Laboratory Design Projects for a Freshman Digital Electronics Course

Gerard N. Foster
Purdue University, School of Technology, Kokomo, Indiana

Abstract

This paper discusses a set of laboratory projects that the author created for a second semester freshman digital electronics course. The following projects have been developed:

- Stepper motor feedback control to allow positioning of motor shaft.
- State machine with PLD to setup smart dot-matrix display.
- Shift register circuit with communication to microcontroller SCI.
- Digital play/record circuit
    - Simple function generator capture and replay
    - Voice recorder and playback
    - Digital organ

Block diagrams are used in presenting the design of systems. Students are divided into groups. They are asked to develop overall concepts and then to set up design procedures for individual blocks. Students are asked to work as a group and then to implement testable sections as individuals or in small groups. Students gain knowledge of memory-mapped systems, microcontroller serial communication interfaces, "smart" displays and PLDs as digital control blocks. Good group dynamics and the ability to switch to a design mode of thinking are important in the success of the implementations. By providing structure when needed, the instructor helps the students raise their level of competence.

Introduction

In the Electrical Engineering Technology curriculum at Purdue, most EET courses have two or three hour laboratories along with the regular classroom periods. Time and again, our students tell us that they like this commitment to hands-on learning. It is in "lab" that the students come to a better understanding of the technical material. It is through laboratory exercises that students develop their writing skills by writing laboratory reports. It is through laboratory exercises and project that students practice troubleshooting and design. It is there that they learn to work in groups to solve technical problems. It is also in lab where students can seize the complexity of a problem such as finally realizing that a filter or a different grounding scheme may solve a real world problem of noise.

Introductory digital electronics laboratory exercises can become limited and cookbook-like. This rote, by-the-numbers approach is not all bad to start the novice learning in a step-by-step fashion. However, there comes a time when introducing design elements into laboratory exercises can stretch the student to use and to more firmly establish previous learning. And, there comes a time when a laboratory exercise can become a bridge to future applications. The following laboratory exercises and projects were developed for freshman students in the last half of a

second course in digital electronics. The author developed these projects for the following reasons:

- To stimulate student interest by doing more challenging exercises.
- To cement in the students' knowledge of basic material by requiring that they use the concepts to solve real digital problems.
- To present problems that require thinking in terms of systems and using meta-design structures such as functional blocks and block diagrams.
- To introduce the students to complex devices such as "smart" displays.
- To require students to work on projects that involve memory devices and schemes for addressing and selecting memory chips.
- To work on hardware that is applicable to microcomputer architecture.

The following four projects are presented in order of their occurrence in our laboratories. Each project is independent of the others. If you wish, you may skip over the first project or two to get to one with more interest to you.

Project #1: Stepper motor feedback control

The stepper motor project involves developing a circuit that will cause the shaft of a stepper motor to turn to a position that is designated by a binary number that a user enters into a switch bank. The design is initiated by thinking of functional blocks. A block diagram is developed. One such diagram is shown below. In this design, the stepper motor shaft turns a number of steps either clock-wise or counter clock-wise until the output of a binary counter matches that of a binary number entered into the switches.

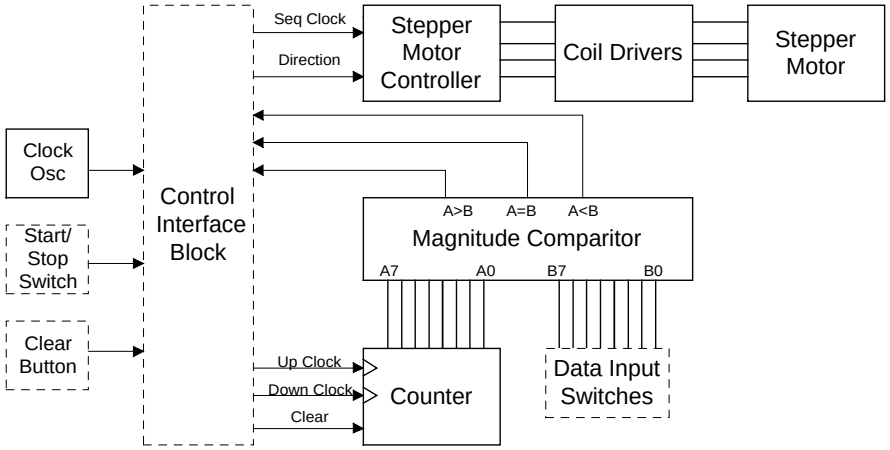


Fig. 1: Stepper Motor Position Controller

Using a digital magnitude comparator, the counter output and the switch value can be compared. The resulting comparator output affects the direction line into the stepper motor controller and the up/down clock input to the counter. Thus the motor is stepped in the right direction and the

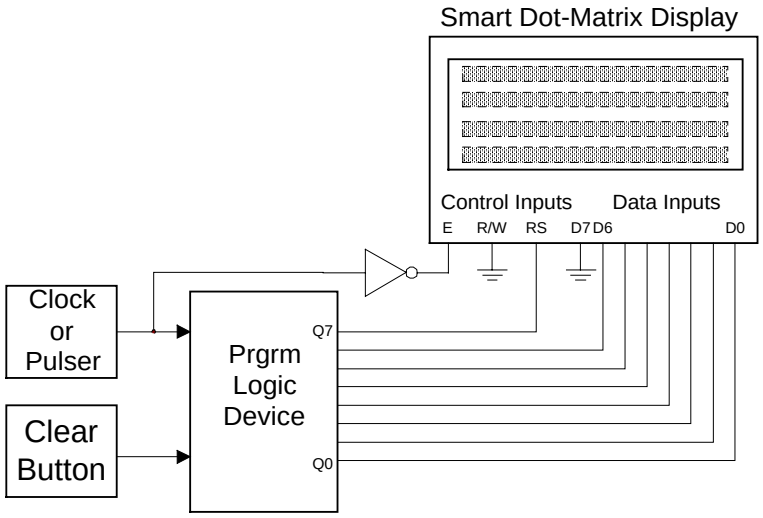
counter counts toward the switch value until the two values are equal at which time the correcting operation stops.

The students are grouped into design teams of 3 to 4 students. They are asked to come up with a design for the control interface block. This design may take place outside of class. The groups return with their designs. The instructor goes over control interface concepts again and the design is smoothed out. The class is again grouped into new teams to do the functional implementation. The functional groups handle the stepper motor section, the counter section, the magnitude comparator /input switch section and the control interface section. The counter and comparator sections can be combined and the control section can be divided into two parts.

This project usually takes two laboratory sessions with guidance from the instructor in class. The concepts that this exercise reinforces are the need for pull-up or pull-down resistors, determining what needs to be latched, knowing active levels of inputs and outputs, laying out inputs and outputs to a system, and visualizing design using block diagrams. In addition, grounding and power line layouts become important when integrating system boards.

Project # 2: State machine with PLD to setup smart dot-matrix display

The exercise with the smart dot-matrix display is perform by individual students. The display is a 4-line (2 split lines), 20-character Optrex device. Each student must write and program a PLD as a state machine to produce a sequence of binary values that initialize the display, write three initials, and continuously rotate the displayed characters in marquee fashion. The block diagram for this circuit is shown below.



Smart D0t-Matrix Display Controller

Fig. 2: Character-oriented dot-matrix display controller

The control inputs consist of a clock input E that latches the input data on the falling edge, a read/write input R/W that is defaulted to the write mode by grounding it, and the register select

line RS that establishes the data type (0 for instructions and 1 for data). To initialize the display, four data inputs are sent as instructions with the RS line low. The instruction values are as follows: 0x38 (Function set: 8 bits, 2 lines), 0x0E (Cursor move), 0x06 (Cursor shift right), 0x01 (Clear display). The hexadecimal numbers given in the previous sentence represent the values sent to the data input port (D0 – D7). Note that D7 is always zero for this exercise. Therefore, the D7 input is grounded. The GAL 16V8 PLD we program has 8 output lines. The Q0 through Q6 outputs are attached to D0 through D6. Q7 is connected to RS and provides the signal for the data type to be employed. In our scheme, the PLD's output is established on the rising edge of the clock and this value is clocked into the display on the falling edge of the clock. The next three values input to the display are entered as "data" data with RS brought high. In the data mode, the values entered into the data port are generally the ASCII values for the characters to be displayed. There are exceptions. Device specification sheet should be consulted. The student enters a sequence of ASCII values for the three initials. The restriction for this design is that no two initials can be the same. Finally the value 0x18 is continuously entered into the data port with the RS line low. This instruction causes the display to shift the characters one space to the left.

The PLD development software CUPL is used to yield a Jedec file. A sample source file is provided to the students along with a template file so that the students will not have to write so much tedious code and they will be less prone to be hopelessly stuck.

This laboratory exercise is relatively easy to implement. A test station is provided at which the students demonstrate results to the instruction. This laboratory shows the students that in "smart" devices data may be considered as instructions or as "data." The idea of a control bus and a data bus is also illustrated.

Project #3: Shift register circuit with communication to microcontroller SCI.

In the serial communication project, shift registers are used to implement an asynchronous serial transmitter. Figure 3 shows a block diagram of this circuit. The shift register block can consist of two 8-bit, or three 4-bit, parallel-in serial-out shift registers. In this circuit, eleven bits are automatically shifted to the output of the shift register when the start button is pressed. The number of bits shifted is determined by the decoding circuitry in the control interface. The Start button sets the shifter into the shift mode and starts the counter. At the end of the shift interval, the decoded value sets the shifter into the load mode, turns the counter clock off, and zeros the counter. In this design, the load mode is continuously active after the shift is complete. In the load mode, the value entered by the user into the switches is loaded into bit2 through bit9 of the shifter. The bit0 input is strapped to a high level. It represents the idle level of the transmitter. Bit1 is connected to ground. It represents the start bit in asynchronous serial communication devices. A separate load button could have been incorporated into the design, but that seemed unnecessary.

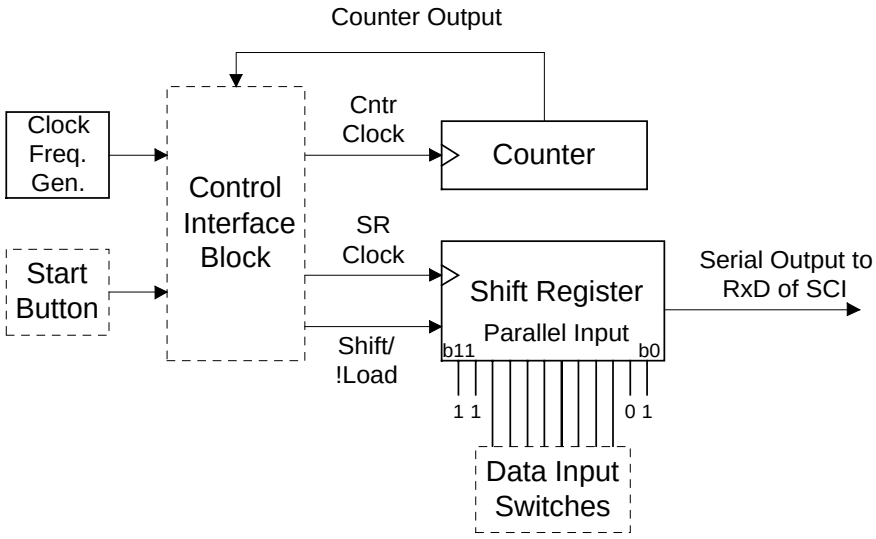


Fig. 3: Asynchronous serial transmitter to serial communication interface of microcomputer

The circuit is implemented and tested by a team of two or three students. It is interfaced to the serial communication interface (SCI) receive input of an HC11 microcontroller. The instructor provides code for the microcontroller with a baud rate of 9600. The clock input to the control interface is connected to a functional generator. The students enter ASCII code values into the data input switches. By pressing the start button, the serial transmission is initiated. A successful transfer of data results in the following message being printed to the PC monitor attached the the HC11EVB: ASCII value = xx ASCII chararacter = x, where x represents the results corresponding to the particular data entered by the student. The students can then increase and decrease the function generator clock frequency until the transmission is no longer reliable. The input value 0x41 for capital A is a good value to use for the frequency match test. A digital oscilloscope can be used to capture the serially transmitted data.

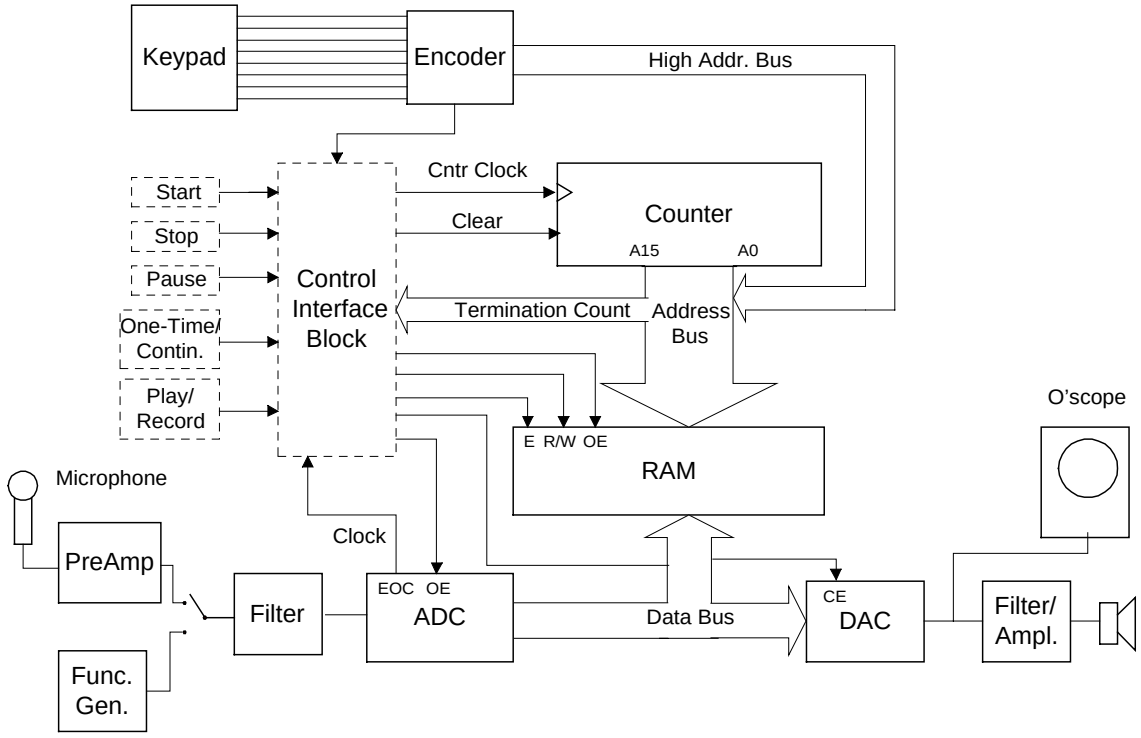
The students learn about shift registers and a shifting application with this project. In addition, the students are exposed to the microcomputer and they use ASCII code in a real situation. The asynchronous nature of the transmission is emphasized. With classroom help, this laboratory project can take only one laboratory period.

Project #4: Digital record/play circuit

By far, this is the most complicated of the projects presented here. It is the capstone project of the course. This project is the design and implementation of a circuit that will capture an audio frequency signal in a RAM and play the captured signal. To date, there are three variations, or stages, of this project:

- Simple function generator capture and replay
- Voice recorder and playback
- Electronic organ

Figure 4 gives a block diagram of this circuit with the electronic organ input included.



Block Diagram of Digital Play/Record Circuit (with Keypad Input)

Fig. 4: Digital play/record circuit

Both 8Kx8 and 32Kx8 (62C64 and 62C256) RAMs have been used for this project. The larger memory chip is preferred. The counter block implements a 16-bit counter whose output is used to scroll through the RAM addresses. The analog-to-digital converter is an ADC0804 8-bit device that uses an internal clock with frequency determined by an external resistor and capacitor network. The ADC is wired in the continuous running mode by connecting the end-of-conversion (EOC, also called !INTR) line to the start-of-conversion (SOC, also called !WR) line. An initial start boot to the ADC may be required to pull the SOC line low on power-up. The ADC clock is the clock for the system providing timing in the play mode as well as in the record mode. This frequency is approximately 8kHz. Two of the switches are SPDT and three of the switches are momentary switches. Their use is explained below.

In the record mode, a microphone or a function generator is selected as an analog input to the ADC. The Play/Record switch is placed on record. The One-Time/Continuous switch is placed on One-Time. When the Start button is pressed, the counter starts counting from zero up to the terminal count. The ADC's output enable (also called the RD, for read, line) line is activated. The RAM is set into the write mode and the converted signal samples are stored into the RAM memory as the counter address is incremented. If the memory space allocated for memory storage is a factor of 2 in length, the terminal count is very simple to decode. Let's say that we

have allocated 32K of RAM for the signal. This arrangement requires the lowest 15 bits (A0 to A14) from the counter to address the RAM. The highest bit (A15) of the counter can be feedback to the control block and used to terminate the counting. Adding a lamp that is on during the recording of the signal is useful when using the microphone as an input.

In the play mode, the Play/Record switch is placed on play. The position of the One-Time/Continuous switch depends on whether one wishes to play the stored signal once or to continuously repeat the playing of the signal. In this mode, the digital-to-analog converter (ADC558) is enabled. The output enable of the ADC becomes inactive, tri-stating the ADC data output. The RAM memory is put into the read mode taking control of the data bus. Upon pressing the Start button, the counter circuit works as before addressing the RAM. The data from the RAM are sampled by the DAC and converted to an analog signal that is displayed on an oscilloscope and directed to an audio speaker. The Pause button causes the counter to stop where it is. The Start button will cause the counter to start from the address of the pause. The Stop button stops the play and resets the counter address to zero. To simplify the design, the Pause and the Stop buttons could be eliminated.

After doing the project several times, the instructor introduced the concept of partitioning the RAM into 8 sectors in which a different note of the musical scale can be recorded. In this scheme, the keyboard of the "electronic organ" provides the highest addresses (A13 through A15) to the RAM. For convenience, we used 8 buttons from a 16-button keypad and a 74C922 keypad encoder for this circuit.

This project takes from 3 to 5 weeks depending on how much design is required and how much direction is given. It has been performed by groups of 3 to 8 students. The smaller groups do as well as the larger groups and generally are more active. Because the students are freshman, they may not have the analog background to design the analog input and output circuits.

There are plenty of concepts illustrated by this project. Among them are the ideas of tri-stated outputs, read and write modes, addressing memory. Indeed, the central concept of this project is how to use binary inputs to address memory and how the higher and lower address bits can be used to partition memory. In microcontroller/microcomputer courses, the acquisition of software skills can overwhelm the student to the point that hardware architecture is not firmly grasped. One of the goals of this project is to implement a system that requires memory addressing hardware. In the area of signal processing, the idea of sample rate and the Nyquist theorem can be introduced at a low level. Examples of filtering are seen. And, of course, how to design the control interface is the part that makes the operation of the circuit come alive.

Other Projects

These projects are the most recent ones developed by the author. In the interest of completeness, here are some other projects developed for this course. Years ago the capstone project was to design and implement a frequency counter and display. The author again used a block diagram approach to present the project. Beyond implementing and testing the basic circuit, the students were asked to add a feature of their own. Some students selected a different display device or added ranging features. This work proved insightful to the author in terms of team member

interaction. A more recent project was to create a user guided mobile robot using an EPROM. One project under consideration is the development of a system in hardware that would simulate a simple microprocessor circuit. Another idea is to build and use a microcomputer in a simple fashion.

Conclusion

This work is encouraging. Using the idea of block diagrams has been fruitful. Students enjoy the projects. There is a greater connection of the projects to larger systems and to microcomputer circuits. The exercises of the past had become stale. With these projects there is new life in the laboratory. But, the work is not done because an even greater tie with the microcomputer courses is desired. New projects are always under construction.

Bibliography

1. Capozzoli, T. K. & Foster, G. N. (1994). An exploratory study of group dynamics in engineering technology students. Proceedings of the 1994 Annual Conference of the American Society for Engineering, 983-986
2. Foster, G. N. (1993). Prescriptions for capstone design courses in electrical engineering technology. Proceedings of the 1993 Annual Conference of the American Society for Engineering Education, 178-181.
3. Foster, G. N. (1991). Team project in an advanced microprocessor course. Proceedings of the 1991 Annual Conference of the American Society for Engineering Education, 124-127.
4. Tocci, R.J.. & Widmer, N.S. (2001). Digital Systems: Principles and Applications, 8th ed.. Upper Saddle River, NY: Prentice Hall. Pp.881.
5. Wakerly, J. F. (2000). Digital Design: Principles & Practices. Upper Saddle River, NJ: Prentice Hall. Pp. 950.

GERARD N. FOSTER

Gerard (Jerry) Foster is an associate professor of electrical engineering technology at Purdue University, School of Technology at Kokomo, Indiana. He supervises and teaches the digital, microcontroller and digital signal processing sequence of courses. His other interests are in the areas of design, laboratory projects, multimedia, and Java. Professor Foster is a past chairman of the Information Systems Division of ASEE. He is actively involved with Project Lead The Way (PLTW), a national alliance for pre-engineering programs for high schools and middle schools.