MAGNETIC CORE MEMORIES:
HOW TO CONSTRUCT ONE AND HOW TO SURVIVE AN OLD IBM DJB 373330 SMS CARD

Del Popolo Salvatore, Didomenico Nicola

Department of Electrical, Electronics and Computer Science (DIEEI)
University of Catania - Prof. Orazio Mirabella

E-mail: popolo@tin.it, nicola.didomenico@gmail.com

Abstract

Writing about magnetic core memories means coming back more than 50 years ago in the digital era and making an effort to survive a technology that represented in a concrete way, the possibility to store data in a nonvolatile manner. In the past century, around forties and fifties, scientists, technicians and engineers all over the world began to project and realize first examples of computers for military aims.

Figure 0: IBM 1440 DATA PROCESSING SYSTEM

One of the fundamental elements of a computer is the possibility to store, either for the functioning of the system itself, or for future elaborations and uses. Magnetic, ferrite core memories made this fundamental function and were the dominant technology among fifties and seventies, before being substituted by transistors first and integrated circuits after.

I. INTRODUCTION

In this short paper, pointing out functioning principles of magnetic core memories, starting from the work of two American researchers Ben North and Oliver Nash and only using open source software, we very briefly summarize how we built our 32 memory array and how succeeded in controlling it by an STMicroelectronics microcontroller.

At the end, we demonstrate that, a DJB 373330 SMS Card, owned by one of our professors and coming from an IBM mainframe of the past century, is still functioning.

II. MAGNETIC MEMORIES: FUNCTIONING PRINCIPLES

Far from being extremely reliable, magnetic core memory was an attractive techonology, as based on a very simple idea.

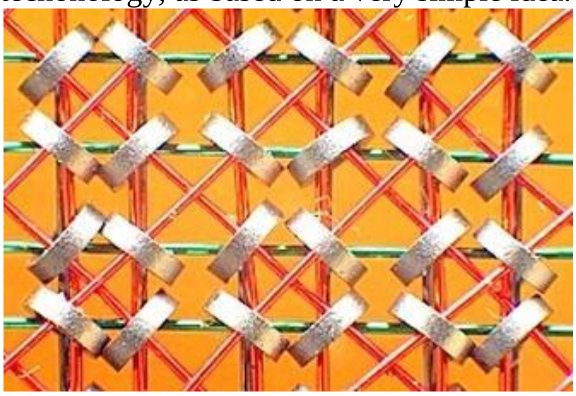

Figure 1: Magnetic Core Ferrite Memory (1940)

A core is a magnetic ring able to store just a bit, depending on the direction of its magnetization, how we can see from the graph in Figure 2.

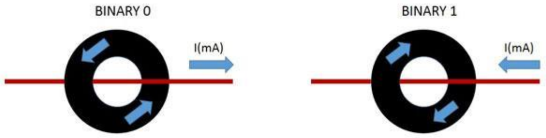

Figure 2: Logic States of Magnetic Core Memories

A magnetic core is a ferrite ring that can be permanently magnetized, either clockwise or anticlockwise, along its own axis. Hereby, a core can represent a bit of digital memory, imposing that the two states of magnetization are interpreted as 0 or 1, respectively, how we can see from the graph in Figure 3.

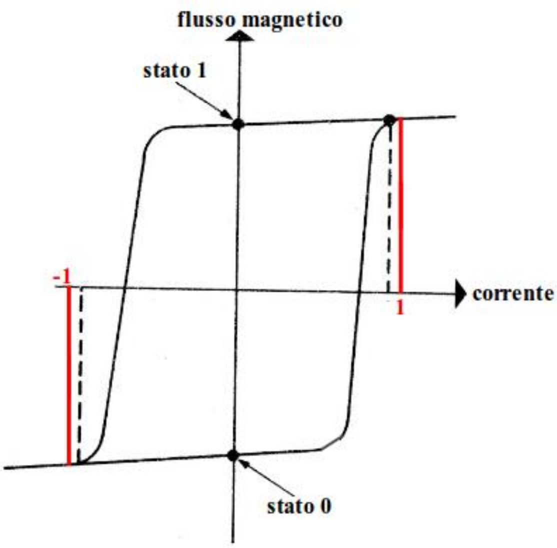

Figure 3: Direct and Opposite Current

The core need not to be powered to maintain its own value, realizing in this manner, a kind of nonvolatile memory as modern hard discs, but with an incomparably lower writing/reading speed.

As the technology evolved, core dimensions' decreased, passing from 2 mm in '50 to 0.4 mm in first years of '70 of past century. At the same time, access speed increased from 200 kHz to 1 MHz and assembling together hundreds of cores, built memories with more than 500.000 bits, how we can see from the graph in Figure 4.

Figure 4: Evolution of memories

The functioning principle of magnetic memories is based on a characteristic affecting all ferromagnetic elements. These can have two permanently states of magnetization. In the case of the ferrite ring, the two states of magnetization are identified by the two directions, clockwise and anticlockwise, around its circumference.

To set the magnetization core state's two conductive wires have to pass through it. A conductive wire generates a magnetic field and varying the intensity and the direction of the current that passes through it, it is possible to induce a change in the magnetization state of the core, creating what is defined as hysteresis cycle, illustrated in Figure 5.

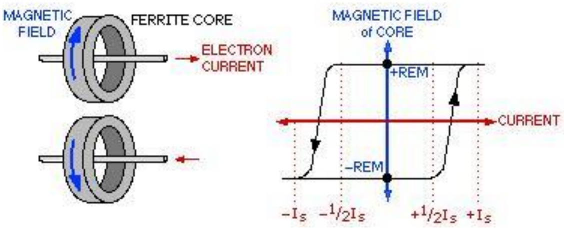

Figure 5: Ferrite Core Magnetization

Hysteresis cycle describes how changes the core magnetic field, as current varies in the wire. Points identified by ± REM represent the remaining magnetic field as no more current flows across the wire, they are the two magnetization states' that indicate the value 0 and one of the memory. Points identified by ± Is represent the required, current values to saturate the magnetic state of the core.

Organizing the cores, forming a two-dimension array, as in Figure 6, the only core affected by a change in the state is the one in which the two wires across each other and the two 1/2 currents sum themselves. Once the state changed, although removing the two 1/2 currents, magnetization core state does not change, storing a possible value. Remaining cores are not affected, as the 1/2 current that they receive, is not enough to induce a change in the direction of magnetization.

The orientation of cores versus currents is fundamental, as the two 1/2 currents must sum to each other to reach the necessary value to obtain the changing in the state. In fact, in this situation currents are defined coincident. To optimize driving lines in the control unit of the memory, it is also applied the mechanism of non-coincident currents, summarized in Figure 6. Finally, associating state of magnetization and logical value zero or one, is absolutely arbitrary.

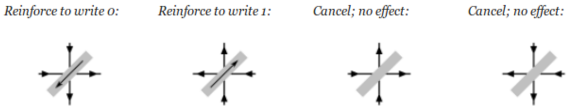

Figure 6: Coincident and Non-Coincident Currents

III. WRITING TO A MAGNETIC CORE MEMORY

We arbitrary impose that the two states of magnetization clockwise and anticlockwise represent values zero and one, respectively. With reference to Figure 7, let the two 1/2 currents flow, in the direct direction, in the two wires that identify the core we desire to write, until the direction of magnetization switches to clockwise. When that happens, the core will contain and maintain he value zero, even if no more current flows.

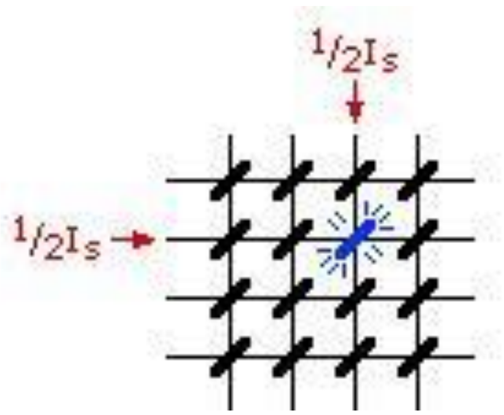

Figure 7: Driving Coincident Currents

To change the value of the core from 0 to 1, it is necessary to reverse the direction of magnetization. The two 1/2 currents have to flow in the opposite direction, until the state of magnetization reverses to anticlockwise. As explained before, the core will retain the value even if no more current flows, Figure 8 summarizes the entire process. Values of currents and time of impulse to obtain reversal of magnetization state are material and thickness dependent and can be found experimentally.

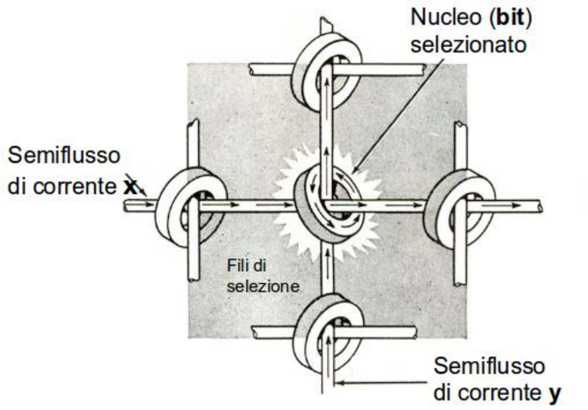

Figure 8: Writing 0 or 1 to a Magnetic Core Memory

IV. READING FROM A MAGNETIC CORE MEMORY

Reading from a magnetic core memory is a bit more difficult and it is necessary to introduce a new concept: a change in a magnetic field creates a current.
So, every time we reverse the magnetic field from clockwise to anticlockwise or vice versa using the two wires to identify the desired core, a little current is produced and can be revealed by a third wire, called the sensing, spread along the memory. See Figure 9.

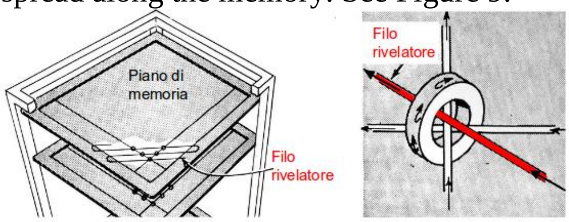

Figure 9: Sensing Wire

Keeping in mind the role of the sensing wire, to read a bit from a magnetic memory, we proceed as follows:

1. We write a 0. Whether the sensing reveals no current, no change in the magnetic field has happened, so, the core contained and will maintain 0.

2. Whether the sensing reveals a current, a change in the magnetic field has happened. So, the core contained 0, but now, that the magnetic field has reversed, it contains 1. Consequently, we lose the correct value 0 contained in the core and substituted it with 1, a wrong value. Now, it is necessary to write a 0 on the core, by reversing the magnetic field again.

This process is defined "destructive reading": in reading process, each time we write a value and the sensing reveals a change in state, we must regenerate the value contained in the core.

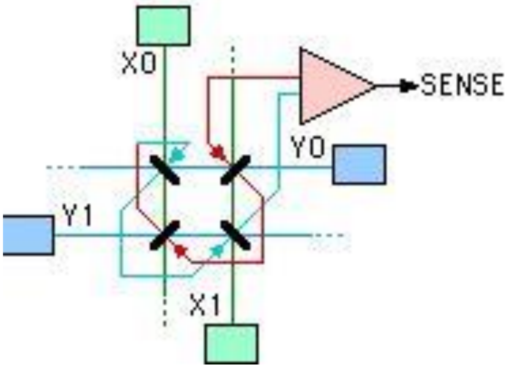

Figure 10: Sensing Wire in a 2*2 Memory Array

This simple schema can be further complicated, whether, instead of considering two-dimensional memories, we are interested in working with memories organized in core planes, one on top of each other, in order not to write a bit at a time, but a byte or a word at a time. In that case a fourth wire, a for each plane, the "inhibit" is inserted. At reading time, it is necessary to activate the inhibit pertaining to the plane containing the core we do not want to modify. Figures 11 and 12 summarize this concept.

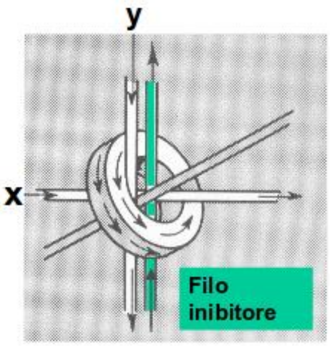

Figure 11: Inhibit Wire

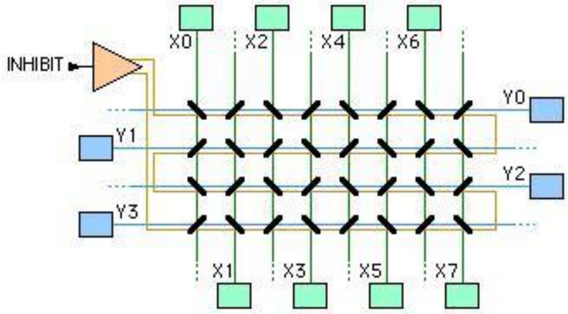

Figure 12: Inhibit Wire in a Bit Array

V. HARDWARE AND SOFTWARE TO CONTROL OUR MAGNETIC CORE MEMORY

Figure 13 summarizes the theoretical background we exposed so far and the required hardware to concretely build a functional magnetic core memory.

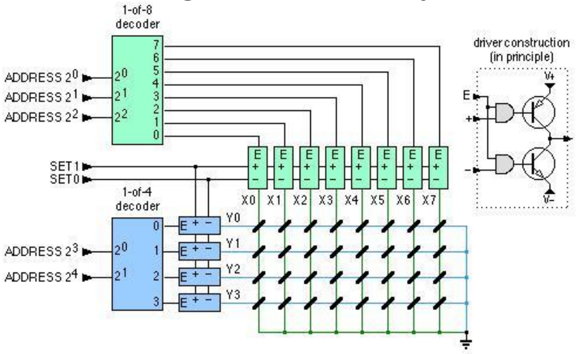

Figure 13: Driving in a 4*4 Bit Array

The circuitry receives X and Y coordinates of the selected core, together with the direction of the two currents and performs either reading or writing task.

As memories grew the simple, driving schema shown above began inappropriate because the required, increasing number of driving lines. To afford the problem, as shown in Figure 14, decoders were inserted. One decoder identifies the slice of the memory, while the other one determines the direction in which the currents have to flow. The theoretically, necessary 64 driving lines have been reduced to 16.

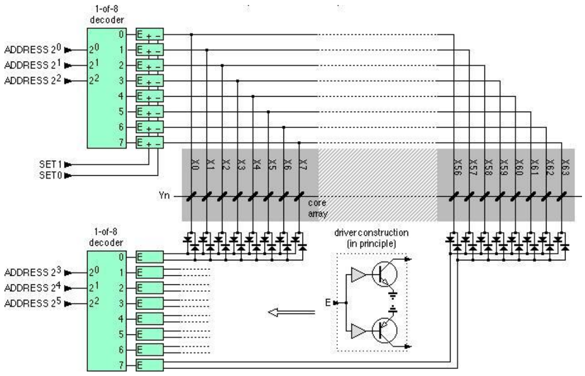

Figure 14: Reducing Driving Lines from 64 to 16

The method of non-coincident currents is used to further halve the number of driving lines. Considering a core identified by its two driving lines; of the four, possible combinations of the two currents, only two of them produce a change in the state of magnetization, those in which the two currents sum. They are defined coincident currents. The other two produce no effect, as being opposite currents, they delete each other.

They are defined non-coincident currents. Figure 6 summarizes these concepts. Considering now the two cores of Figure 15. We have still two driving lines, but one of them, describing two right angles, goes through one of the two cores in the opposite direction.

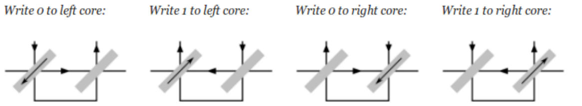

Figure 15: 4 Possible states of currents' Magnetization

Considering again all the four, possible combinations. We notice that all four states become valid, two for each core. It is like whether the array was divided in two slices and each core of the left side driven by coincident currents, has an homologous in the right slice driven by non-coincident currents, utilizing though the same two driving lines.

Starting from the project of Ben North and Oliver Nash, we built our magnetic core memory. After soldering components, one by one and many tests on
Arduino, uploading the firmware written by the two American researchers, we obtained the shields shown in Figure 16 and 17.

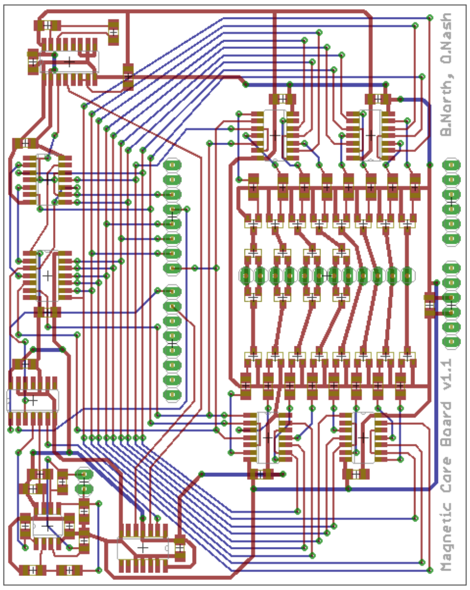

Figure 16: Eagle Drive Shield Layout

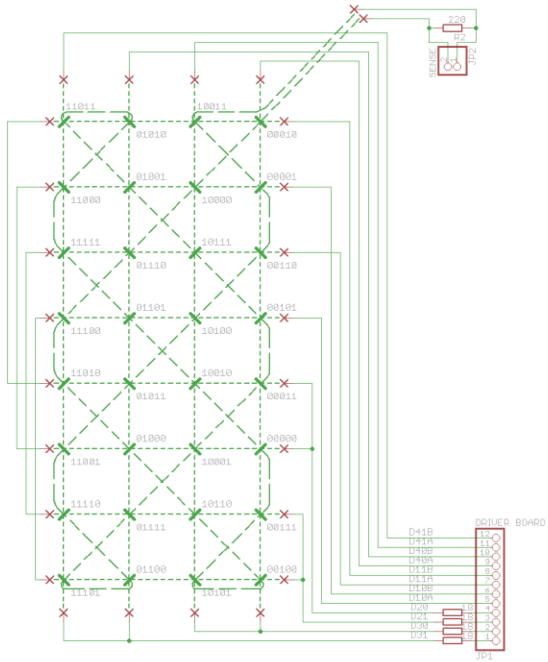
Figure 17: Eagle Core Shield Layout

Then, we went a step further. We ported the firmware from Arduino Uno to STMicroelectronics' STM32 and connected our shields to a Nucleo F411RE. Figures 18 and 19 below, show the assembled hardware: Nucleo, Core Shield and Drive Shield.

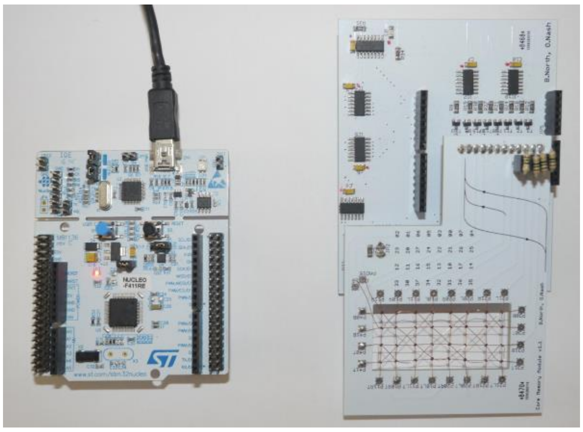

Figure 18: Nucleo F411RE, Drive and Core Shield (View From Above)

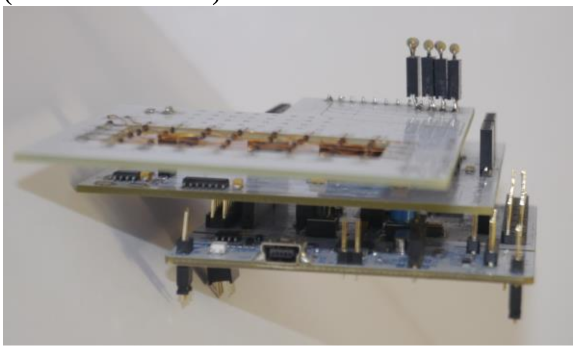

Figure 19: Nucleo F411RE, Drive and Core Shield (Front View)

The software we wrote is similar to that written by North and Nash, apart from tracing, logging and current calibrating functions that we did not implement. Our development work was entirely done under the Linux operating system, using only open source software. We also wrote some, little templates to automate compiling, uploading and debugging the code for Nucleo F411RE.

In Figure 20 you can see a screenshot of the interactive menu, in which: t stands for 'testing all bits' array', r for 'reading a specific bit', R for 'reading the entire array', w for 'writing a specific bit' and W for 'writing the entire array'.

Single bits are specified by binary addresses from 0 to 31.

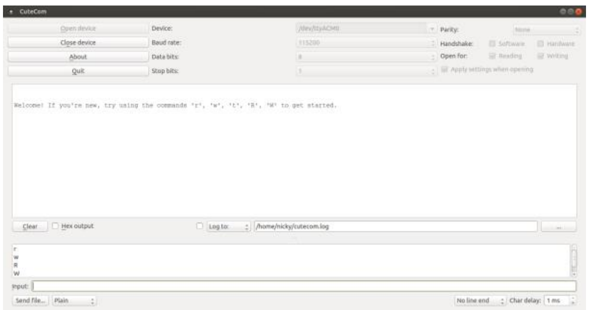

Figure 20: Screenshot of the Interactive Menu of the Firmware

We finally substituted the Core Shield with an IBM DJB 373330 SMS card, shown in Figure 21, owned by our professor.

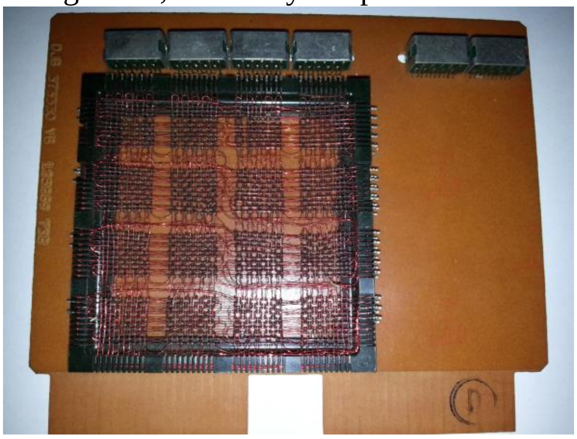

Figure 21: DJB 373330 Card

Unfortunately, although we contacted people all over the world, we did not find any documentation about electric schemes and circuitry of IBM DJB. So, using an oscilloscope and an electronic microscope, we drew the CAD representation shown in Figures 22, 23 and 24.

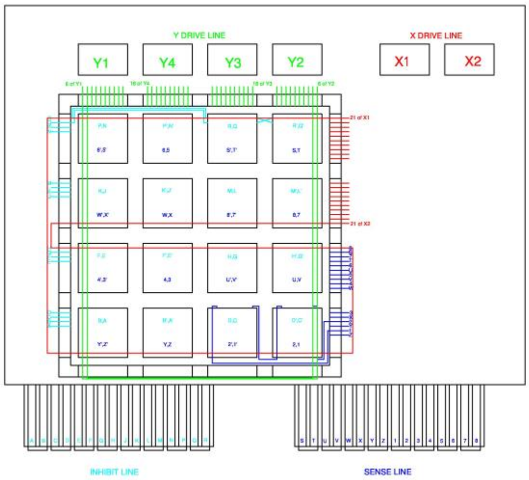

Figure 22: CAD DJB 373330 Schema 1 of 3

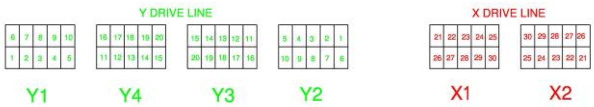

Figure 23: CAD DJB 373330 Schema 2 of 3

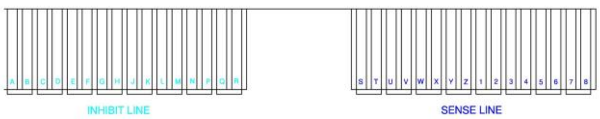

Figure 24: CAD DJB 373330 Schema 3 of 3

After a hard testing work, we succeeded in identifying the pins that addressed two of the DJB's 1600 cores, that could be driven by our Drive Shield. Figures 25, 26 and 27 show these connections together with a connecting board.

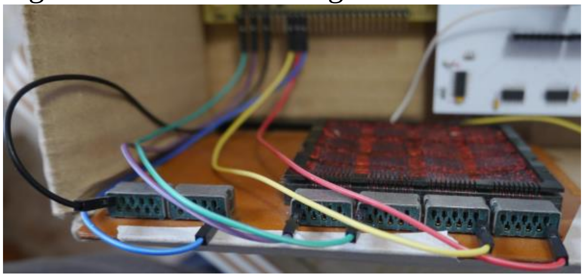

Figure 25: Connections for addresses 4 and 14 on the DJB Card

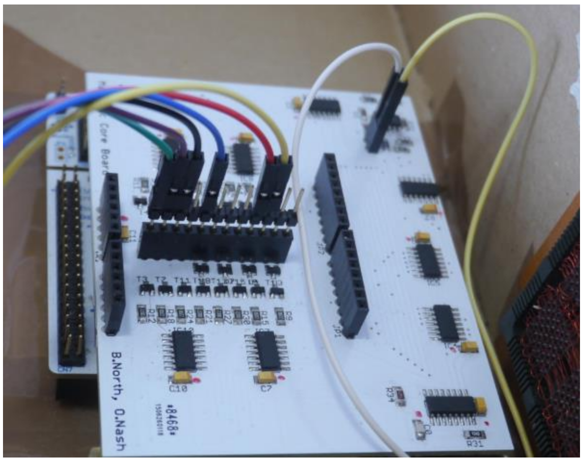

Figure 26: Connections for addresses 4 and 14 on the Drive Shield

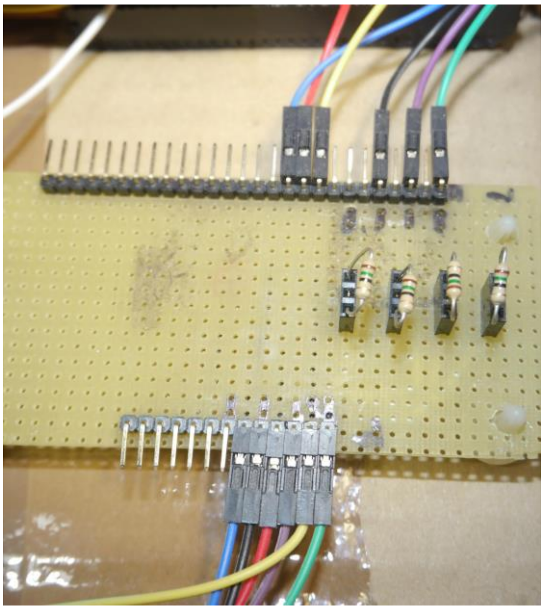

Figure 27: Connecting Board between DJB and Drive Shield

We could write and read two cores, demonstrating that, at least partially, the DJB card was still functioning. In Figure 28 you can see our final, assembled project, whereas, a screenshot of the firmware uploaded on the Nucleo is shown in Figure 29. It specifically refers to the two addresses, 4 and 14, identified on the DJB.

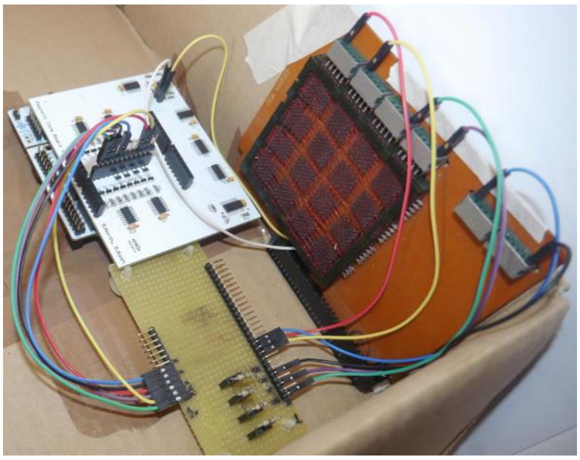

Figure 28: DJB Card's Entire, Driving System

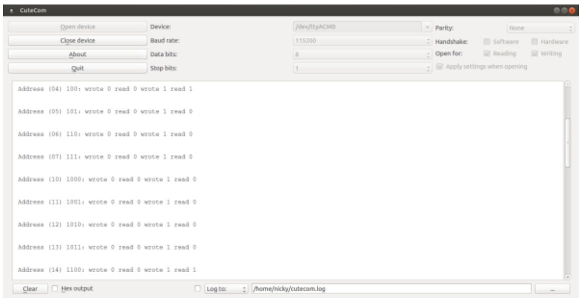

Figure 29: Screenshot of the Software controlling DJB Card

VI. CONCLUSION

Working with Magnetic Core Memories was an exciting experience. It is not easy to tackle with this sort of problems, especially nowadays, that this technology is no more used and find technical references is almost impossible. Moreover, owning an original IBM DJB 373330 SMS Array built more than 50 years ago is really a pleasure for future engineers. So, although the difficulties we encountered to develop our project and although we started from a very well done work of North and Nash, we very much thank our Professor Orazio Mirabella and our Tutor Engineer Antonio Raucea, who gave us the possibility to concretely experiment with a piece of technology that represented an important step in modern computer science.

We very much desire to thank all people from many countries who helped us to end our work and being coherent with our idea of knowledge sharing, at the addresses GitHub or Corememory Shield is freely available all the documentation and software we produced for our project. A video is available at Magnetic Core Memory, as well.

REFERENCES

[1] Geoffrey Brown “Discovering the STM32 Microcontroller”, 10 Aprile 2015.
[2] Joseph Yiu “The Definitive Guide to ARM Cortex M3 and Cortex M4 Processors”, Newnes, 2013.
[3] ARM Limited “Cortex-M4 technical reference manual”, 2010.
[4] Trevor Martin “The Insider’s Guide To The STM32 ARM Based Microcontroller”, Hitex, 2009.
[5] STMicroelectronics “Datasheet STM32F303xB STM32F303xC”, rev. 7, 2013.
[6] STMicroelectronics “STM32F3DISCOVERY peripheral firmware examples AN4157”, rev. 1, 2012.
[7] STMicroelectronics “Datasheet STM32F405xx STM32F407xx”, rev. 4, 2015.
[8] STMicroelectronics “STM32F4DISCOVERY peripheral firmware examples AN3983”, rev. 2, 2011.
[9] STMicroelectronics “Datasheet STM32F401xD STM32F401xE”, rev. 3, 2015.
[10] STMicroelectronics “STM32 Nucleo boards UM1724”, rev. 7, 2015.
[11] STMicroelectronics “Description of TM32F30xx/31xx Standard Peripheral Library UM1581”, rev. 1, 2012.
[12] STMicroelectronics “Description of STM32F4xx HAL drivers UM1725”, rev. 1, 2014.
[13] Free Software Foundation “Using the GNU Compiler Collection”, for gcc version 4.8.4 (GNU Tools for ARM Embedded Processors).
[14] Free Software Foundation “Debugging with GDB”, Tenth Edition, for GDB version 7.9.
[15] Free Software Foundation “GDB Quick Reference”, version 5.
[16] Free Software Foundation “GNU MAKE”, Version 4.1 September 2014.
[17] ARM Limited “ARM and Thumb-2 Instruction Set Quick Reference Card”, 2008.
[18] Free Software Foundation “OpenOCD User’s Guide”, for release 0.10.0-dev 5 October 2015.
[19] STLINK development team “Using STM32 discovery kits with open source tools”
[20] Brian W. Kernighan, Dennis M. Ritchie “Il linguaggio C. Principi di programmazione e manuale di riferimento”, Pearson 2004.
[21] Texas Instruments “The Integrated Circuits Catalog for Design Engineers”, First Edition January 1970
[22] DEC Equipment “PDP-8 Maintenance Manual”, 1966
[23] “Byte Magazine - Core Memories”, James R. Jones Number 11 July 1976

[S1] www.st.com/web/en/catalog/tools/FM116/SC959/SS1532/PF254044
[S2] http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF252419
[S3] http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/LN1847/PF260000
[S4] http://www.arm.com/products/processors/cortex-m/cortex-m4-processor.php
[S5] http://www-micrel.deis.unibo.it/LABARCH/2015/materiale.html
[S6] http://homepage.cem.itesm.mx/carbajal/MR2010/slides.htm
[S7] http://www.openstm32.org
[S8] https://github.com/gnunicky
[S9] http://jeremyherbert.net/get/stm32f4_getting_started
[S10] https://launchpad.net/gcc-arm-embedded
[S11] https://gcc.gnu.org/
[S12] https://sourceware.org/gdb/download/onlinedocs/
[S13] http://www.gnu.org/software/binutils/
[S14] http://www.gnu.org/software/make/manual/
[S15] http://www.doc.ironwoodlabs.com/gnuarm/gnuarm-distribution/web/www.gnuarm.com/
[S16] http://beej.us/guide/bggdb/
[S17] www.wolinlabs.com/blog/linux.stm32.discovery.gcc.html
[S18] www.openocd.org
[S19] https://github.com/texane/stlink.git
[S20] http://www.emcu.it/STM32.html
[S21] http://www.carminenoviello.com/category/elettronica-it-it/
[S22] https://github.com/RIOT-OS/RIOT/wiki/RIOT-Platforms
[S23] http://mdda.net/oss-blog/
[S24] http://web.stanford.edu/class/cs107/resources.html
[S25] http://www.pixelbeat.org
[S26] http://www.corememoryshield.com/report.html
[S27] http://www.cs.ubc.ca/~hilpert/e/corememe/index.html
[S28] http://ed-thelen.org/comp-hist/Byte/76jul.html
[S29] http://nid.dimi.uniud.it/history/history.html
[S30] http://www.retrocomputing.net/eventi/toscana/pi/pisa/20090518/corsobonfanti/corso.htm
[S31] https://sites.google.com/site/wayneholder/one-bit-ferrite-core-memory
[S32] http://www.ffellico.com
[S33] http://members.iinet.net.au/~daveb/
[S34] http://tinyurl.com/pdp8core
[S35] http://www.qrp.gr/microwave/
[S36] http://www.aholme.co.uk/
[S37] http://other-1.webs.com/
[S38] http://www.righto.com/2015_03_01_archive.html
[S39] https://www.flickr.com/photos/132738729@N05/sets/72157653027710200