A Digital Control Laboratory Built around Texas Instruments' C3x DSK Evaluation Board

Daniel J. Block

University of Illinois at Urbana-Champaign
1406 W. Green Street
Urbana, IL 61801

Abstract

This paper will give an overview of the digital control laboratory developed at the University of Illinois Urbana/Champaign. First the laboratory equipment used in the experiments will be discussed. Specifically, the TI DSP system that was developed around the C3x DSK will be detailed. The DSP system uses the C3x DSK as a motherboard and in-house built daughter cards are used to give the system its I/O capability. Four daughter cards were built: 1) 16K-256K external zero wait-state SRAM board, 2) LCD display controller board with additional 16 lines of TTL digital input and 16 lines of TTL digital output, 3) 4 channel ±10V DAC / 4 channel optical encoder input board, 4) 4 channel ±10V ADC board. The main design challenge (beside the normal noise and ground loop problems) in building these daughter cards was forcing all boards to have compatible wait-states because of the DSK's limitations in the use of the TSM320C31's RDY signal. The general design of these boards will be discussed giving the reader a start in the design of their own data acquisition boards.

This DSP system is primarily used by the Digital Control of Dynamic Systems course offered by the General Engineering department at the University of Illinois. The digital control course covers classical and modern discrete control theory. The lecture portion of the class is theoretical, covering all the mathematics of the z-transform and different discrete control design methods. The laboratory portion of the course deals with design and implementation of the discrete controllers discussed in lecture. First simulations are performed as a first check of the control design. Then the students are asked to implement their control designs on the C3x DSK system and control the actual plant. Section 3 overviews each of the six laboratory experiments that the students perform.

The future plans for the digital control course are to develop new experiments that will involve the students more with the C3x DSK. Presently a single daughter card is being developed that would give the students all the I/O needed for specific control experiments. The goal is to make the daughter card inexpensive enough so that each student in the digital control class could be given a C3x DSK board along with the single daughter card. Then both in the laboratory or at home they could develop and implement their source code for the different experiments. This "ownership" of the experiment involves the student more in the lab work and therefore teaches the student more. Obviously the challenge is keeping the system "cheap" but versatile enough to perform the different experiments. The control experiments also have to be relatively inexpensive but complicated enough to show the topics studied in the class. The current ideas and prototypes of these experiments will be presented.

1. Introduction

At the University of Illinois, it has always been very important to have hands on laboratories accompanying the Digital Control of Dynamic Systems course offered by the General Engineering department. When the lab was first created in the early 1980's, students were asked to program Apple IIe personal computers with internal data acquisition boards to control simulated systems wired on an analog computer. Over the years the lab obviously moved to faster PCs and also started introducing actual electro-mechanical systems to control. Five years ago the laboratory experiments were moved to the new College of Engineering Control Systems Laboratory [5]. One year after the move the decision was made to change the laboratory content from programming a PC with internal data acquisition cards to programming an embedded TMS320C31 DSP system with comparable data acquisition hardware. The TMS320C31 [4] was chosen as the processor because of its floating point capability and its easy-to-use optimizing C compiler. Specifics of the DSP i.e. machine instructions, assembler programming, and fixed point mathematics could not be covered in depth, though, due to time constraints in a single semester course. The focus of the lab portion of the course remains digital control design and its programming implementation in the C language. However, with the addition of the C3x DSK we now teach the use of an embedded system along with embedded system debugging techniques.

We initially purchased TMS320C31 DSP development systems from Integrated Motions Inc. These systems worked very well and we had very few problems with the equipment itself. The limitation and downfall of these systems was that they communicated with the PC through the serial port at a baud rate of 9600. Thus, real-time data download or upload was not very feasible. Slow communication proved to be an annoyance for students learning DSP programming for the first time and it became one of the largest complaints about the lab assignments.

To find a relatively inexpensive solution to this data transfer problem we began to develop our own DSP development system around the C3x DSK [3] from Texas Instruments. The C3x DSK communicates over the parallel port making its communication much faster then a 9600 baud serial port. The C3x DSK also gives access to nearly all the TMS320C31's pins, making it a low-cost motherboard to build a DSP development system around. To encourage student involvement in the development of this DSP system, the College of Engineering Control Systems Lab manager [5] offered a special projects course in which groups of two students were given the task to design and build the prototype for an I/O daughter card needed in the system. The final versions of the daughter cards were reproduced for use in the digital control class. The new system includes the C3x DSK along with four stackable daughter cards. A zero wait-state external SRAM card was built to expand the C3x DSK's memory to 66x32K words. The three remaining cards add analog and digital I/O and a LCD display. In total, the system has 4 channels of ADC input, 4 channels of DAC output, 4 channels of quadrature optical encoder input, 16 lines of digital input, and 21 lines of digital output. Note that the I/O listed above does not include the AIC (analog interface circuit) chip on the C3x DSK, which is disabled in our design. We introduced these systems into the lab portion of the digital control course three semesters ago and have been very pleased with their performance.

2. Daughter Cards for the C3x DSK

Details of the four in-house built daughter cards are given in this section. Table 1 shows the memory map for the external I/O space used by the four daughter cards. Details of the C3x DSK and the individual peripheral chips used on the boards are not given in this paper. Please refer to the TMS320C3x user manuals [3,4] and the peripheral chips' data sheets [6,7,8,9,10] for detailed information.

External SRAM	
0x80A000-0x8FFFFF	Reserved by External SRAM
0x900000-0x93FFFF	256K External SRAM
0x940000-0xBFFFFF	Reserved by External SRAM
LCD/Digital I/O Board	
0xC00000	LCD Data Read/Write
0xC00001	LCD Control Read/Write
0xC00002	Digital IN Read
0xC00003	Digital OUT Write
0xC00004-0xC0FFFF	Reserved by LCD/Digital I/O Board
0xC10000-0xC1FFFF	UNUSED
Encoder/DAC Board	
0xC20000	Encoder Read
0xC20001	DAC/Encoder Write
0xC20002	DAC/ENC Dir. And DAC Reset
0xC20003-0xC2FFFF	Reserved by Encoder/DAC board
0xC30000-0xC3FFFF	UNUSED
ADC Board	
0xC40000	ADC Write
0xC40001	ADC Read
0xC40002-0xC4FFFF	Reserved by ADC Board
0xC50000-0xDEFFFF	UNUSED

Table 1: Memory Map for all Four External Daughter Cards Fabricated for the C3x DSK.

2.1 External SRAM Board.

The C3x DSK board itself only has the TMS320C31's 2Kx32 words of on chip RAM. This is enough memory to do quite a number of control experiments even when compiling C programs for the DSP. Section 4 talks about future lab experiments that plan to use only the internal memory of the C3x DSK. For this more general purpose DSP development system, it was important to have access to more memory for use in local data collection and data storage. Also, the additional memory is used for character manipulation functions for the LCD screen which take up quite a bit of program space.

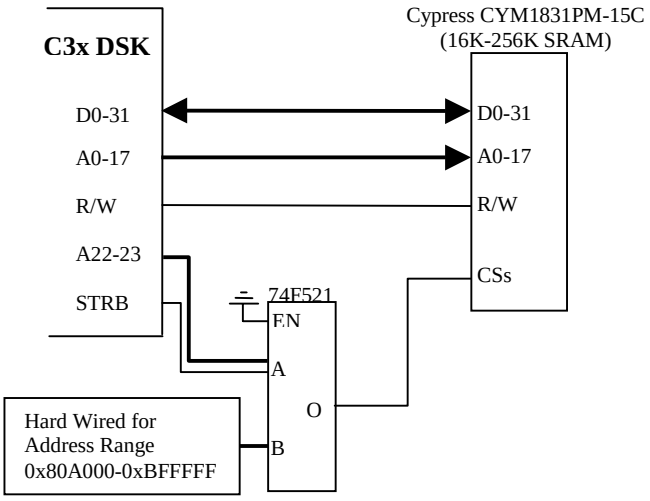

Figure 1: SRAM Daughter Card Schematic

As seen in Figure 1, the design of the external memory board for the C3x DSK is quite simple. The major design issue is finding fast enough memory and glue logic to perform read and write cycles in less than 40 ns (zero wait-state). Cypress's CYM1831PM-15C (15ns, 64x32K) SRAM SIMM was chosen for the board. 15ns is approximately the lowest value available for the zero wait-state interface. As shown below, address lines 22 and 23 along with external I/O strobe (STRB) are used to generate

the chip select pulse during the read and write cycles. Note that a "Fast TTL" comparator chip (74F521) was used to create this particular chip select (CS) signal. The fast 5ns propagation delay of this device is very important in making the external memory reliable.

One issue with this design is that it is very wasteful when it comes to address space. The board takes up the address space 0x80A000 to 0xBFFFFF. As seen later in our other board designs we also were very wasteful of the address space (See the memory map in Table 1). This is not of concern since the TMS320C31 has much more address space then the development system requires. As a result, only 256K of the 4M of address space taken by the SRAM board is useful. For the starting address of the external memory, any number of addresses could be chosen, but some choices are more obvious. We specified to the C compiler to start the external memory at 0x900000 and access until 0x910000 (0x940000 if a 256K SIMM had been installed). 0xA00000 or 0xB00000 could have also been chosen and would have accessed the same memory location as 0x900000.

A remaining design issue that we ran into with the external memory board was electrical noise generation. Interfacing external memory and analog I/O integrated circuits with the same data-lines (and more importantly the same ground plane) can induce unwanted electrical noise in the analog signals. We were not able to totally remove this noise but we did reduce it significantly by experimenting with different ground plane configurations on the memory board. Enabling the TMS320C31's cache also reduced the memory accesses enough to see a noticeable difference in the noise.

2.2 DAC Output and Optical Encoder Input Daughter Card.

A DAC/ENC card was designed to give the C3x DSK the capability to implement input/output tasks for standard industrial motion control applications. The DAC4813 [8] chip from Burr-Brown was used to produce 4 channels of ±10V analog output. Two LSI LS7266R1 [6] chips were used to give the board 4 channels of quadrature optical encoder input. Figure 2 shows a general schematic for this card. Design of this board was more challenging when compared to the memory daughter card; the main challenge (also true for the remaining two boards to be discussed) was overcoming the necessity for the zero wait-state required by the SRAM card. The C3x DSK's parallel port interface dedicates the INT2, RESET and RDY lines for its use. This requires that the daughter cards not use the RDY line to dictate the length of their read and write cycles. Because the interface with the memory card must be zero wait-state, the interface to the DAC/ENC must also be zero wait-state. This is a problem for both the DAC2815 and the LS7266R1 chips since the minimum write cycle for both is 50ns and a zero wait-state write cycle is approximately 40ns. Our solution to this problem was to interface "Fast TTL" latch and buffer chips with the DSP's address and data bus at the zero wait-state speed. The outputs and inputs of these latches and buffers are connected to the data and control pins of the I/O chips. Through repetitive software steps, the correct read and write cycles for the specific chip are manually generated by writing the correct 1 and 0 combinations to the latches. One way to look at this type of interface is communicating with a chip through a general purpose I/O port instead of the normal address and data bus read and write cycles. Looking at Figure 2, all the data lines and control signals (RD, WR, C/D, etc) of the DAC and ENC chips are connected to the output of 74F377 latches or the input of 74F541 buffers. For example, a typical write sequence is to first write the data for the I/O chip to the latches connected to D0-D7 of the I/O chip. With the same write instruction have the WR line latched HI (state 1). Then, leaving D0-D7 at their same states, pull the WR line low. Insert a delay (multiple NOP's) in your program to hold the WR line low for the I/O chip's specified time. After the delay, pull the WR line back high to complete the write cycle.

Data Bus	DAC4813	LS7266R1
D0	**D0**	**D0**
D1	**D1**	**D1**
D2	**D2**	**D2**
D3	**D3**	**D3**
D4	**D4**	**D4**
D5	**D5**	**D5**
D6	**D6**	**D6**
D7	**D7**	**D7**
D8	**D8**	**WR**
D9	**D9**	**RD**
D10	**D10**	**C/D**
D11	**D11**	**X/Y**
D12	**EN1**	**HI**
D13	**EN2**	**HI**
D14	**EN3**	**HI**
D15	**EN4**	**HI**
D16	**LDAC**	**HI**
D17	**WR**	**HI**
D18	**HI**	**CS1**
D19	**HI**	**CS2**
D20	**NC**	**NC**

Table 2: DAC/ENC Board Data Line Assignments

In the same fashion the read cycle is started by pulling the RD line low. A delay is again performed to meet timing specifications. After the delay the data is read from the buffer chip and stored in a variable on the DSP. Then completing the read cycle the RD line is pulled high. The example programming sequence for each I/O chip is given later in the paper.

With this type of interfacing method, the general purpose I/O lines have to be assigned certain I/O pins. Table 2 gives the pin assignment for the DAC/ENC daughter card. Notice that most pins are shared by both the DAC4813 and the LS7266R1 chips. The CS (WR in the DAC's case) lines, though, are dedicated to one chip. Thus, the program communicating with the DAC chip must make sure to keep those lines at the correct state (HI); otherwise, the ENC chips will also be selected and cause bus contention. For this reason, Table 1 lists some of the pins as HI.

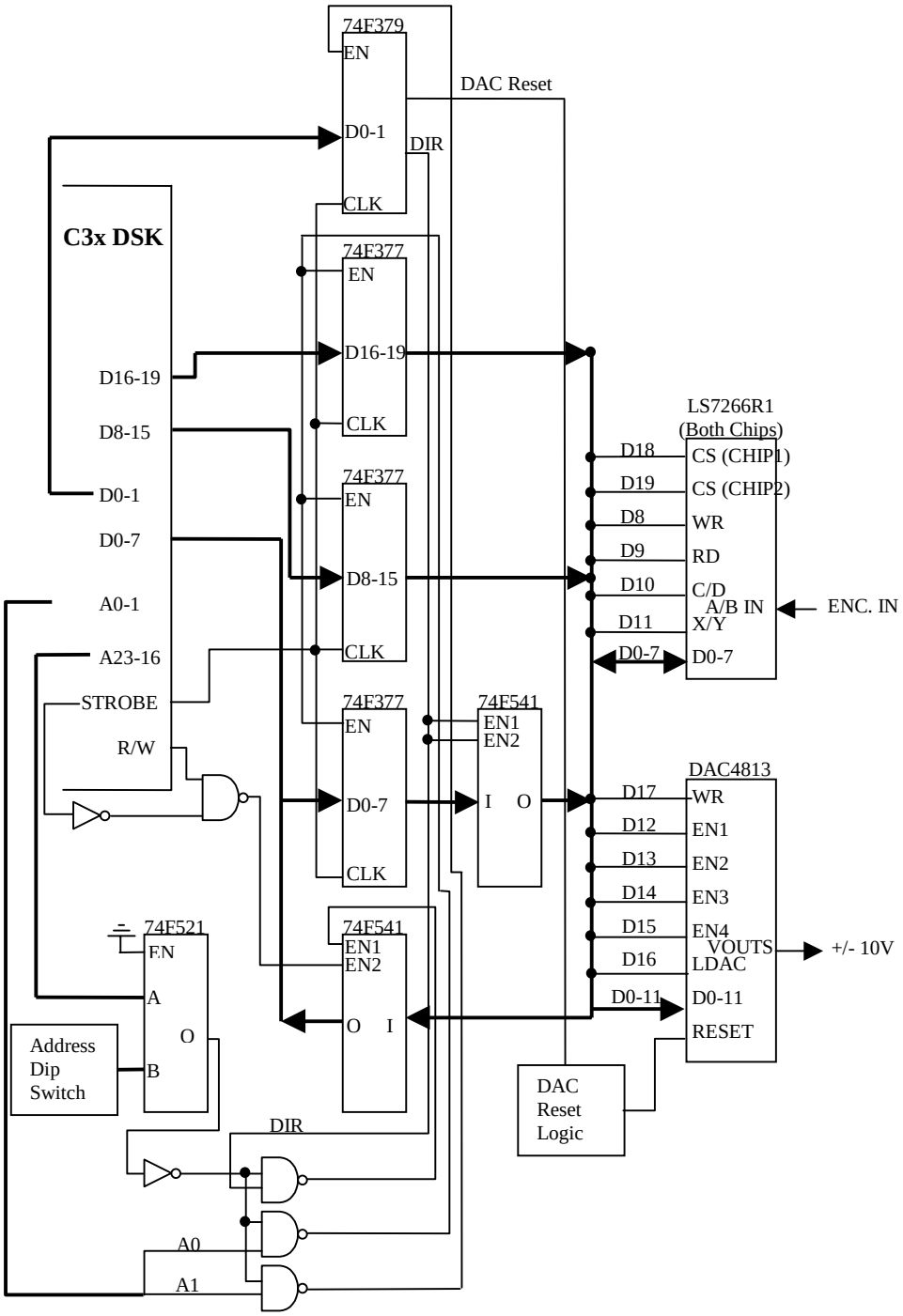

Figure 2: DAC/ENC Daughter Card Schematic

Another important aspect of this type of interface method is the delays inserted between output commands. Delays were generated by inserting NOP (No Operation) instructions between the output commands. To be precise with this method a logic analyzer should be used to determine the number of NOP instructions between the output commands. It is advisable to use a liberal number of NOP instructions for the delays when writing the DSP code in C and using the C compiler. The number of assembler instructions between output latch writes (and input reads) will vary depending on where the code is located and what optimizations have been set for the C compiler. Four NOPs between each command was chosen as the basic delay for the three I/O daughter cards. (Note that the LCD display does need longer delays in certain sections of its code.) In most sections of code this is probably longer than required but we have not had any performance problems requiring us to optimize the delays.

2.2.1 Example Programming Sequences for the DAC/ENC Card

This section demonstrates to the reader how the C3x DSK is programmed to communicate with the different I/O

chips. Note that appropriate delays need to be added between some of the instructions to produce the correct timing for the signals writing and reading the I/O chips. Reference Table 2 to decipher the different hexadecimal numbers written to the latches.

A. Writing a Value to DAC4813 Channel 0's output Register and Latching that Value to the DAC stage.
   i. Set D0 of 74F379 Low to Set the Direction of the 74F541s to WRITE.
   ii. Write 0xFE000 + 12Bit DAC value to the 74F377s
   iii. Toggle WR (D17) Low, then High, to Write the DAC value to the DAC4813's DAC0 output register.
   iv. Write 0xEF000 to the 74F377s.
   v. Toggle WR (D17) Low, then High, to Latch DAC 0's value to the output stage.

B. Writing to LS7266R1 Control Register (Both Chips at the same Time)
   i. Set D0 of 74F379 Low to Set the Direction of the 74F541s to WRITE.
   ii. Write 0x3F700 + 8 bit Register designator and value to 74F377s.
   iii. Toggle WR (D8) Low, then High, to Latch the value into the specified Register

C. Writing to LS7266 Pre-Set Register, Counter X and Y.
   i. Set D0 of 74F379 Low to Set the Direction of the 74F541s to WRITE.
   ii. Write 0x3F781 to 74F377s (RLD, Reset Byte Pointer)
   iii. Toggle WR (D8) to Latch Command in LS7266.
   iv. Write 0x3F300 + LSB 8 bits of Pre-Set value to 74F377s
   v. Toggle WR (D8) to Latch Data and automatically increment BP to next byte.
   vi. Write 0x3F300 + MIDSB 8 bits of Pre-Set value to 74F377s
   vii. Toggle WR (D8) to Latch Data and automatically increment BP to MSB.
   viii. Write 0x3F300 + MSB 8 bits of Pre-Set value to 74F377s
   ix. Toggle WR (D8) to Latch Data
   x. Repeat Steps 2-9 for Counter Y Writing 0x3FB00 + Byte in steps 4,6 and 8.
   xi. Write 0x3F788 to 74F377s (RLD, Transfer PR to CNTR)
   xii. Toggle WR (D8) to Latch Command.

D. Reading LS7266 Chip 1 Counter X's value.
   i. Set D0 of 74F379 Low to Set the Direction of the 74F541s to WRITE.
   ii. Write 0x3F791 to 74F377s (RLD, Transfer CNTR to OL and Reset BP)
   iii. Set D0 of 74F379 HI to Set the Direction of the 74F541s to READ.
   iv. Write 0xBF300 to 74F377s. (Note: In READ mode so least significant 8 bits are Don't Care)
   v. Toggle RD (D9) Low to initiate Read Cycle
   vi. Read D0-D7 (LSB byte)
   vii. Toggle RD (D9) HI, LS7266 BP is automatically incremented.
   viii. Toggle RD (D9) Low, Second Read
   ix. Read D0-D7 (MID SB byte)
   x. Toggle RD (D9) HI, LS7266 BP is automatically incremented.
   xi. Toggle RD (D9) Low, Third Read.
   xii. Read D0-D7 (MSB byte)
   xiii. Toggle RD (D9) HI.

2.3 ADC Input Daughter Card.

We have designed an ADC Daughter Card to further enhance the I/O capability of the C3x DSK. Many of the feedback signals used in the control experiments in the Control Systems Lab have analog feedback. Linear and rotational potentiometers and analog DC tachometers are the best examples, however students have also used strain gage and temperature measurements for special projects. All these signals can then be brought into the C3x DSK through the 4 channels of analog to digital converters on the ADC daughter card. Four individual AD1674 [7] (Analog Devices) chips were used. This gives the board simultaneous sampling capability and a four times greater sampling rate when compared to a multiplexed ADC circuit. Looking at Figure 3, note that the ADC board is not quite as complicated as the previous DAC/ENC board. It does require latches and buffers to interface with the chips on the board (since the read and start conversion cycles require delays of 50ns or greater). There are fewer control signals for these chips. For instance, there is one output control-line for each chip's chip select (CS) and the read/convert (R/C). The remaining lines, D0-D11 and each chip's STATUS are inputs. Table 3 indicates the I/O line assignments used. To start a convergence of an ADC, the CS line is pulled low and then the R/C line is pulsed. The C3x DSK program repeatedly reads (polling) the status line for the specified chip (or chips). When the status line is low, the ADC has converged and the data lines can be read.

An improvement to this board would be to have the status line interrupt the DSP when the AD1674 has converged. We are presently working on a board with this capability for our future DSP system. Again, in the attempt to minimize electrical noise pickup on the ADC input, separate analog and digital ground planes were used on the board. These ground planes were then connected together at a single point on the board. Refer to the AD1674 data sheet for recommendations on board layout for this chip.

Data Bus	AD1674s
D0	**D0**
D1	**D1**
D2	**D2**
D3	**D3**
D4	**D4**
D5	**D5**
D6	**D6**
D7	**D7**
D8	**D8**
D9	**D9**
D10	**D10**
D11	**D11**
D12	**STATUS1**
D13	**R/C**
D14	**CS1**
D15	**CS2**
D16	**CS3**
D17	**CS4**
D18	**STATUS2**
D19	**STATUS3**
D20	**STATUS4**

Table 3: ADC Board Data Line Assignments

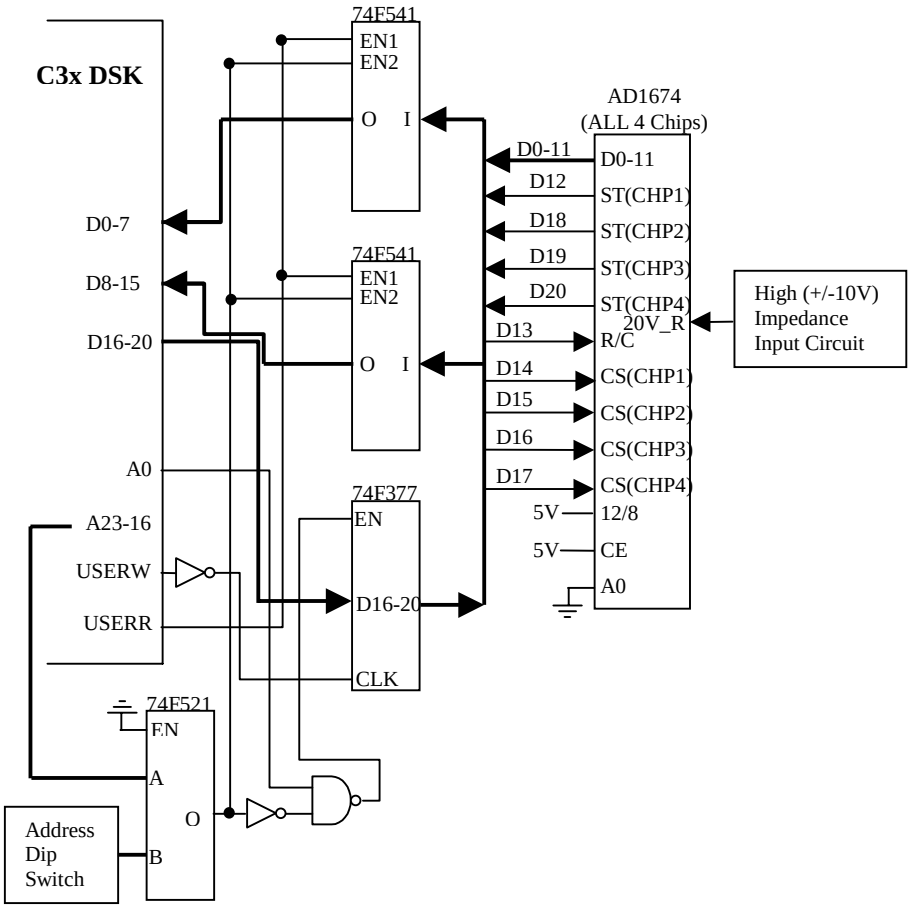

Figure 3: ADC Daughter Card Schematic

2.3.1 Example Programming Sequences for the ADC Card

A. Converting and Reading the ADC, 3rd Channel
   i. Write 0x2E000 to the 74F377 (Set CS for Chip 3).
   ii. Toggle R/C (D13) Low, then HI, to start ADC Conversion.
   iii. In a continuous While Loop Read in D0-D15 checking for STATUS3 (D19) to go Low, Conversion Complete.
   iv. Perform a final Read to pull in the 12 bit ADC value in D0-D11.
   v. Write 0x3E000 to the 74F377 to deselect Chip 3.

2.4 LCD/Digital I/O Daughter Card.

The final daughter card designed for the DSP development system was a general purpose digital I/O board that has one digital I/O port dedicated to driving a Optrex DMC-20261A LCD screen [10]. This board adds 16 lines of digital input and 21 lines of digital output to the C3x DSK system. The LCD screen has proven to be a very important part of the development system for students. Without the LCD screen, students would have a harder time debugging their program's source code. The IMI DSP system that we initially purchased for the digital control labs had an 8 character LCD display. It was somewhat useful, but became problematic when trying to print out floating point numbers. For this new system we chose a 40 character display (2x20). With this display size, students have a reasonably large console for printing their debug messages.

After the design of the previous three I/O daughter cards, the design of the LCD board was straightforward. The LCD daughter card only requires the front-end latches and buffers of the DAC/ENC or ADC board. We switched from the 74F377 latch to the 74F573 latch since it had tri-state outputs with an output enable (OE) control line. This eliminated the need for additional buffers for the read operations. Figure 4 shows the schematic for the LCD section of the board only. Table 4 shows the pin assignment for the LCD section of the daughter card. The additional 32 digital I/O section of the board is not shown because it is in effect an exact copy of the LCD control section. The Y2 and Y3 pins of the 74F138 decoder chip are used to select this section of the board for either 16 bit digital reads or 16 bit digital writes.

Data Bus	74F541/ 74F573 Data	74F541/ 74F573 Control
D0	**LCD D0**	**LCD RS**
D1	**LCD D1**	**LCD R/W**
D2	**LCD D2**	**LCD E**
D3	**LCD D3**	**Spare Out**
D4	**LCD D4**	**Spare Out**
D5	**LCD D5**	**Spare Out**
D6	**LCD D6**	**Spare Out**
D7	**LCD D7**	**Spare Out**

Table 4: LCD Board Data Line Assignments

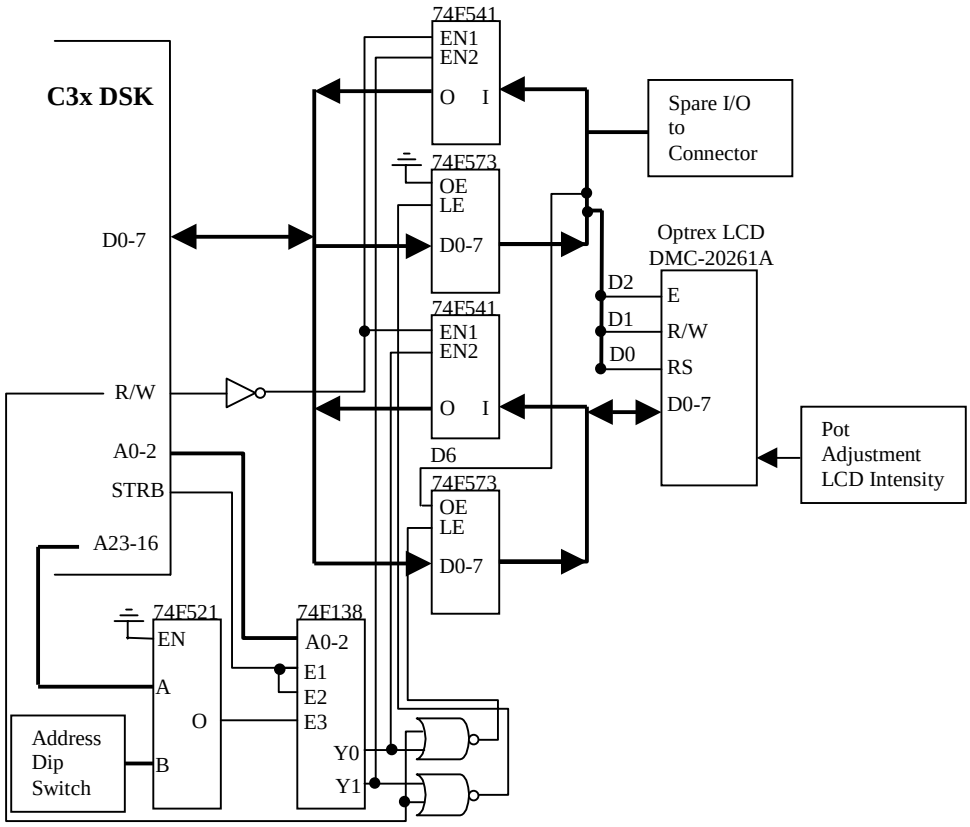

Figure 4: LCD display daughter Card Schematic

2.4.1 Example Programming Sequences for the LCD card.

LCD displays are much slower devices when compared to the ADC or DAC chips. Long program delays are needed in the given programming sequences and in-between instructions to the LCD. Refer to the DMC-20261A [10] data sheet for timing specifications.

A. Writing a Character to the LCD display.
   i. Write 0x1 to the "Control" 74F573: WRITE Mode, Data Input.
   ii. Write 8 bit Character to the "Data" 74F573.
   iii. Toggle E (Control D2) HI then LOW Latching in Character.

B. Writing an Instruction to the LCD.
   i. Write 0x0 to the "Control" 74F573: WRITE Mode, Instruction Input.
   ii. Write 8 bit Instruction to the "Data" 74F573.
   iii. Toggle E (Control D2) HI then LOW Latching in Instruction.

C. Reading Status from the LCD.
   i. Write 0x2 to the "Control" 74F573: READ Mode, Instruction Input.
   ii. Toggle E (Control D2) HI.
   iii. Read in 8 bit data from "Data" 74F541.
   iv. Toggle E (Control D2) LOW.

3. Laboratory Experiments

The digital control class utilizing the C3x DSK system is a 4 credit hour undergraduate and beginning graduate level course with approximately 20-30 students per semester. The class is offered by the General Engineering department; however, a large number of Electrical and Mechanical engineering students also take this class. The lecture portion of the class remains mostly theoretical, covering all the mathematics of the z-transform and many different discrete control design methods. The lab portion of the class allows the students to use the theory taught in the lecture and apply it to an actual system. Each lab has a simulation portion that the students are asked to complete before coming to lab. Simulink is used for most of the simulations, while Matlab for the controller design. The in-lab assignments ask the students to program their controller designs on the C3x DSK system. The first controller's source code is given to them for use as a "starter shell" to implement the remaining lab experiments. Three "plants" are controlled through out the course of the semester: a DC motor system with potentiometer angle feedback (Figure 5), the same DC motor with a flexible torsion spring coupling two flywheel inertias (Figure 6), and an in-house built inverted pendulum experiment that has been nicknamed the "Pendubot" [1,2] (Figure 7).

3.1 System Identification of a DC Motor.

As with any control design, the first step is to identify a model for the system to be controlled. In this lab the students are asked to use two techniques to identify a DC motor system. Students are first asked to use a HP35670A dynamic signal analyzer to identify a linear transfer function for the DC motor. The DSA returns the continuous transfer function for the motor in terms of tachometer voltage over amplifier input voltage. The

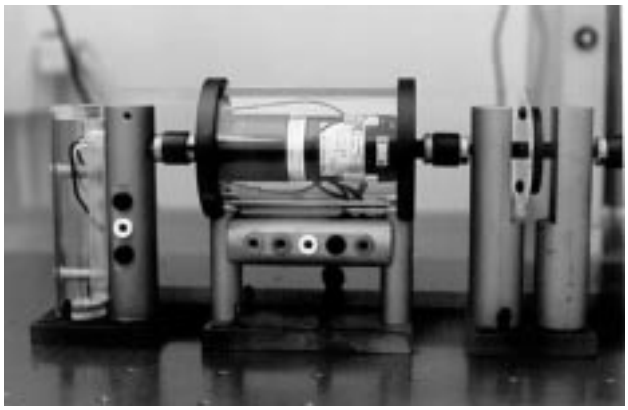

Figure 5: DC Motor Plant

students then discretize the DSA transfer function using four sample rates, 1ms, 5ms, 15ms and 40 ms. These discrete transfer functions are taken to be the "correct" transfer function for comparison. A simple least-squares technique is then used to directly identify the discrete difference equations/transfer function for the motor in the time domain. Open-loop impulse and step inputs are applied to the motor while the response data is collected in Matlab. With this collected data and the assumed structure of the difference equations, a set of over-determined linear equations can be formed. Solving this set of equations produces the difference equation's coefficients. Using several impulse and step inputs at different sample rates, the students repeat this same procedure. One lesson we attempt to teach the students is that the impulse inputs do not do a good job in identifying the system mainly because they do not excite the system well enough. They also find that for identification purposes faster sample rates are not necessarily better. The 1ms sample rate's runs produce larger error when compared to the other sample rates due to the numerical sensitivity of the z-transform at fast sample rates. This leads students then to see that the fastest sample rate possible (hardware dependent) is not necessarily the correct choice for a given control design.

3.2 PD Position Control of the DC Servo Motor.

The bulk of this laboratory is the calculation and simulation of the student's PD controllers. The students are given two different design specifications and asked to design PD controllers with sample rates 1ms, 5ms, 15ms, and 40ms. Using hand calculations and continuous-time second-order system rules of thumb, they design their PD controller to give the closed loop system the desired design specifications. Using the backward difference approximation of the differentiator they simulate their PD controllers at the different sample rates. Here they discover that the continuous gains that worked for a 1ms sample rate controller no longer meet the specifications when used with the slower sample rates. This is a perfect introduction to root locus design in the z-domain. They also find that at the slower sample rates the specifications cannot be met. Manually they tune the PD controllers to achieve either better performance or what they feel is the best design that can be achieved with the given sample rate. In lab each student is given the source code for a PD controller implemented on the C3x DSK system. They will use this starter file in the upcoming labs to implement their other control designs. With the starter file they go through the process of compiling a COFF file for the DSP, downloading the COFF file to the C3x DSK and modifying the code to print out desired debug messages to the LCD screen. Using the gains found in their simulations, they control the DC motor at the four different sample rates. Here they will notice that unmodeled non-linearities, mainly friction, cause their simulations not to match the actual step responses. Manual tuning is performed to make the actual response meet or exceed the design specifications.

The given PD controller code uses a 1 ms sample rate and implements a digital filter to remove some of the noise from the velocity calculation. At slower sample rates the noise is not as prominent and the students find that they must reduce the order, or even remove the filter, in order not to introduce significant phase lag in their PD controller.

3.3 PID Position Control of a DC Servo Motor.

Lab 3 is an extension to lab 2's PD control. In this laboratory assignment students are asked to control the same DC motor used in the past two labs with a PID controller. Using root locus techniques, they first design their PID controller in the continuous domain. The PID introduces a second pole and zero making its design quite a bit more complicated. The students see that with the extra zeros of the PID controller the standard rules of thumb for a second order system do not necessarily give accurate approximations. For the simulation section of the lab, they are asked to first emulate their continuous controller with both the bilinear (integral term) and backwards difference (differentiation term) integration rules. With Simulink, the students then run simulations of their controller at the two sample rates. If design specifications are not met, the students manually tune the control to meet or exceed the specifications.

To complete the experiment the students are asked to implement their PID controller on the C3x DSK. Taking the PD control program given in Lab 2 as a starter shell, they modify the PD controller's source code to implement the integral portion. They are asked to think about the inherent integral wind-up problem of integral control and develop a software solution for this problem. Running their controller, they often find that the unmodeled dynamics of the DC motor may cause disagreement with simulation. If needed, they are asked to manually tune the PID control to attempt to exceed the design specifications.

3.4 State Space Control of a DC Servo Motor with Added Flexible Torsion Link.

In Lab 4, a flexible torsion link is added to the DC servo motor setup of labs 1-3 (See Figure 6). On the opposite end of the torsion rod there is a second flywheel along with another potentiometer to add higher-order dynamics to the DC motor system. Four states are assumed to be known with this system: the angle (θ_1) and angular velocity (θ_1', found by discrete differentiation of the position signal) of the DC motor and its colocated flywheel, the angle (θ_2) and angular velocity (θ_2', also found by discrete differentiation) of the non-colocated flywheel. The students are asked to design a position controller meeting the given step response design specifications for the non-colocated flywheel.

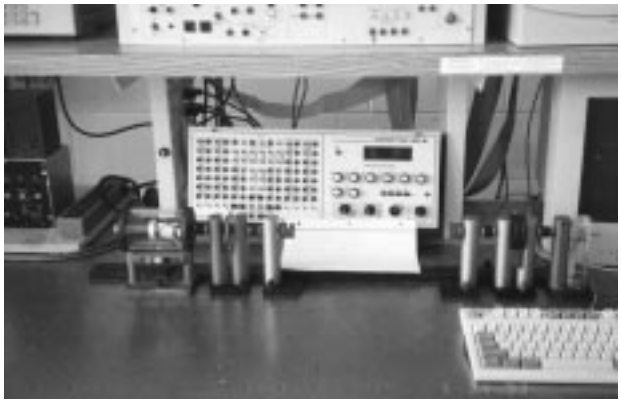

Figure 6: Flexible Torsion System

With this plant we introduce the students to state space control methods. Given a state space model, identified with the DSA, students use discrete pole placement techniques to find a state feedback control law of the form u = -K*x where $x_1=\theta_1$, $x_2=\theta_1$', $x_3=\theta_2$, $x_4=\theta_2$'. This controller regulates the torsion system at one position. To track the reference step input, students must modify the control law to add the reference signal. The final control law (u= -K*x + Kpf*r) adds the reference multiplied by a prefilter gain. The students choose the prefilter by finding the amount of gain that makes the DC gain of the closed loop system (Y(z)/R(z)) equal to one. Simulations of their controller are performed to check performance and reference tracking.

With their state space controller completed, the students are asked to implement their design on the C3x DSK. In running their controllers they again find differences between the simulation runs and the actual runs. Unmodeled friction is the cause for most of these differences. This creates an introduction to the addition of an integral state in the state space design. The lab assignment walks the students through the addition of a state that is the integral of the error (r-θ_2). The students repeat the pole placement design for the new augmented system and implement the design on the actual system noting any improvements and/or problems.

3.5 State Space Control of an Inverted Pendulum experiment (The Pendubot).

In this lab students are introduced to an inverted pendulum experiment that we have built in house and nicknamed the "Pendubot" [1,2] (See Figure 7). The Pendubot is a two link revolute robot that has both of its rotational joints in the vertical plane. The second link of this robot is unactuated, giving it the same properties as a free-swinging pendulum (thus the name pendu-bot). The goal of the control design for this system is to balance the second link in a inverted vertical position using the actuation of the first link and the angular feedback of each link.

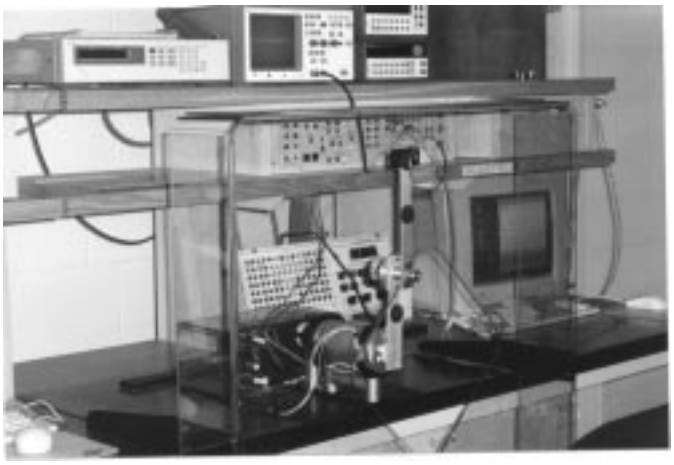

Figure 7: The Pendubot

As in the previous lab, this lab emphasizes state space design and pole placement. However, a difference in the Pendubot's control compared to the previous lab's torsion plant is that the dynamic equations of the Pendubot are non-linear. With this plant students must work with the linearized dynamics of the Pendubot about a desired operating point to design a stablizing controller. The goal of the control is only to stablize the second link in the balancing position so their control law is in the form u = -K*x, where x ($\delta\theta_1$,$\delta\theta_1$',$\delta\theta_2$,$\delta\theta_2$') are the "delta" states of the linear Taylor series approximation [1]. The students are asked to first simulate their controller in Simulink using a given non-linear simulation block of the Pendubot. By using the non-linear simulation of the Pendubot, students investigate in simulation the region of attraction of their linear controller. They also test the robustness of their controller by simulating an external impulse force (figure tap) applied to the second link.

Implementation of the state feedback controller on the C3x DSK is similar to the state space controller for the torsion plant. However, since the linear balancing controller is only locally stable, the students are asked to add a safety check feature to their controller. This safety

check monitors the positions of the links. If the links move outside of an approximate region of attraction of the balancing controller, the code switches to a PD controller that controls only the position of link one. This way if the student gives too large of a tap to the second link the (no longer valid) balancing controller will not cause the linkage to spin wildly out of control. If the student then moves the second link back into the region of attraction the balancing controller will switch back on. This introduces the student to programming a type of switching control.

3.6 Design of a Linear Observer used in the Control of the Pendubot.

The sixth lab is a repeat of the previous lab controlling the Pendubot. Again the goal of the control design is to stablize the Pendubot's unactuated second link in the vertical (open-loop unstable) position. The difference between these labs is the method in which the link's velocities are calculated. In the previous lab the velocities are found by discrete differentiation (backwards difference) of the position signals. This method of differentiation creates noise in the velocity signal. To reduce this noise jitter a digital filter must be applied to the velocity signal.

This lab demonstrates the use of a full order observer to calculate the velocities given only the positions of the links. Using the linearized system found in the previous lab, students are asked to first design and test their state space control plus linear observer in Simulink. Again the non-linear simulation block is used in the simulations so that the students can see the limitations of their linear controller on the non-linear system. An approximate region of attraction for their controller plus robustness to external forces are simulated and compared to the previous lab's results. The estimated states are also plotted to see their convergence rate to the actual states.

Implementation of the linear observer on the C3x DSK is straight forward but has quite a few more calculations when compared to the previous lab's implementation. These larger number of calculations leads into a discussion question on why in higher order systems reduced order observers are much more feasible. Here students will more than likely need to take advantage of the debugging techniques they have learned from the previous lab assignments. Students are again asked to program a safety check into to their controller that switches to a stable (at least for link one) controller when the links move outside of a region of attraction for the balancing control.

4. Future Developments

At the College of Engineering Control Systems Laboratory [5] at the University of Illinois, we are working on some new experiments that could be used in this digital control course or in a second advanced digital control course. The concept is to develop experiments around the C3x DSK that are inexpensive enough to reproduce in large numbers. This will allow us to hand each experiment out to the students for the duration of the semester. We have found that this type of ownership of the lab experiments generates a large amount of interest in the material taught in the class. This past semester we offered a mechatronics class that tried this method of lab assignments using the "Basic Stamp" micro-controller from Parallax Inc. Listening to student feedback and looking at the level of difficulty students chose for their final projects for the course, we determined the class a large success.

With this mechatronics class as an example we looked at our other classes and determined that digital control using the C3x DSK would be another route to take to develop inexpensive laboratory experiments. The current embedded controller system on the drawing table is to use the C3x DSK along with a single daughter card that produces and reads the control signals needed. At this point we would like to try to develop the experiments so that only the 2K of internal memory on the TMS320C31 is needed. This will alleviate the need for external SRAM on the daughter.

All the miniaturized control experiments will be based on a small low torque 24 volt DC motor from Pittman Inc. ($90 a piece in quantity of 20). This motor also comes with a 500 cnt/div quadrature optical encoder. To drive this motor an inexpensive ($4.50 a piece) PWM amplifier was chosen from Allegro Inc. This small 16 pin dip amp chip can drive 30 volts with 3.5 amp peak current and 2.0 amp continuous current; a good match for the Pittman motor that has a stall current of 2 amps. So far with this small motor and amp system we have developed two control experiments. The first experiment is a small version of the Pendubot (Figure 8). Its links are made of

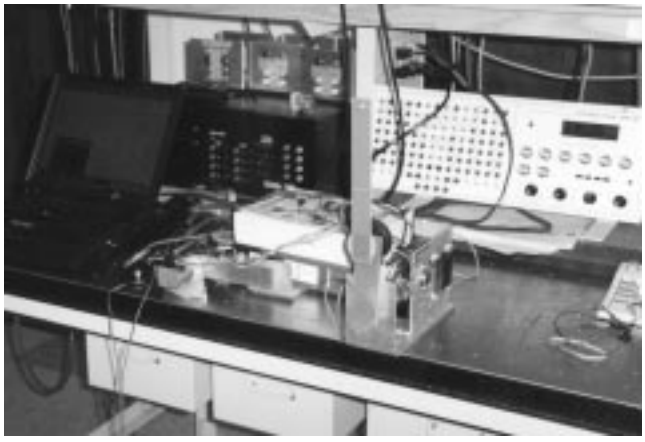

Figure 8: Miniature Pendubot

plastic making it light enough for the small motor to control. The second is another inverted pendulum experiment. This plant is simply a single pendulum with a motor attached at the free end (Figure 9). The motor's

flywheel inertia can be used to generate a torque on the pendulum by accelerating the inertia of the motor and

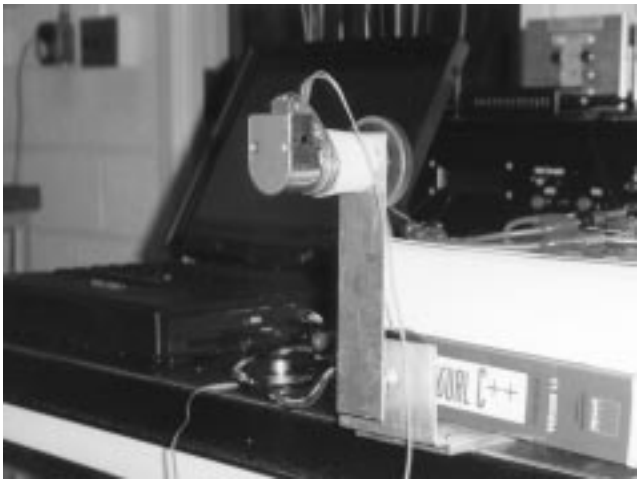

Figure 9: Inertial Balanced Inverted Pendulum

flywheel. The dynamics of this system are the same as a simple swinging pendulum making it a good simple introduction to non-linear system control. For a third experiment we have plans of making a simple fly-ball governor plant that demonstrates speed regulation.

For each of the describe plants, the DSP controller will need to output a PWM signal to drive the motor and read quadrature optical encoder channels for the feedback signals. The daughter card is then quite simple only requiring two main chips. The LS7266R1 chip to read two optical encoder signals and for the PWM output a single timer chip (i.e. 82C54) can be used. Since we plan on not using an external memory daughter card, there is no longer a requirement for having the new daughter card communicate at a zero wait state interface. This removes the needed latches and buffers used in our general purpose DSP system. The DSP data bus can be programmed for 1 or 2 wait states and the chips can be written to in the normal read and write cycle fashion.

All this development is in the beginning stages and there are a number items to over come before we implement these types of experiments in our lab course. One issue is developing all the labs to fit in the 2K of internal memory of the TMS320C31. The main reason this small amount of memory could be a problem is that we would like to stay with C programming for the implementation of the digital controllers. Also we plan on adding a LCD screen to the system for debugging purposes. Since character manipulation functions like "sprintf" take up relatively large amounts memory, we will have to develop some basic character manipulation functions take perform our needed tasks. We are also looking at using a serial LCD screen instead of a parallel interface type to see if the interface code is reduced.

The packaging of all these experiments is going to be a challenge. We need to come up with some kind of student "toolbox" that will contain all the items needed for their experiments. This way the students will be able to bring the experiments into lab to get started and also get help from the professor or teaching assistants and then pack it all up and take it home to finish up the lab.

Probably the largest issue is the maintenance and support for all these experiments. Not only repairing components that break, but also helping the students in the installation and support of the software they will need to run on their home PCs. This could be become a large problem and if not handled well create the opposite effect and make the students uninterested in the digital control experiments. Extensive, detailed documentation and trained teaching assistants will be a key in getting this method to work well in the class environment.

References

[1] Block, D.J., *Mechanical Design and Control of the Pendubot*, M.S. Thesis, Department of General Engineering, 1996.

[2] Block, D.J., and Spong, M.W., "Mechanical Design & Control of the Pendubot," *SAE Earthmoving Industry Conference*, Peoria, IL, April 4-5, 1995.

[3] *TMS320C3x DSP Starter Kit User's Guide*, Texas Instruments, Inc., Dallas, TX, 1996.

[4] *TMS320C3x User's Guide*, Texas Instruments, Inc., Dallas, TX, 1997.

[5] Alleyne, A., et.al., "A College-wide Laboratory-Based Program in Control Systems Technology at The University of Illinois at Urabana-Champaign," *1996 Conference on Decision and Control*, Kobe, Japan, Dec. 1996.

[6] LS7266R1 24-Bit Dual-Axis Quadrature Counter, LSI Computer Systems, Inc., Melville, NY, November 1996.

[7] AD1674 12-Bit 100 kSPS A/D Converter Rev. C, Analog Devices Corp., Norwood, MA, March 1994.

[8] DAC2815 Dual 12-Bit Digital-To-Analog Converter, Burr-Brown Corp., Tucson, AZ, 1995.

[9] CYM1831 64K x 32 Static RAM Module, Cypress Semiconductor Corp., San Jose, CA, May, 1995.

[10] DMC-202621 (20character x 2 lines) Liquid Crystal Display, Optrex America Inc., Plymouth, MI, November 1995.