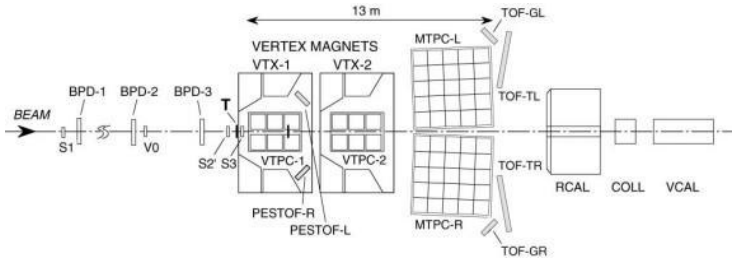

Figure 2: The set-up of the NA49 experiment. The Buda walls are TOF-GL and TOF-GR in the picture.

The walls in total have 372 analog channels where the particles flying through create electrical pulses. We can measure when these pulses are created with an estimated time resolution of 85 ps, and we can also measure their amplitude which can be used to calculate the energy of the particles. For these measurements we use LeCroy 1885F ADC and 1875A TDC FASTBUS cards. The need to use these

instruments in a modern way, and the lack of viable commercially available solutions are the main motivation of creating this new readout system.

1.1 The legacy system

The legacy readout system used a FASTBUS-VME bridge. On the FASTBUS side there was an F68B7 FASTBUS master and an FADAP cable segment connector which was used to connect more crates together and to connect the crates to an CES LDA9212 on the VME side controlled by a VME/VSB controller. The controller used the FVSBI interface to communicate with the FASTBUS master, sending it commands to execute and receive data back from it. Then a VME to USB bridge was used to communicate with the outside world.

1.2 The goals of our new system

Our goal was to eliminate this dependence on a VME/VSB environment (or any other "glue bus" systems for that matter), and to make using FASTBUS more straightforward, manageable and more easily integrable to new systems. For that reason we propose a novel approach of reading out FASTBUS instruments directly. This new DAQ (data acquisition) system which consists of a FASTBUS master card which houses the needed electronics to translate between the ECL logic levels used in the bus and the internal TTL/CMOS logic levels of the master. Furthermore this master card has a microcomputer with an I/O extender or an FPGA to run the DAQ program and to emulate the FASTBUS functions, managing the bus real time and storing data for future analyses. This can be easily accessed through a network by an external computer where the data can be stored analyzed.

2 FASTBUS

Historically, FASTBUS was created in response to the limitation of CAMAC (Computer Automated Measurement and Control)\cite{CAMAC}, using large 32 bit data lines, and ECL logic to speed up the bus itself. It also standardized multicrate and MASTER-MASTER communications. But with the rise of high speed serial protocols and developments in CMOS technologies a large, power hungry parallel bus wasn't feasible anymore. So it was superseded by serial bus solutions like VME and its later versions.FASTBUS at its core is a MASTER-SLAVE system. At any given time in a crate segment there can only be one MASTER and it has control over the whole segment. If there are multiple MASTERS they take turns in controlling the bus, with strict conditions not to halt the bus. For our use case we will only use one master per crate segment to

eliminate unwanted complexity. A normal readout cycle is initiated by the MASTER and it grants bus access to the SLAVEs one at the time, reading out their content. The means of communications are laid out in the IEEE 960-1986 (FASTBUS) standard, and in the data sheets of the given SLAVE.

A FASTBUS crate has a backplane with connectors for the inserted cards. This PCB (printed circuit board) is what enables the communication and also connects to the power supply. The backplane is what the standard calls a crate segment. Furthermore a crate also has an ancillary logic unit that has hardwired bus access and manages the MASTER-MASTER communications, addressing, and the Run/Halt logic. In addition, the crate has a side bus called "Auxiliary". This segment is isolated from the main bus, providing user or manufacturer defined purposes. Our MASTER will not use this option. The precise electrical workings of the backplane and the functions of the logic unit are also outlined in the FASTBUS IEEE standard.

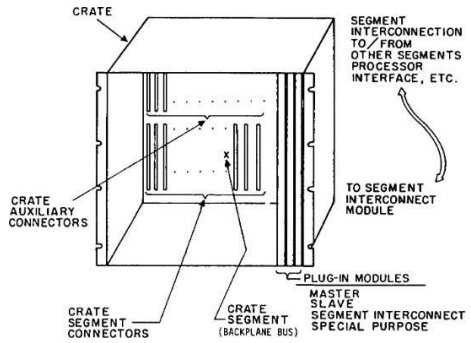

Figure 3: The structure of a FASTBUS crate, its backplane.

2.1 Topology

The FASTBUS standard calls for at least one MASTER per crate segment, but these MASTERs can't communicate with the outside world directly. For that the segment interconnects (SI) can be used to connect the bus to a processor interface (PI) which in turn can be connected to any desired system that has host processors. In the legacy systems PI was the FVSBI hardware and the host processor was in the VME environment. Our new readout system strictly speaking violates the FASTBUS standard in the regard that it integrates the PI and the host processor into the crate MASTER, simultaneously giving this "new MASTER" module communication, storage, and computing capabilities. This approach has the drawback of eliminating the multi-MASTER and multicrate capabilities of FASTBUS. But these use-cases aren't very common, moreover even a Raspberry

PI 3 compute module (which we are using) has more than enough computing capacity to handle any task that required a multi-MASTER setup. And for multicrate operations our new MASTER solution can feasibly be installed in every crate separately due to its low cost. We believe this solution greatly reduces complexity and eliminates potential error sources.

3 System Design

The new MASTER module will have 3 parts: The computer, the general purpose IO which the computer can control, and the ECL-TTL and TTL-ECL converters to translate between the BUS signals and the GPIOs of the computer.

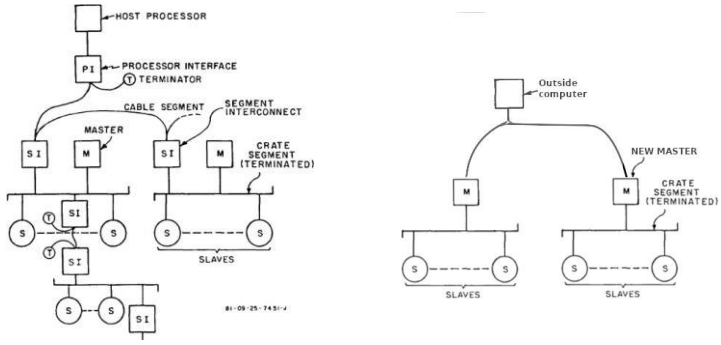

Figure 4: The original and modified topology

3.1 Raspberry Pi

We chose the Raspberry Pi platform for the main computer in the MASTER module (to be precise the PI 3B+, but the system will work with any newer and more powerful PI compute modules as well). The main motives behind this choice are price, availability and - most importantly - ease of use. The PI platform is very well documented and has its own minimalist Linux system which makes development and system integration fast and easy.

The PI 3B+ has a 64bit ARM Cortex-A53 CPU running at 1.4 Ghz, with integrated USB ISP I2C and UART controllers. It has no problem emulating any and all FASTBUS protocols that were originally running on the Motorola CPUs.

Figure 5: The Raspberry PI 3B+ compute module.[6]

The only problem with this is that the integrated GPIO controller of the PI doesn't provide enough IO for our needs. But this problem can easily be remedied by using one of the serial communication ports of the Raspberry to connect it to external IO controllers. In our system we will use I2C because of the large number of IO controller ICs. In the current hardware revision at the time of writing this I2C bus is the main bottleneck of the system; this bus is only 400 KHz, for our use case it will not cause any serious dead time but in the future with new hardware revisions we will explore potential solutions to greatly speed up this internal communication.

3.2 The IO expander

The main PCB of our master is a modular design, so if any of the problems mentioned above become too great these modules can be swapped out. The IO expander modules of this revision consist of five MCP23017 ICs.

3.2.1 MCP23017

The MCP23017 [7] is a register based I/O extender. It has 16 pins that can be configured either inputs or outputs, which is implemented by serial data transmission. I2C is a two-wire serial communication protocol that uses bidirectional SDA (data) and SCK (clock) lines to enable data transmission between two or more external peripheral devices and the microcomputer. The speed of the I2C line can be configured in the software, allowing it to be adjusted to the needs of the system. In our solution, we will use the "Fast-Mode" which performs at a 400kbit/s rate, but this device also supports 1.7Mbit/s speed. The SDA and SCL are defined to be high in idle state, changing it initializes the communication process. The process begins with a particular START condition, in which case the SDA is pulled low, but the SCL remains high. Next, the SCL will begin clock generation, and SDA will change always when SCL is low, except START and STOP conditions. According to the standard, after each byte

is sent, the peripheral device pulls the "9th bit" low, which is the ACK (Acknowledged) bit. This will confirm the data transfer success.

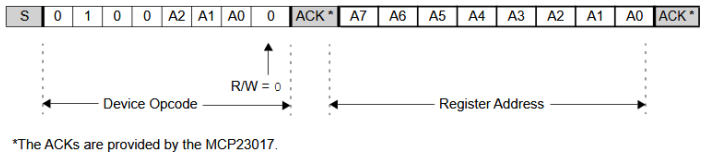

Figure 6: MCP23017 addressing by I2C protocol

In this case, the first 4 bits are predefined by the IC, so they can't be changed. The next 3 bits are defined addressing bits, that determine which integrated circuits we are talking to. The last bit is a R/W (read/write) bit, which can be configured in the software. After these 8 bits are recognized, the SLAVE pulls down the ACK bit to acknowledge the address, indicating other devices not to listen to the data-line. The next byte will be the data sent by the controller or the MCP23017. Data transmission continues until a STOP condition appears, indicating the end of communication. The device includes two built-in 8-bit registers (PORTA and PORTB) whose input or output configuration can be set using IODIRA/B registers. In addition, 11 register pairs provide functions and settings required for the device's operation, and navigation between them can be done in two modes. These are "Byte mode", which disables the default incrementation of the "Address Pointer" and "Sequential mode", which enables it. In our software design, we will use the special byte mode, which enables us to step between particular register pairs (for example GPIOA and GPIOB). Because of this setting, our addressing does not waste unnecessary resources, allowing the system speed to be optimized. Based on the speed of the Raspberry Pi's I2C output and the number of bits sent, we can calculate the maximum speed of the parallel signals appearing on the bus.In our case, we will use the "special byte mode" so that we do not need to send the target register address with each bit sequence. Thus, the total number of transmitted bits will consist only of the slave addressing and the data sequence.

$$\text{Number of bits} = \text{address byte} + \text{register A byte} + \text{register B byte} = 9 + 9 + 9 \approx 30$$

(with ACK)

$$f_{bus} = \frac{f_{RP}}{\text{number of bits}} = \frac{400\ \text{kHz}}{\approx 30} = 13.33\ \text{kHz}$$

This means that the IC will increase the number of I/Os, but at the cost of reducing the maximum speed of the bus to 13.33 kHz. The output frequency of the I/O is sufficient translation between ECL and TTL.

The FASTBUS operates on ECL levels while the controller uses TTL logic. Due the significant difference between the two standards, direct connection is not possible, so level shifters need to be inserted between the two circuits.

3.2.2 Emitter Coupled Logic

ECL [8] was considered the fastest logic of its time, used in applications that required short switching times. This logic system is based on a differential transistor pair. One of the transistors always lets through current, which is why this logic family is often referred to as the current-steered logic family. The speed of the switching is due the fact that the transistors do not enter the saturation territory, instead operating in the active region. As a result, the voltage difference between collector-emitter is low, so the parasitic capacitance will discharge faster due the lower charge voltage. This means that the switching time between high and low is a few nanoseconds, but because of the current-steering, it has a large power consumption. With the development of the CMOS standard, ECL lost much of its popularity, as similar speeds became achievable without the disadvantage of high power consumption.

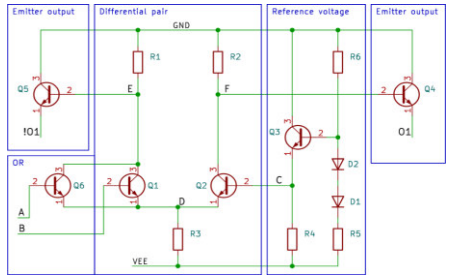

Figure 7: ECL OR/NOR gate

The ECL family also includes special types developed in later generations. The classic type, MECL (Motorola ECL), used 0 V and -5.2 V as supply voltages for negative logic outputs. Later, a TTL-compatible variant called PECL (Positive ECL) was introduced, which used a 5 V supply and had positive logic output levels. Furthermore, the positive family also included a lower-voltage version called LVPECL (Low Voltage Positive ECL) for outputs at reduced positive voltages.

3.2.3 Transistor-transistor logic

TTL [9] (Transistor-Transistor Logic) is a logic family built on bipolar transistors, which implements both logical operations and output driving using transistors. The technology is an improved version of DTL (Diode-Transistor Logic), and due to its simplicity and speed, it quickly became popular in industry and among hobbyists. During circuit operation, the logic levels are typically represented by discrete values of 0 V (low) and 5 V (high). Its switching time is on the order of tens of nanoseconds, which is moderately fast among logic families. But in many applications, this timing is entirely adequate. The switching speed is largely determined by the "totem pole" configuration used for output driving.

The TTL logic family has several improved variants, each targeting the enhancement of a specific property. These types differ from each other in terms of speed, power consumption, and noise immu- nity, and thus each optimized for particular application area. For example low-power TTL (TTL-L) has a few milliwatts of consumption, but this result in the decreased switching speed. Also notable is the Schottky TTL family, whose use of Schottky transistors allows switching speeds (as low as 3 ns) comparable to those of ECL. Additionally, we can mention the classic form of TTL logic, Standard TTL, which uses NAND gates to construct logical functions.

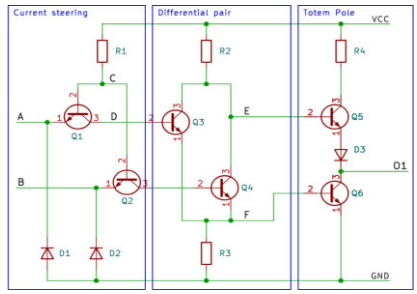

Figure 8: TTL NOR gate

3.2.4 10124

This type of integrated circuit [10] is a 4 channel device capable of unidirectional translation between TTL and ECL logic families, with negligible propagation delay. Since it consist of ECL, the power consumption of these ICs is high, which will be important in the design of the power rail. It is important to note that this device does not have a tri-state mode.

This particular device was selected because the existing FASTBUS cards already use the same type, thereby eliminating the risk of compatibility issues. Additionally, the decision was influenced by commercial availability of this type and the ease of its practical implementation.

3.2.5 10125

The 10125 [11] is the counterpart to the 10124, meaning this device is capable of translating ECL differential signals into TTL logic levels. Since it consists of ECL logic it has a high current consumption, therefore it has a high power dissipation. Similary to the 10124, it does not have a third, high impedance state and thus the output is ether high or low.

To ensure reliable operation both inputs need to be connected properly. In a single wire application the VBB reference (typically -1.3V) voltage should be properly connected to the unused input leg. For power supply the MECL logic requires -5.2V and 5V with a GND. These voltages are available from the FASTBUS backplane.

4 New architecture

Since the planned module would form a large unit that would complicate future upgrades, the system therefore consists of two separate modules. The first part is the main board, which contains the mentioned translators, the second is the I/O extension module. This modular approach ensures that if the Raspberry Pi provides insufficient performance, the system can be upgraded or replaced without any obstacles.

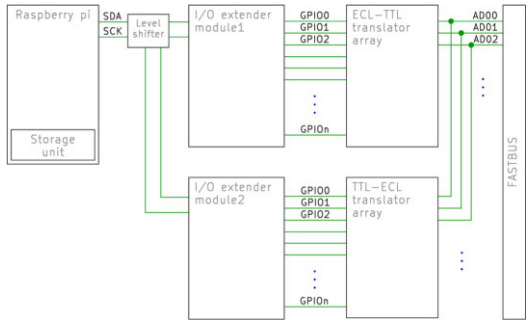

Figure 9: Logic diagram of the new system

The required number of expanders can be determined from the number of data and control lines to be managed. The FASTBUS has 42 control lines and 32 data lines, meaning a total of 74 lines are required for full bus control. Since the expanders ICs have 16 channels each, 5 units would suffice in an ideal case. However, because of the translators' two-state property, outputs can't be connected together, meaning that 10 unit will be needed for reliability. This quantity also includes spare, currently unused outputs reserved for future expansion. Since the level translator supports only unidirectional communication, one group of the expanders will function as inputs and the others as outputs. With this connection scheme, once the system is assembled

the modules will be fully capable of bidirectional communication between FASTBUS and Raspberry Pi.

4.1 I/O extender modules

A key design consideration was the TTL side drive of the level translators which require 0V and 5V levels. The MC23017 is capable of operating with 5V drive; however, the main problem came from the Raspberry Pi, whose SDA and SCL high level outputs are only 3.3V. Therefore a simple bidirectional FET level shifter was inserted between the I/O extender and the Raspberry Pi. The IRLZ44N MOSFET was selected primarily because is was already available in stock, and secondarily because of its fast switching speed. Since the number of required devices does not reach the maximum addressable limit of 10, the RESET pin was utilized. This trick effectively allows the use of multiple devices with identical hardware addresses on the same bus, thereby increasing the total number of usable extenders beyond its normal limit.

Due to the current draw of the ICs during switching, the supply rails can experience voltage drops, which is a problem for signal integrity. To prevent this, decoupling capacitors are placed close to each IC. Thanks to their charge-storage capability, these capacitors suppress switching transients. For stable operation, a combination of 100nF and 1uF ceramic capacitors is used in parallel at each chip.

4.2 The main PCB

The main PCB has the crucial job to connect to house the compute and IO expander modules, further more this is where the ECL-TTL and TTL-ECL converts are also placed, and of course this is where the FASTBUS connector is mounted.

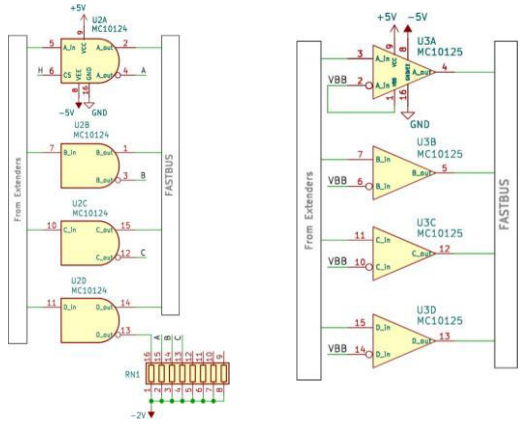

Figure 11: Main PCB schematic

To enable the MC10124, its CS (Common Strobe) is permanently tied to TTL high level. Its non-inverted (true) outputs are directly connected to the corresponding FASTBUS pin, which are internally terminated on the backplane. The unused ECL outputs must be terminated according to the datasheet, which is accomplished through a DIP-16 100Ω resistor network. The inputs of the MC10125 are directly connected to the FASTBUS pin, although the two IC types share the same bus line and are therefore not physically isolated. Muting the I/O expander boards prevents them form driving the translators, so they cannot interfere with unidirectional data flow. A reference voltage is also provided at the 10125, this voltage appears on V_{BB} pin as soon as power is applied.

The Raspberry Pi connection is made via a 2x20 pin header. All signals to the expander boards are routed through this header. The Pi and the entire system supplied by the FASTBUS power rail. The schematic also includes a custom FPGA socket which can be used for future extensions or upgrades.

5 Future plans

In the future, we intend to design and manufacture the hardware - that is, the PCB -, and also implement the required signal chain in a software environment. We would like to assess the limit values of the structure we create, and based on this, perform optimization of the system. We have also considered using an FPGA (Field Programmable Gate Array) instead of the I/O expanders that determine the speed limitations, since it is compatible with our existing Raspberry-based controller. Within the unit built this way, we aim to establish MASTER–MASTER communication, where the Raspberry issues commands to the FPGA, and the FPGA reads and stores the data.

6 Summary

In the above study we have designed a Raspberry-based FASTBUS readout system, to be used with a reactivated particle detector at ELTE. System control is performed by a Raspberry Pi 3B+, whose computational performance is more than sufficient. However, the Pi uses 3.3V TTL logic levels and therefore cannot directly interface with FASTBUS ECL signals. To solve this incompatibility, a level shifter and dedicated unidirectional level translators are used. Since full FASTBUS operation required control of 74 signals (32 data + 42 control lines) and the Raspberry Pi has far fewer GPIOs availible, I/O expanders were used. Each MCP23017 extender provides two 8-bit registers (16 I/O lines in total), and are controlled via I2C serial bus. The main drawback of this approach is the serialization of a naturally parallel bus, which reduces the overall speed of the bus. Therefore special configurations are used for speed optimization.

The I/Os of the MCP23017 expanders are directly connected to the inputs of the translators, thereby establishing reliable bidirectional communication between the modern compute module and the legacy FASTBUS.

Special thanks

Our work was supported by the Department of Atomic Physics at ELTE who kindly let us use one of their laboratories for our research. The Buda Wall and its related components were made by our dear colleagues at KFKI and were donated to ELTE by the CERN collaboration. We give special thanks to Dr. Gábor Veres (ELTE) who is the project manager of the Buda Wall reactivation efforts, and also to Dr. József Molnár (ATOMKI) who offered us remarkable insight in the design of the readout system.

References

[1] *IEEE Standard for FASTBUS Modular High-Speed Data Acquisition and Control System*, 1985.

DOI: 10.1109/IEEESTD.1985.121456.

[2] *Ieee standard for a versatile backplane bus: Vmebus*, 1988. DOI: 10 . 1109 / IEEESTD . 1988 . 8957719.

[3] R. Dettmer, "The VXI bus-an open standard for modulator instrumentation," *IEE Review*, vol. 35, no. 9, pp. 327–330, 1989.

[4] G. Pálla et al., "The grid-geometry time-of-flight detector used in the NA49 experiment at the CERN-SPS," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 451, no. 2, pp. 406–413, 2000. DOI: 10.1016/S0168-9002(00)00318-1.

[5] *IEEE Standard Modular Instrumentation and Digital Interface System (CAMAC)*, 1975. DOI:

10.1109/IEEESTD.1975.7431915.

[6] *Raspberry pi 3b+*. [Online]. Available: https://www.raspberrypi.com/products/raspberry- pi-3-model-b-plus/.

[7] *Mcp23017, 16-bit i/o expander with serial interface*, pp. 1–40. [Online]. Available: https : //ww1 .microchip.com /downloads/aemDocuments /documents/APID/ProductDocuments/ DataSheets/MCP23017-Data-Sheet-DS20001952.pdf.

[8] *Ecl 10k and 100k logic families*, 1986, p. 720. [Online]. Available: https://bitsavers.org/ components / philips / _dataBooks / 1986 _ IC08 _

Philips _ ECL _ 10K _ and _ 100K _ Logic _ Families.pdf.

[9] *Fast ttl logic series*, 1986, p. 916. [Online]. Available: https://bitsavers.org/components/ philips/_dataBooks/1986_IC15_Philips_FAST_TTL_Logic.pdf.

[10] *Mc10124, quad ttl to ecl translator*, pp. 1–8. [Online]. Available: https : / / project - cms - rpc-endcap.web.cern.ch/rpc/Chambers/Components/Electronics/LVDS%20to %20NIM% 20module / MC10124P % 20pdf , %20MC10124P % 20Description , %20MC10124P % 20Datasheet,

%20MC10124P%20view%20_%20ALLDATASHEET%20____.pdf.

[11] *Mc10125, quad mecl to ttl translator*, pp. 1–8. [Online]. Available: https://www.farnell.com/ datasheets/103015.pdf