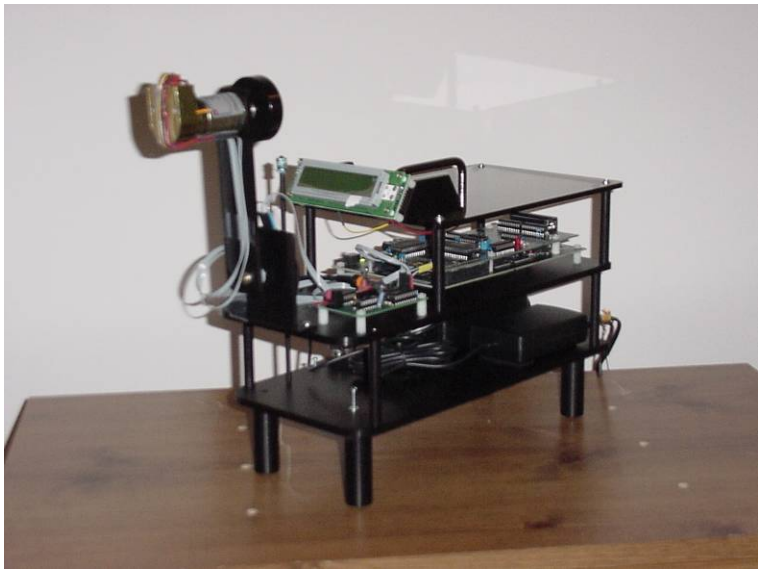
MECHATRONICS CONTROL KIT
USER'S MANUAL



IMPORTANT NOTICE!

Do not attempt to operate the Mechatronics Control Kit without proper supervision and without first reading this manual. Quanser Consulting Inc. assumes no responsibility for injuries or damage caused by improper usage.

Table of Contents

Chapter 1 Introduction

Thank you for purchasing the **Mechatronics Control Kit** from **Quanser Consulting Inc.** The Mechatronics Control Kit is developed for instruction and research in real-time control and identification. The Mechatronics Control Kit comes with four ready to assemble plants, a **DC-motor,** a **Pendubot,** a **Reaction Wheel Pendulum** and a **Furuta Pendulum**. The digital electronics are fully integrated and include a Texas Instruments DSP development system, the TMS320C6713 DSK Board (a DSP board with USB interface), a PWM/Optical Encoder data Acquisition Daughter Board, a PWM amplifier, 5 Volt and 24 Volt DC power supplies and all required cables. Additional hardware includes a 24 Volt DC motor with 1000 counts/rev optical encoder. A second 1000 counts/rev optical encoder and aluminum links and mounts to construct the above experiments. An interface board for running other Quanser experiments is also included.

The software supplied with the Mechatronics Control Kit includes the Texas Instruments *Code Composer Studio* supplied with the DSK Board, the TI C6x Optimizing C-compiler, the Code Composer Development/Debug IDE (integrated development environment), as well as DSP BIOS/RTDX realtime debugging/plotting capabilities. Example source files of different controllers for the plants and Visual Basic Interface software examples are also included as is all source code.

The Mechatronics Control Kit can be used at all levels of instruction:

- In freshman level courses as a demonstration tool to motivate important concepts in dynamics and control and the systems approach to engineering design,
- In junior level courses in frequency domain and state space control system analysis and design,
- In senior level courses in digital control, mechatronics, and real time programming, and,
- In graduate level courses in linear and nonlinear control, intelligent control, and robotics

The Mechatronics Control Kit is also sophisticated enough to be used as a research tool to investigate:

- Geometric nonlinear control
- Robust and adaptive control and identification
- Intelligent control, including neural networks, fuzzy logic, and genetic algorithms
- Hybrid and switching control
- Modeling, identification and control of friction
- Analysis and control of chaotic dynamics

Before attempting to install and operate the system, it is important that you familiarize yourself with all hardware and software supplied. Read all sections of this manual completely. In addition, you should read the manuals included from the third party vendors.

1. About Quanser Consulting, Inc.

Quanser is the world leader in the design and manufacture of advanced systems for real-time control design and implementation used in education and research. Our control challenges and solutions are ideal for implementing and evaluating feedback strategies such as PID, LQG, H infinity, fuzzy, neural nets, adaptive, and nonlinear controllers. Quanser control challenges and solutions are operational in over 500 institutions worldwide including universities, research laboratories and commercial organizations.

Quanser offers a variety of control challenges that are appropriate for all levels of university education and research. Our control experiments are distinctively modular which enable you to cost effectively employ the same power plant to perform experiments of varying complexity. By coupling the appropriate module to the plant you achieve configurations ranging from simple position servo control to advanced MIMO systems such as the Seesaw/Pendulum. When all is said and done, you receive a greater variety of experiments to explore and save money in the process! See the Quanser Difference for more information on how we stack up in the marketplace.

All our control challenges can be managed via our complete line of control solution software and data acquisition hardware so you can create fully self-contained control workstations. Our solutions can be used to deploy an embedded system, explore hardware-in-the-loop applications, as well as control a number of alternative hardware systems. Our systems are fully compatible with MATLAB, Simulink and Real-Time Workshop and operate in a variety of computing environments including: Windows 95/98/Me/NT/2000 and RTLinux.

For more information on Quanser or a free video detailing our complete line of control challenges and solutions please contact us directly at info@quanser.com.

Quanser was launched in 1990 by Dr. Jacob Apkarian, for the purpose of enhancing and advancing control theory education. While teaching at a Canadian University Jacob discovered that control theory was often a difficult concept for engineering students to grasp and many courses lacked the tools and means to translate control into tangible concepts. In light of this gap, Jacob set to work and began to envisage a highly engaging environment that enabled engineering students to explore a range of control concepts as well as develop, implement and test ideas of their own.

Since that time Quanser has evolved considerably and employs a diverse team of professional engineers and support staff who have been charged with the mandate to continually create, build and deliver a dynamic range of control theory challenges and solutions to the educational and industrial marketplaces. Today, Quanser is recognized as the world leader in the development and manufacture of real-time control design and implementation used in education and research. Although our systems are utilized in a range of organizations and applications, they continue to enrich the learning experience and enable researches to realize greater and more exciting advancements in the field of control theory.

1. How to Contact Quanser

FOR SALES AND SERVICE:

Quanser Consulting, Inc.

80 Esna Park Drive Unit 1-3
Markham, ON, L3R 2R6
Canada

Tel: 1-905-940-3575
Fax: 1-905-940-3576

E-mail: info@quanser.com

FOR TECHNICAL SUPPORT:

World Wide Web: *http://www.quanser.com/english/html/support/fs_support.html*

E-mail: *tech@quanser.com*

2. What Can You Do With the Mechatronics Control Kit?

The Mechatronics Control Kit possesses many attractive features for control research and education. Using this kit one can investigate system identification, linear control, nonlinear control, optimal control, learning control, robust and adaptive control, fuzzy logic control, intelligent control, hybrid and switching control, gain scheduling, and other control paradigms. One can program the plants supplied with the kit for position and speed control, friction compensation, swing-up control, balancing, regulation and tracking, identification, gain scheduling, and disturbance rejection to name just a few of the applications.
Below is a description of several interesting problems that you can use to develop control projects and to design and implementing your own algorithms.

- **Identification:** The first step in any control system design is to develop a mathematical model of the system to be controlled. The Mechatronics Control Kit comes with parts to assemble four distinct plants: A DC-motor with load inertia, a Reaction Wheel Pendulum, a Pendubot and a Furuta Pendulum. The dynamic equations for these plants are given in terms of certain parameters, such as the link masses, moments of inertia, torque constants, etc. Identification methods can be applied to determine the numerical values of these parameters for later use in controller design.
- **Friction Compensation:** The effect of friction in the motor brushes and bearings generally results in limit cycle behavior unless actively compensated by the controller. Friction modeling and control is of current research interest and Mechatronics Control Kit is a good vehicle for experimental verification of theoretical results in this area.
- **Position and Speed Control:** The Kit can be configured to control a DC-motor with attached load inertia. Position and speed control of this second order system is the natural first laboratory exercise for beginning students.
- **Balancing:** A more challenging laboratory exercise is to configure the Kit for either a Reaction Wheel Pendulum or a Pendubot. These systems are fourth order and can be used to investigate stabilization of either the lower equilibrium configuration or the inverted equilibrium configuration. We have supplied controllers designed using Linear Quadratic Optimal Control theory to balance the three inverted pendulum experiments at various equilibrium configurations. You may design additional such controllers using other methods such as fuzzy control, pseudo linearization, robust control, variable structure control, etc.
- **Swing-up:** The problem of swinging the Reaction Wheel Pendulum, the Pendubot or the Furuta Pendulum from the open loop stable configuration to the inverted equilibrium is an interesting problem because of the strong nonlinearity and dynamic coupling between the degrees of freedom. We have supplied controllers that are based on the notion of partial feedback linearization and nonlinear zero dynamics that can be used for swing-up control. You may design additional controllers using heuristic, fuzzy logic, or machine learning methods, or a host of other available techniques.
- **Swing-up and Balance:** Combining the operations of swing-up and balance is an ideal problem to investigate so-called hybrid and switching controllers. So-called Logic Based Switching control is a relatively new approach to designing robust controllers for complex systems.

Chapter 2 Getting Started

We have supplied you with most of the reaction wheel experiment assembled. We are going to first have you assemble this experiment and control it with the program flashed on the C6713DSK board. First step is to unpack all the pieces. Second step is to read the rest of this manual to get a better feel for the experiment you will be building.

1. Running Your First Controller

1. After familiarizing yourself with the manual and the various components of the Kit, you are ready to assemble the reaction wheel experiment and run the demo control program residing in flash memory. The next few steps will help you assemble the reaction wheel experiment. The encoder is shipped already attached to the encoder mount. You will need to attach the encoder mount and encoder to the base plate using the two 6-32 screws supplied. These screws are located in mounting holes at the front end of the base plate. Remove them and use them to attach the encoder mount to the base plate.
2. Now slide the reaction wheel link coupler over the encoder shaft. Slide it almost to the gap in the encoder shaft but leave a little space so that the coupler is not rubbing on the stationary portion of the encoder. **DO NOT force the coupler on the encoder shaft**. Take your time lining up the hole so that it slides easily on the shaft. See the reaction wheel assembly instructions if you have trouble sliding the link on.
3. Tighten the link coupler socket head cap screw with a 5/64 inch Allen wrench.
4. Now use the cable routing section for the reaction wheel to guide you through the routing of the motor and encoder cable.
5. Again, using the cable routing section as a guide, plug the motor encoder cable into ENC2 on the Quanser interface board. Plug the US Digital encoder cable into ENC1 on the Quanser interface board. Note that Pin 1 is ground on the encoder connectors.
6. Screw the M+ (red) and M- (black) leads into the PWM Amplifier screw terminals.
7. Make sure that the power to both power supplies is off and plug the two power COAX cables into both the C6713DSK and the Quanser interface board. The 5VDC power supply is the larger of the two power supplies. The 5VDC power supply powers the C6713DSK. The 24VDC power supply is the small power supply. The 24VDC power supply powers the Quanser Interface and PWM AMP board.
8. Verify that the three USER_SWITCHES on the C6713 DSK are in the UP position.
9. Now you should be ready to turn on power. Check the PC boards making sure no metal objects or obstacles are laying on them. Then plug the two power cords into a dedicated power strip and turn on power.
10. The LCD should print out a message indicating that it is ready to start the Reaction Wheel controller. The three LEDs should also be blinking on and off.
11. Switch the PWM AMP enable switch from OFF to ON. The LED should turn ON. Note: The motor should not be spinning at this point, and if it is there is some kind of problem, possibly a wiring error. The motor should not spin until you tell the controller to GO in the next step.
12. Wait for the reaction wheel link to be a rest and then tap on DIP_2 switch. The reaction wheel should start to swing up to the TOP position. You may need to manually displace the link to get it started.
13. If the balancing control is not able to catch the link when it passes through the equilibrium, stop the link yourself and bring it to the equilibrium. It should switch to the balancing controller and hold it there. DO NOT let the link wrap up the cable too tightly. This can damage the cable.
14. You have just performed your first control with the Mechatronics Kit.

15. Now power OFF the system and go onto the Software installation.

2. Sample Controllers

We have supplied you with a number of different example programs to control the Mechatronics Control Kit. Below is a list of these controllers and their source file names.

1. **Power On Flash Program**. This program runs when you power on or hardware reset the Mechatronics Control Kit. Three different controllers can be run with this example and the configuration of the DIP switches of SW1 on the C6713DSK determines which controller to run.

DIP_0	***DIP_1***	***DIP_2***	
UP	**UP**	**UP**	**Swing Reaction Wheel to Balancing point**
DOWN	**UP**	**UP**	**Swing Pendubot experiment to TOP position**
UP	**DOWN**	**UP**	**Swing Pendubot experiment to MID position**
DOWN	**DOWN**	**UP**	**Swing-Up Furuta Pendulum**

All three of these controllers require the links to be at rest before the control algorithm can begin. Tapping on DIP_3 in all three cases starts the control algorithm. The source code for this program can be found in the bootcode directory. Read the source code and readme.txt file to familiarize yourself with this example.

2. **DSP/BIOS II Example Program**. This example requires the use of both Code Composer Studio and Visual Basic 6.0. (Visual Basic 6.0 is not required but highly recommended for ease of use). With this example, you use CCS to download the example "balrtdx.out" file to the 6713DSK board. Then you use the Visual Basic "plug-in" to start the different controllers and to download gain settings. The plug-in also adds a mechanism for saving response data in a Matlab M-file format. This is the perfect environment for testing and tuning different control algorithms. With this example, five different controllers can be selected and run from the VB plug-in
    1. Swing Pendubot experiment to the MID position and balance it there.
    2. Swing Pendubot experiment to the TOP position and balance it there.
    3. Swing Reaction Wheel experiment to the inverted position and balance it there.
    4. Swing the Furuta Pendulum to the inverted balancing position and balance it there.
    5. PI speed controller for the DC Motor with attached fly wheel

    These example files are found in the dspbiosii/rtdxproj directory.
3. **MathWorks Real-Time Workshop** examples. Along with DSP C source code examples, we have also supplied Real-Time Workshop examples that run with either WinCon from Quanser or Windows Target from MathWorks. To run the examples using Wincon you will need the software Matlab/Simulink 6 or higher, Real-Time Workshop toolbox, Microsoft Visual C++ 6.0 or Microsoft Visual Studio .NET 2002-2003, and Wincon 3.0 or higher. To run the examples using Windows Target you will need the software Matlab/Simulink 6.0 or higher, Real-Time Workshop toolbox, Microsoft Visual C++ 6.0 or Microsoft Visual Studio .NET 2002-2003, and Real-Time Windows Target 2.0 or higher.

The following controller files are supplied

1. swpendmid.mdl Swing the Pendubot from its hanging down position to the MID equilibrium point and balance it there.
2. swpendtop.mdl Swing the Pendubot from its hanging down position to the TOP equilibrium point and balance it there.

3. swpendtopobs.mdl Swing the Pendubot from its hanging down position to the TOP equilibrium point and balance it there. The balancing controller implements a linear observer to estimate the link velocities.
4. swIwhlE.mdl Swing the Reaction Wheel from its hanging down position to the inverted position and balance it there.
5. swIwhlEobs.mdl Swing the Reaction Wheel from its hanging down position to the inverted position and balance it there using a full order observer to estimate the velocities of the links.
6. swfurutaEnergy.mdl Swing the Furuta pendulum from its hanging down position to it inverted position and balance it there. Swing up uses an energy based algorithm.
7. swfuruta.mdl Swing the Furuta pendulum from its hanging down position to its inverted position and balance it there. Swing up uses a simple PD controller.

3. Parts List

1. 24Volt DC Motor with 1000 Cnt/Rev Optical Encoder from Pittman Inc. Special order part number 8222D116. See Pittman website for specifications at www.pittmannet.com/pdf/lcm_bulletin.pdf.
2. 1000 Cnt/Rev Optical Encoder from US Digital Inc. Part number S1-1000-B. See U.S. Digital website for specifications at www.usdigital.com/products/s1s2/.
3. C6713DSK from Texas Instruments, which includes DSK board, 5VDC at 5AMP desktop power supply, USB cable and Code Composer Studio software installation CD. Part number TMDS320006713. See www.ti.com for more information.
4. 24VDC at 2.1AMP Power Supply from ELPAC Power Systems. Model W4024. You can find this part at DigiKey. www.digikey.com.
5. 20Char X 2Row LCD Screen from Matrix Orbital. Part number LK202-25 www.matrixorbital.com
6. C6XDSK_DigIO daughter built by Quanser
7. “Quanser Standard” Interface / PWM Amplifier board built by Quanser
8. LCD cable built by Quanser
9. US Digital Optical Encoder Cable built by Quanser
10. PWM Amplifier signal cable built by Quanser
11. Aluminum pieces all built by Quanser
    a. Bottom Plate
    b. Eight 2.75 inch standoff posts
    c. Base Plate
    d. Plastic Top
    e. Two LCD mount pieces
    f. Motor or Optical Encoder Mount
    g. Pendubot Link-1 piece
    h. Pendubot Link-2 piece with 4 8-32 nuts
    i. Pendubot Link-2 coupler piece
    j. Reaction wheel Link piece
    k. Reaction wheel Fly Wheel
    l. Four extension post for the Reaction wheel setup
    m. Furuta base piece
    n. Furuta upright piece
    o. Furuta Link-1
12. Installation CD for Kit example code

4. Additional Items

REQUIRED ITEMS

1. Pentium II or higher PC with free USB port running Windows 2000/XP.
   Optional: If using the DSP board for data acquisition, then also need a free parallel port on the PC.
2. Clean sturdy table that allows the rubber feet to get a good grip on the table.
3. Following tools:
   a. ½ inch socket
   b. 7/64, 3/32 and 5/64 inch Allen wrenches
   c. Small Phillips screw driver
   d. Small Standard Flat screw driver (i.e. 3mm precision screw driver)
4. Scotch Tape for taping the routed cables in place
5. Dedicated Power Strip for the 24V and 5V power supplies.

RECOMMENDED ITEMS

1. Microsoft Visual Basic 6.0 to use the Visual Basic Interface.
2. Matlab with Simulink and Real-Time Workshop (RTW) and either the Quanser WinCon real-time software (preferred) or Mathworks Real-Time Windows Target.
3. To use either the Quanser Wincon 3.0 (or higher) software or Mathworks Real-Time Windows Target 2.0 (or higher), the following software needs to be installed:
   a. Matlab 6.0
   b. Simulink
   c. Real-Time Workshop toolbox
   d. Microsoft Visual C++ 6.0 or Microsoft Visual Studio .NET 2002-2003

Chapter 3 Software Installation

The Mechatronics Control Kit is supplied with source code of all programs, including example controllers, Matlab/Simulink programs and a Visual Basic GUI plug-in. In addition, the Code Composer Studio software (CCS) from Texas Instruments is included for developing and running your own application programs on the DSK. CCS v3.00 requires that you have a Windows 2000/XP PC with 500MB of hard disk space and 128 MB of RAM.

1. Installing CCS under Windows 2000/XP

Before installing Code Composer Studio, make sure that you have assembled your Mechatronics Control Kit and run the flash memory demo program according to the instructions in the Getting Started Section of this manual. This will ensure that the DSK is functioning properly and is ready to be connected to the PC.

Locate the Code Composer Studio for C6713 DSK Quick Start Installation Guide supplied with the TI CCS and follow the instructions to connecting the PC to the DSK and install the CCS software. Make sure you run the C6713 DSK Diagnostic Utility to verify that the connection between the PC and the DSP can be made.

2. Installing the Mechatronics Kit Software

The Mechatronics Kit software is supplied on a single CD. Locate the CD labeled *Quanser Mechatronics Kit Software* and insert it in your CD drive. Directories and files may be copied from the CD as needed or all at once to a directory on your hard drive (i.e.: c:\mechkit\).

Chapter 4 Software Description

There is quite a bit of software to digest in this mechatronics kit. This section will give you an overview of the Code Composer Studio software package and the example source files supplied with the kit. Two CDs come with the Mechatronics Control Kit. The CD entitled "Code Composer Studio ™ IDE" is supplied by Texas Instruments with the C6713 DSK and it contains the DSK development and interface software and Code Composer Studio. The "Getting Started" section earlier in this manual discussed installing this software on your PC. See this section for a discussion of installing this software.

The Mechatronics Control Kit CD contains all the example code for the four experiments. Below you will find a breakdown of all the supplied code. We make a point to supply you with all our source code in order that you have all the tools to develop your own special controllers. This also allows you to expand the kit to control your own developed experiments. The "Software Installation" section discusses the installation instructions for this CD.

1. Code Composer Studio and TI example code

The Code Composer Studio IDE (which includes an assembler and C compiler for the C6000 DSP) is a development environment that allows you to create projects, create and edit source files, compile source files into a downloadable COFF .out file, download programs to the DSK target and debug your programs with breakpoints, watch windows and RTDX real-time monitoring "plug-ins". Code Composer Studio is a very powerful software package and this manual will not go into all the details of this package. The Code Composer Studio IDE CD already contains an extensive amount of documentation and tutorials to get you up to speed quickly with the software. However, we will try to pass some of our experience with CCS onto you in this section and give a brief introduction to some well-deserving examples that are supplied with CCS.

Recommended Reading and Tutorials for CCS

1. Before starting with CCS, read the "General Help" topics found under the "Help" menu. There is a lot of information here so initially skim through the material mainly just to know the location of the different help topics so that you can come back to that section as you get increasingly familiar with the software. The TMS320C6713 DSK discusses the DSK board, which is the "motherboard" for the mechatronics kit DSP system.
2. The "C6713DSK Tutorial" found in the "C6713 DSK Help" files (i.e. c6713dsk.hlp).
3. Manuals found in the "Help->User Manuals" menu in CCS. These are the manuals that are the most useful when developing on the DSK.
    a. Code Composer Studio IDE Getting Started Guide (SPRU509)
    b. TMS320C6000 DSP/BIOS User's Guide (SPRU423)
    c. TMS320C6000 DSP/BIOS Application Programming Interface (API) Reference Guide SPRU403)
    d. TMS320C6000 Peripherals Reference Guide (SPRU190)
4. Tutorials under the "Help->Tutorials" menu in CCS are excellent for getting you started in CCS. We recommend at least going through the Code Composer Studio Tutorial before working with our examples and developing your own projects.

Tips when using CCS

1. To keep the USB port interface and C6713 DSK synchronized there are two good practices to get into the habit of performing when running CCS connected to the DSK.

a. Never press the hardware-reset button on the DSK board when running CCS. Doing so can interrupt communication with CCS and you will probably need to exit CCS and restart it to regain communication.
b. Periodically before loading or re-loading an *.out file, first run the "Reset CPU" command. Debug->Reset CPU.

2. If you lose communication with the DSK you will need to EXIT CCS and before re-launching CCS you will probably need to power ON and OFF the DSK system.
3. Note that when CCS is loaded the hardware RESET button on the DSK board is disabled. Pressing the RESET button still does perform some reset functions but does NOT cause the DSK to boot from its flash memory. You should not use the hardware RESET button when CCS is connected to the DSK.
4. Note also that when you exit CCS the hardware reset button is still deactivated. You will need to power ON and OFF the DSK to re-enable the RESET button.

After installing CCS your PC should have a directory structure similar to the following:

CCStudio_v3.1
|-------- BIN: *Various applications and utilities such as flashburn.*
|-------- C6000:
| |-------- BIOS: *Include, src and lib files for DSP/BIOS II code*
| |-------- CGTOOLS: *C/Asm Compiler binaries, include files and libraries*
| |-------- CSL: *Include and lib files.*
| |-------- DSK6713*: Files pertaining to the C6713DSK*
| | |-------- INCLUDE:
Various include files for the HOST side programs that are used to communicate with the codec, DIP switches, LEDs, and flash on the C6713DSK.
| | |-------- LIB: *Library file for the DSK6713.dll*
| |-------- RTDX: *Include and Library files for RTDX functionality on the TARGET*
| |-------- XDAIS: *eXpressDSP Algorithm Standard examples.*
|-------- C6200: C62x *DSP and Imaging Signal Processing Library.*
|-------- C6400: C64x *DSP and Imaging Signal Processing Library.*
|-------- CC: *CCS executable and GEL files.*
|-------- DOCS: *All documentation for CCS 2.0*
|-------- DRIVERS: *Drivers needed for communicating over the USB to the DSK in CCS*
|-------- EXAMPLES: *A Wealth of examples for CCS and the DSK*
| |-------- DSK6211: *Examples for DSK6211.*
| |-------- DSK6711: *Examples for DSK6711*
| |-------- DSK6713:
| | |-------- BIOS: *Examples for using many of the different DSP/BIOS II features*
| | |-------- BSL
| | | |-------- DSK_APP:
Illustrates how to use CODEC.
| | | |-------- LED:
Turn on an LED using BSL.
| | | |-------- LEDPRD:
Same as LED example but using a DSP/BIOS thread.
| | | |-------- POST:
TARGET source files for the Power-on Self Test program supplied by TI.

| | | |--------TONE:
Illustrates how to use CODEC.
| | |--------CSL: *Examples using CSL API.*
| | |--------RTDX:
Some examples in using RTDX to transfer data to and from other Active X clients.
| |--------EMU64XX: Examples for DSK EMU 64XX hardware.
| |--------EVM6201: Examples for EVM 6201/6701 hardware.
| |--------EVMDM642: Examples for DM642 Evaluation Module (EVM).
| |--------HOSTAPPS: *RTDX Host examples*
| |--------SIM55XX: C55XX simulation examples that do not require the DSP.
| |--------SIM62XX: C62XX simulation examples that do not require the DSP.
| |--------SIM64XX: C64XX simulation examples that do not require the DSP.
| |--------TEB6416: Examples for DSK 6416 hardware.
|--------MYPROJECTS: *Default directory for projects in CCS*
|--------PLUGINS: *All DSP/BIOS II plugins*
|--------SDKv3.0: *Software development kit.*
|--------SPECDIG: Contains the Spectrum Digital driver files for the C6713 DSK.
|--------TUTORIAL: *All project and source for the CCS Tutorials. Code in DSK6713 directory*
|--------UNINSTALL: *CCS uninstall information files.*

With the RTDX communication, your program must be written using DSP/BIOS II. Data is sent and received from the DSP by the use of DSP/BIOS II idle tasks. This way large data arrays can be sent too and from the DSP without halting the processor for long periods.

Take a little time to browse through the different directories installed with CCS. Read the readme.txt files in many of the directories to give you an idea of what the files located in those directories perform.

Power On Self Test (POST)

If you had purchased a C6713DSK directly from TI, it would have come to you installed with the "POST" source code programmed on its flash. This POST software checks the different peripherals available on the DSK, i.e. SDRAM, CODEC, Internal RAM, LEDs. We reprogram the flash with our "bootcode" program before we send the system to you. Therefore, if you would like to change the system to power on with the POST program you will have to re-flash the DSK. See the "\CCStudio_v3.1\examples\dsk6713\bsl" directory for more details.

2. Quanser Source Code and Documentation

This section gives an overview of the source code supplied on the Mechatronics Control Kit CD. This CD contains the software that we have developed specifically for the Kit. Our main documentation for these examples is in the source files (*.c and *.h). We have commented this code quite extensively so after reading through this manual you should also read the source files for greater detail.

ROOT: *Source code supplied with the Mechatronics Kit*
- DSPBIOSII: *DSP/BOIS II example for controlling the five experiments.*
  - readme.txt
  - RTDXPROJ *Example RTDX project with Visual Basic PlugIn*
    - readme.txt
    - DSP *Source code for the DSP portion of the project*
      - balboth.c
      - cc_build_Custom.log
      - balrtdx.cdb
      - balrtdxcfg.s62
      - balrtdxcfg.cmd
      - balrtdxcfg.h62
      - balboth.obj
      - c6xdskdigio.obj
      - balrtdxcfg.obj
      - balrtdx.pjt
      - balrtdx.map
      - balrtdx.out
      - balrtdx.paf
      - balrtdxcfg_c.c
      - balrtdxcfg_c.h
    - VB *Visual Basic Plugin Source Code*
      - RTDXfrm.frm
      - RTDXproj.vbp
      - RTDXproj.vbw
      - RTDXproj.exe
      - SETUP
        - All setup files for use on PCs without Visual Basic
- INCLUDE: *Common Include files for example source code*
  - boot.asm
  - c6x11dsk.h
  - c6xdskdigio.h
  - c6xdskdigio.c
  - rts6701.lib
  - switch_led.c
  - switch_led.h
- BOOTCODE: *Source example that can be flashed to the DSK, Default Flash Program*
  - readme.txt
  - bootflsh.c
  - lnk.cmd

```
|       |       myvectors.asm
|       |       bootflsh.pjt
|       |       bootflsh.out
|       |       bootflsh.map
|       |-------FLASH:  Files and Instructions for re-flashing the DSK
|               |       FBTC6713.out
|               |       flashconf.cdd
|               |       flashfile.hex
|               |       flashit.bat
|               |       hex.cmd
|               |       hex.map
|               |       LogFlashBurn.txt
|-------MATLAB:  Matlab M-files used to identify and form the models of the two experiments
|       |       readme.txt
|       |       id_pendubot.m
|       |       id_furuta.m
|       |       LINPENDU.M
|       |       linRwhl.m
|       |       id_reaction_wheel.m
|       |-------RTW:  Files needed to run the Mechatronics Kit using Real-Time Workshop
|       |       |       readme.txt
|       |       |-------WINCON:  Files to setup Quanser WinCon with the Mech Kit
|       |       |       |       readme.txt
|       |       |       |       c6xbd.h
|       |       |       |       c6xen.dll
|       |       |       |       c6xen.h
|       |       |       |       c6xen.tlc
|       |       |       |       c6xlib.mdl
|       |       |       |       c6xpwm.dll
|       |       |       |       c6xpwm.h
|       |       |       |       c6xpwm.tlc
|       |       |       |-------EXAMPLES:  Example External Simulink models for WinCon
|       |       |       |       |       swpendtop.mdl
|       |       |       |       |       swfuruta.mdl
|       |       |       |       |       swfurutaEnergy.mdl
|       |       |       |       |       swpendtopobs.mdl
|       |       |       |       |       swpendmid.mdl
|       |       |       |       |       swIwhlE.mdl
|       |       |       |       |       swIwhlEobs.mdl
|       |       |       |       |       goodRWhlobs.mat
|       |       |       |       |-------HOWTO_DLLS:  How to compile the mex (dll) files
|       |       |       |               |       readme.txt
|       |       |       |               |       c6xen.c
|       |       |       |               |       c6xpwm.c
|       |       |       |               |       c6xbd.h
|       |       |       |               |       c6xen.h
|       |       |       |               |       c6xpwm.h
|       |       |       |               |       DataTypes.h
```

```
|       |       |               |       hardware.h
|       |       |-------- WINDOWSTARGET:  Files for MathWorks WindowsTarget
|       |       |       |---------WINDOWSTARGET2.0-2.2: Files for WindowsTarget 2.0-2.2
|       |       |       |
|       |       |       |       readme.txt
|       |       |       |       c6xbd.h
|       |       |       |       c6xen.dll
|       |       |       |       c6xen.h
|       |       |       |       c6xen.tlc
|       |       |       |       c6xlib.mdl
|       |       |       |       c6xpwm.dll
|       |       |       |       c6xpwm.h
|       |       |       |       c6xpwm.tlc
|       |       |       |-------- EXAMPLES:  Example External Simulink model for WTarget2.0
|       |       |       |       |       swpendtop.mdl
|       |       |       |       |       swpendtopobs.mdl
|       |       |       |       |       swpendmid.mdl
|       |       |       |       |       swIwhlE.mdl
|       |       |       |-------- HOWTO_DLLS:  How to compile the mex (dll) files
|       |       |               |       readme.txt
|       |       |               |       c6xen.c
|       |       |               |       c6xpwm.c
|       |       |               |       c6xbd.h
|       |       |               |       c6xen.h
|       |       |               |       c6xpwm.h
|       |       |       |---------WINDOWSTARGET2.5: Files for WindowsTarget 2.5
|       |       |       |       readme.txt
|       |       |       |       c6xbd.h
|       |       |       |       c6xen.dll
|       |       |       |       c6xen.h
|       |       |       |       c6xen.tlc
|       |       |       |       c6xlib.mdl
|       |       |       |       c6xpwm.dll
|       |       |       |       c6xpwm.h
|       |       |       |       c6xpwm.tlc
|       |       |       |-------- EXAMPLES:  Example External Simulink model for WTarget2.0
|       |       |       |       |       swpendtop.mdl
|       |       |       |       |       swpendtopobs.mdl
|       |       |       |       |       swpendmid.mdl
|       |       |       |       |       swIwhlE.mdl
|       |       |       |-------- HOWTO_DLLS:  How to compile the mex (dll) files
|       |       |               |       readme.txt
|       |       |               |       c6xen.c
|       |       |               |       c6xpwm.c
|       |       |               |       c6xbd.h
|       |       |               |       c6xen.h
|       |       |               |       c6xpwm.h
|       |-------- SIMULINK:  Example simulation files for the two experiments using Simulink
```

| | readme.txt
| | pendubot_blocks.mdl
| | swIwhlsimE.mdl
| | swIwhlsimEobs.mdl
| | swpendmidsim.mdl
| | swpendtopsim.mdl
| | swpendmidsimobs.mdl
| | swpendtopsimobs.mdl
| | pendubotanim.m
| | Iwhlanim.m
| | howto.mak
| | msvc60opts.bat
| | parfdbk.c
| | parfdbk.dll
| | pendubot.c
| | pendubot.dll
| | pend_sfunc_exmpl.mdl
| | furutaanim.m
| | goodfurutaobs.mat
| | goodIwhobs.mat
| | swfurutasim.mdl
| | swfurutasimobs.mdl
| |-------- LINUXINCONTROL: *Method for controlling the Mechatronics Kit in Linux and Simulink. See the readme.txt file in this directory for more information.*
|-------- DOCS: *Mech Kit documentation and Board Schematics*
| | kitmanual.pdf *PDF version of the Mechatronics Control Kit manual*
| | kitmanual.doc *Microsoft Word version of the Mechatronics Control Kit manual*
| |-------- DATASHEETS *Datasheets for different components and chips of the Mechkit.*
| | | pittmanmotor.pdf
| | | LCD_LK202-25.pdf
| | | S1 & S2 Data Sheet.pdf
| | | LS7266R1 Data Sheet.pdf
| | | LS7266R1 Manufacturer's Data Sheet.pdf
| | | CTS8254.pdf
| | | 3952.pdf
| |-------- SCHEMATICS *Schematics of C6xDSK_DigIO daughter card*
| | board_schematics.pdf
| |--------PAPERS *This folder contains several tutorials and papers describing the Mechatronics Control Kit.*

3. Bootcode

The code that runs when the DSK system is first power ON is the source code found in the “bootcode” directory. There are four source files that make up this project: bootflsh.c, myvectors.asm, boot.asm and c6xdskdigio.c (found in the “mechkit\include” directory). Bootflsh.c is a more classical DSP program that does not use the DSP/BIOS II operating system. Myvectors.asm sets up the interrupt vector memory locations for the program and boot.asm is the initial boot code that copies the code from flash to SDRAM on reset. C6xdskdigio.c has all the functions for communicating to the C6xDSK_DigIO daughter card. (A NOTE:

DSP/BIOS II programs can also use “boot.asm” to allow them to be programmed on the flash. DSP/BIOS II programs do take a bit large memory footprint so we chose to go with a “classical” (no operating system) program for our flash routine so it had more room to expand if needed in the future.)

On reset the bootflsh is run. Bootflsh first reads the state of the three DIP switches (0-2) on SW1 of the DSK board. Depending on the state of the switches, five possible modes can be run. Below are the modes corresponding to the state of the switches.

DIP_0	*DIP_1*	*DIP_2*	
UP	**UP**	**UP**	**Swing Reaction Wheel Experiment to Balancing point**
DOWN	**UP**	**UP**	**Swing Pendubot experiment to TOP position**
UP	**DOWN**	**UP**	**Swing Pendubot experiment to MID position**
DOWN	**DOWN**	**UP**	**Swing Furuta Pendulum to the inverted position**
UP	UP	DOWN	Presently NO function
DOWN	UP	DOWN	Presently NO function
UP	DOWN	DOWN	Presently NO function
DOWN	**DOWN**	**DOWN**	**Puts the system into "dummy" data acquisition mode**

After reading the USER_SWITCHES the program prints a message to the LCD screen indicating which mode it is in. (**NOTE**: The boot code will actually hang here at the print to the LCD if jumper J8 is not installed on the C6xDSK_DigIO daughter card. That jumper is required for any LCD print function.) If the mode is the “dummy data acquisition mode”, the DSP immediately starts monitoring the parallel port connector on the daughter board for commands from the HOST PC (see the RTW section for more details). If the mode is one of the three swing up and balance modes the program sits and waits for the user to tap on DIP_3. All three of the swing-up modes require that the linkage be at rest to start. This allows the program to initialize the optical encoders to a zero position (or home) before starting the control algorithm. Once the user switched the PWM Amps ENABLE switch to ON and taps DIP_3 the control algorithm starts and attempts to swing the linkage to an equilibrium position and catch and balance it there. The LCD will also be displaying status information on the control variables for encoder position #1, #2, and the control effort being applied to the motor. See Figure 4-4-1 for a description of the LCD displayed information.

Figure 4-4-1: LCD Display Panel

If the swing up algorithm was not able to swing the unactuated link to the balancing position you can manually move the link to the balancing position and the control will switch and balance the link there. See the tuning section for more information on getting the system to swing-up correctly.

If you would like to re-run the same controller, simply move the PWM Amp ENABLE switch to OFF and depress the hardware RESET button, and the program will start over. Wait for the links to be at rest and then ENABLE the Amp and tap on DIP_2. To switch to a different mode, again DISABLE the Amp, switch DIP_0 and DIP_1 to the desired mode and then press the hardware-reset button. The new mode will then run and you can enable the amp and tap DIP_2 to start this controller.

The bootflsh.out file does not only have to be run from the flash. You can in fact download and run bootflsh.out from CCS. When done in this fashion, the boot.asm code is ignored. This way you can modify the bootflsh source code and debug it using CCS. Then when your final version is complete you can flash the program to the DSK. See the files in the "FLASH" directory for information on re-flashing the DSK.

4. DSP/BIOS II code and Visual Basic Interface Plugin (BALRTDX and RTDXPROJ)

This section describes the DSP/BIOS program found in the DSPBIOSII\RTDXPROJ\DSP directory along with its companion Visual Basic program found in the DSPBIOSII\RTDXPROJ\VB directory. The target program must be loaded and run from inside CCS and then commanded to start the appropriate control algorithm from within a Visual Basic plug-in called RTDXPROJ. This means that in order to run this example code you will need Visual Basic Version 6.0 installed on your PC. Actually, it is not required, but we highly

recommended that you purchase VB V6.0. The VB programming language is very easy to learn and can be used to build very impressive and powerful graphical user interface programs. Students pick up the VB programming language very quickly. If you choose not to install Visual Basic V6.0 on your PC you can use the setup.exe file located in the DISPBIOSII\RTDXPROJ\VB\SETUP directory on the Mechatronics Control Kit CD to install the needed *.dll and *.ocx files to run the plug-in. The main documentation for this example is found in the source files for both the DSP TARGET program along with the Visual Basic plug-in source files. Please read the source files for all the details that this section does not cover.

Steps for Running the RTDX DSP/BIOS II example

1. Make sure the USB on the DSK is connected to the PC.
2. Power On the DSK system and Press the Hardware Reset button. The power on code should be running and a message should have been printed to the LCD screen. If you have just finished building one of the experiments, you may want to use the power on code to test that the experiment has been built properly. See the bootcode section above and the bootcode.c source file for information on operating the power on code.
3. Now launch CCS and connect to the DSK by selecting “Connect” in the “Debug” menu. The LEDs on the DSK should stop blinking when CCS has finished loading.
4. Run the GEL file “QuickTest” by selecting the menu “GEL”, clicking on “Check DSK”, and choosing “QuickTest”. Verify the “Target OK” message.
5. Open the project file “balrtdx.pjt” in the folder “dspbiosII\rtdxproj\dsp\”.
6. Software Reset the DSP by selecting Debug->Reset CPU from the menu. When running Reset CPU you should wait for the reset to complete before running any other commands. So after running the Reset CPU command watch the LEDS on DSK board. When they finish cycling, the DSP has finished the Reset function.
7. Load the out file “balrtdx.out” in the folder “dspbiosII\dsp\balrtdx\”. The download takes a few seconds.
8. Run the DSP program by clicking on Debug->Run in the menu. All three LEDs should start blinking on and off and the LCD Screen should print a message indicating that the DSP is waiting for a command from the VB plugin.
9. Launch the VB application “rtdxproj.exe” found in “dspbiosII\rtdxproj\vb\”. The GUI interface similar to Figure 4-2 should appear.

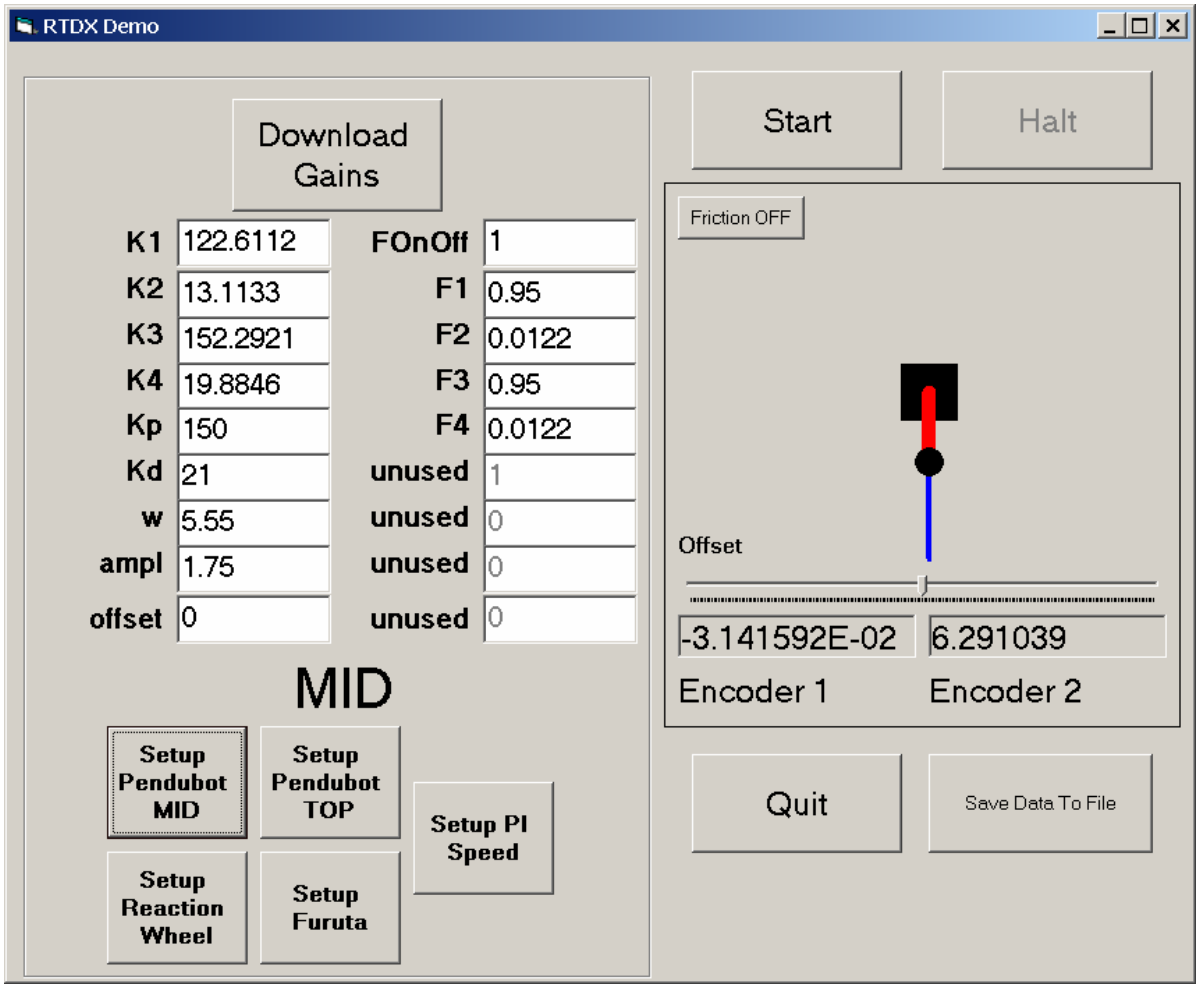


Figure 4-2 Visual Basic GUI

10. Enable the PWM Amp to ON (the LED should be bright red).
    a. If you want to run the "Mid" swing-up controller for the Pendubot experiment, you can simply click the "Start" button in the VB plug-in. This will however use the default gains that are hard coded in the balboth.c source file. If you would like to use the gain values found in the edit boxes of the VB application click on the "Download Gains" button to use those gains. Then press the "Start" button to start the control.
    b. To run the "Top" controller for the Pendubot experiment first click on the "Pendubot Top Gains" button. Then to download those gains click on "Download Gains". Click "Start" to start the controller
    c. To run the reaction wheel experiment's swing-up algorithm first click on the "Reaction Wheel Gains" button. Then to download those gains click on "Download Gains". Click "Start" to start the controller.
    d. To run the Furuta pendulum experiment's swing-up algorithm first click on the "Furuta Gains" button. Then to download those gains click on "Download Gains". Click "Start" to start the controller.

Note: The default control gains loaded for the various experiments in the Visual Basic Interface are not always the same as the control gains used on the flash memory or in the bootflsh.c program.

11. While the controller is running angular position (or velocity data) data is updated in the display edit boxes and the same data is used to update the linkage animation.
12. To Halt the Controller, click the “Halt” button.
13. The “Download Gains” button can be clicked both when the controller is and is not running. You can play around with changing gains on the fly to see how the experiment responds. The “offset” gain is a good gain to play with on the Pendubot experiment. Adjusting this gain causes the Pendubot to move to a new balancing point. In this way, you can compensate for offset torques created by the link cables or just move the Pendubot to equilibrium.
14. After running a swing-up routine, you can save stored data values to a file. You have to be in the “Halt” state for this button to become active. Simply click this button and it will ask you for a file to save the data in. When the data upload is complete, the other buttons will become active again. This data had been stored on the DSP during the swing up run. See the source code for modifying the data to be saved and uploaded.
15. To quit the application first make sure you are in the “Halt” state. Then click on “Quit” and the VB plug-in should quit. Then, go back to CCS and halt the DSP program by selecting Debug->Halt in the menu.

RTDX transfer rates will vary depending on the speed of the PC you are running CCS on. The DSP program is presently programmed to send data to the RTDX upload channel (data, see balboth.c) at a rate of 10Hz. On slower PCs, this may be too fast and the data displayed on the VB program will lag the actual data. In this case, you may need to adjust the upload rate. The comments in both the VB project file and the source file balboth.c explain how to adjust the upload rate.

5. Matlab M-files

We have supplied you with five M-files to generate the linear and non-linear models for the Pendubot, Furuta and reaction wheel experiments. Run the id_furuta.m, id_pendubot.m and id_reaction_wheel.m files to calculate the parameter values of the respective experiments. These values are found by using mass and length measurements of the different parts. There are comments in the files that state how each measurement was arrived at. Then use the linpendu.m and linIwhl.m files to find the linear models for the Pendubot and reaction wheel experiments at the enter equilibrium value. Type “help linpendu” or “help linIwhl” at the Matlab prompt for information on using these functions. See the dynamic modeling and control sections for more information on the models of these plants and the definition of the parameter values spoke of here.

6. Simulink Simulation files

We have supplied you with a number of Simulink simulation files. There are also animation M-files that are used to animate single and double pendulum systems. The files swfurutasim.mdl, swIwhlsim.mdl, swpendmidsim.mdl, swpendmidsimobs.mdl, swpendtopsim.mdl and swpendtopsimobs.mdl all use standard Simulink blocks to simulate the non-linear equations of motion of the linkages and the controller algorithm. The *.mdl files with “obs” in their name implement a linear observer to estimate the velocities of the links when in the balancing position. The remaining *.mdl file, pend_sfunc_exmpl.mdl, demonstrates how to use Simulink S-functions to simulate both the non-linear equations of motion and control algorithm. The file “howto.mak” describes how to compile the S-functions.

7. Real-Time Workshop

The Mechatronics Kit comes equipped to support the Real-Time Workshop toolbox using both the WinCon from Quanser (www.quanser.com) and the Windows Target Toolbox from Mathworks (www.mathworks.com). The way Real-Time Workshop is supported is by turning the DSP system into a simple data acquisition board or as we also call it a "dummy data acquisition board". What we mean by this is the DSP is not doing any control calculations. Its only purpose is to monitor commands from the host PC and receive or send data to the PC. That means that all the control calculations are done on the host PC. So what we are supplying you with in this mode is a data acquisition card that plugs into your PC's parallel port.

To activate this mode on the DSP system you will need to have the bootcode program flashed on the DSKs flash chip. Power off the DSK and connect the parallel port connector on the C6xDSK_DigIO daughter card into the PC's parallel port. Then switch all the USER_SWITCHES to the DOWN position and power on the DSK. On reset, the LCD screen should print a message indicating that the system is in dummy data acquisition mode. Now in this mode you are ready to run your RTW controller files. Four example *.mdl files have been supplied to get you started.

Two "dos device driver" blocks are needed to communicate to the DSK system, an encoder block and a PWM block. These two blocks can be found in the model library file c6xlib.mdl. Read the readme.txt files in both the Wincon directory and either the WindowsTarget 2.0-2.2 or WindowsTarget 2.5 directory for detailed instructions on how to setup Matlab and RTW to work with the given drivers. Ensure the WindowsTarget directory chosen corresponds to the WinRT version in your Matlab. The readme.txt does not go into the details of the WinCon and Windows Target software. You will need to use the help files from those software packages to get yourself up to speed on RTW. One thing to note about RTW is that usually the hardest thing to get setup on your PC is the correct compiler variable settings because WinCon and Windows Target call the C compiler using command line calls. For RTW and WinCon or Windows Target to build your projects properly you need to have all the environment variables set correctly for your compiler. Both WinCon and Windows Target do a good job of automating the setup process but errors can still occur. So, if you have problems compiling your real-time models, investigate your environment variable settings to see if there are any errors.

8. Tuning the controller gains

We supply you with non-linear controllers to swing the linkages to the equilibrium points but you will find that swing up gains will need to be adjusted when working with the systems.

The goal of each control algorithm is to use a non-linear controller to swing the linkage to an equilibrium position and when/if the linkage arrives at the equilibrium position switch to a linear balancing controller to stabilize the linkage at that unstable equilibrium. The following sections will describe how to design an example linear controller for the experiment and discuss tuning the given non-linear controller to swing the linkage to the equilibrium.

Also note in your controller designs that the sample rate of the given example controllers is run at 200 Hz (5ms period). You can design your controllers assuming that the system is continuous but to be a bit more precise you will want to perform your designs in the discrete domain.

1) PENDUBOT – MID:

Follow these instructions to tune the controller that swings-up link 2 of the pendubot into the "mid" position:

1. To design a linear full-state feedback controller for the Pendubot at the "Mid" equilibrium point (-pi/2,0,pi,0) we have supplied two Matlab M-files to perform the parameter identification and the linearization of the non-linear equations about the equilibrium point.
    a. Run the script file "id_pendubot". This produces a Theta parameter vector.
    b. Run the "linpendu" function as follows: [A,B,ur] = linpendu(Theta,-pi/2,pi). This will produce the linearized A and B matrices.
    c. Now with this linear model, you can design a linear controller to stabilize the system. As an example, find a full state feedback controller using the "place". (i.e. K = place(A,B,[-13.6 + 7.2i, -13.6 - 7.2i, -8.75 + 1.45i, -8.75 - 1.45i]))
2. You will find that the encoder cable for link-2 and the friction of the motor can cause some offset problems for different control designs. The offset problem is that the controller tends to balance the unactuated link offset from the desired equilibrium position. These offset errors are more pronounced with lower gain controllers. We have added an open loop offset voltage parameter to our control algorithm to adjust for this problem somewhat, but with low gain controllers it does not always solve the problem. Routing the cable with minimum disturbance to link 2 is a key factor in reducing this offset also. While the controller is balancing the Pendubot play around with the cable to see how it affects the control.
3. Our swing-up controller implements a partial feedback linearization controller. See the controller section and given references for more details on this control algorithm. There are four gain values that you can tune to get the second link to swing to the MID equilibrium point slowly enough so that the balancing controller can catch and stabilize the system. They are:
    - Outer-loop proportional gain *Kp*
    - Outer-loop derivative gain *Kd*
    - Swing-up trajectory amplitude *ampl*
    - Swing-up trajectory frequency *w*

    The amplitude and frequency determine the initial pump trajectory for the swing up control. If your goal is to experiment with the non-linear partial feedback linearization controller then feel free to adjust all four of the gain values and observe how the control reacts.

    However, if your main goal is to just get the Pendubot to swing up to the equilibrium value then we recommend that you fix three of the gain values and just adjust the "ampl" gain of the swing up trajectory as follows:
    a. Set *Kp* = 150.0, *Kd* = 21.0, and *w* at 5.55.
    b. Start with an *ampl* value of around 1.7.
    c. Ensure the links are at rest before running the control (i.e. the encoders are reset when the controller begins running).
    d. Run the swing up control and examine if the swing-up works.
    e. If the swing-up control did not add enough energy to link 2, i.e. the link did not make it up to the equilibrium point, increase *ampl* by about 0.05 and try the run again.
    f. Continue this until you find the *ampl* gain that brings link 2 to the equilibrium. If the swing-up control adds too much energy to link 2 and link 2 swings quickly through the equilibrium, reduce the value of *ampl* by about 0.05 and try the run again.

    Note that there are many gain values that will perform the swing-up but the above gains are values that we have found to work well. Also, after successive swing-up tests the motor and amplifier chip become warmer and this causes the motor torque constant to slightly decrease. As a result, this may require you to tune the *ampl* gain again to get the link to swing to the equilibrium.

CAUTION: Do not let the motor get stuck in a position say by a wrapped up cable or some other obstacle. If the motor stalls with maximum torque being applied, the motor could get very hot and become damaged. Always take care to power off you system when it is unattended so that an issue like this can be avoided. Also very high gain controllers can cause the motor to heat up. When attempting new control algorithms monitor the motor's temperature from time to time and power off if the motor gets hot.

4. Perform tuning either by using the VB plugin supplied with the DSP/BIOS II code example or by using CCS watch windows with the use of the bootcode example source files. See the readme.txt files in those respective directories for instructions.

2) PENDUBOT – TOP

Follow these instructions to tune the controller that swings-up link 2 of the pendubot into the "top" position:

1. To design a linear full-state feedback controller for the Pendubot at the "Top" equilibrium point (pi/2,0,0,0) we have supplied two Matlab M-files to perform the parameter identification and the linearization of the non-linear equations about the equilibrium point.
    a. Run the script file "id_pendubot". This produces a Theta parameter vector.
    b. Run the "linpendu" function as follows: [A,B,ur] = linpendu(Theta,pi/2,0). This will produce the linearized A and B matrices.
    c. Now with this linear model, you can design a linear controller to stabilize the system. As an example, find a full state feedback controller using the "lqdr" function. (i.e. K = lqrd(A,B,Q,100,.005); where Q = diag([.05,850,10000,0])
2. You will find that the encoder cable for link-2 and the friction of the motor can cause some offset problems for different control designs. The offset problem is that the controller tends to balance the unactuated link offset from the desired equilibrium position. These offset errors are more pronounced with lower gain controllers. We have added an open loop offset voltage parameter to our control algorithm to adjust for this problem somewhat, but with low gain controllers, it does not always solve the problem. Routing the cable with minimum disturbance to link 2 is a key factor in reducing this offset also. While the controller is balancing the Pendubot play around with the cable to see how it affects the control.
3. Our swing-up controller implements a partial feedback linearization controller. See the controller section and given references for more details on this control algorithm. There are three gain values that you can tune to get the second link to swing to the TOP equilibrium point slowly enough so that the balancing controller can catch and stabilize the system:
    - Outer loop proportional gain *Kp*
    - Outer loop derivative gain *Kd*
    - Back pump open-loop excitation value, hard coded to a value of 0.475 seconds, before swinging the linkage up to the TOP equilibrium.
4. As with the MID position, if your goal is to experiment with the non-linear partial feedback linearization controller then feel free to adjust the three gain values and observe how the control reacts.
5. However, if your main goal is to just get the Pendubot to swing up to the equilibrium value then we recommend that you fix the Kp and Kd gain values and only adjust the open loop amplitude gain. This open loop amplitude gain causes the linkage to pump backwards adding enough energy to the system so that link-2 can be swung up over link-1.
    a. Set *Kp* = 350.0 and *Kd* = 24.3.
    b. Start with a *backopenloop* value of around 5.5.

c. Ensure the links are at rest before running the control (i.e. the encoders are reset when the controller begins running)
d. Run the swing up control and see if the controller is able to catch the linkage at the TOP position. If the swing-up control did not add enough energy to link-2 (i.e. the link did not make it up to the equilibrium point) then increase *backopenloop* by about 0.1 or so and try the run again.
e. Continue the process until the *backopenloop* gain brings link-2 to the equilibrium.
f. If the swing-up control adds too much energy to link 2, causing it to swing quickly through the equilibrium, reduce *backopenloop* by about 0.1 and try the run again. When you are getting close to the link being able to be caught you will want to adjust the gain by smaller amounts then 0.1.

6. There are many gain values that can be used to swing up the Pendubot but the gains used above are values that were found to work well. Note also, that throughout theses tests the motor and amplifier chip will heat up and cause the motor torque constant to drop a bit. This may require some additional tuning to the *backopenloop* gain again to get the link to swing to the equilibrium.

> **CAUTION**: Do not let the motor get stuck in a position say by a wrapped up cable or some other obstacle. If the motor stalls with maximum torque being applied, the motor could get very hot and become damaged. Always take care to power off you system when it is unattended so that an issue like this can be avoided. Also very high gain controllers can cause the motor to heat up. When attempting new control algorithms, monitor the motor's temperature from time to time and power off if the motor gets hot.

7. Perform tuning either by using the VB plugin supplied with the DSP/BIOS II code example or by using CCS watch windows with the use of the bootcode example source files. See the readme.txt files in those respective directories for instructions.

3) REACTION WHEEL

Follow these instructions to tune the reaction wheel swing-up controller:

1. To design a linear full-state feedback controller for the reaction wheel at the "Top" equilibrium point (pi,0,0) we have supplied two Matlab M-files to perform the parameter identification and the linearization of the non-linear equations about the equilibrium point.
    a. Run the script file "id_reaction_wheel". This produces a P parameter vector.
    b. Run the "linpendu" function as follows: [A,B] = linpendu(P,pi). This will produce the linearized A and B matrices.
    c. Now with this linear model you can design a linear controller to stabilize the system. As an example find a full state feedback controller using the "lqdr" function. (i.e. K = lqrd(A,B,Q,1,.005); where Q=diag([1,0,0.0005]).
2. You will find that when the reaction wheel is balancing the inverted pendulum it normally does not spin down to zero velocity. This is due to the disturbance torque applied to the link by the encoder and motor cable. While the link is balancing you can change the bends in the cable to get the wheel spinning at a slower rate. You do not want the motor spinning too fast because then it has a smaller region of torque to apply. Torque is achieved by accelerating the flywheel and if the motor is already spinning close to its maximum speed it will not have a way to accelerate in that one direction. So routing the cable with minimum disturbance is one way to minimize this problem. You also, though, may have to tune slightly the K3 state feedback gain multiplied by the flywheel velocity state. Using theVB plugin along with the DSP/BIOS II example you can easily update the

K3 value while the balancing control is in operation. Increasing the K3 gain slightly from the gain calculated by "place" or "lqr" in Matlab will help this problem. Increasing K3 too much though will make your control unstable. Also note that as with the Pendubot experiment, low gain controllers have a harder time balancing the pendulum.

3. The swing-up control for the reaction wheel experiment is an energy/passivity based nonlinear controller. See our controller section and references along with the given source code to understand the details of the controller. There are three gains that can be tuned to adjust the swing-up. Always ensure the link is at rest before starting the controller. Zero reference for the optical encoders is found when the controller is started.
4. You will also find that as the motor and amplifier chip heat up in use, the motor torque constant will start to drop a bit.

> **CAUTION**: Do not let the motor get stuck in a position or run at its maximum speed while rubbing against an obstacle. If the motor stalls with maximum torque, it could get very hot and become damaged. Always take care to power off you system when it is unattended so that an issue like this is avoided. Also very high gain controllers can cause the motor to heat up. When attempting new control algorithms, monitor the motor's temperature from time to time and power off if the motor gets hot.

5. Perform tuning either by using the VB plugin supplied with the DSP/BIOS II code example or by using CCS watch windows with the use of the bootcode example source files. See the readme.txt files in those respective directories for instructions.

FURUTA PENDULUM

Follow these instructions to tune the controller that swings-up the Furuta pendulum:

1. To design a linear full-state feedback controller for the Furuta at the inverted equilibrium point (0,0,0,0) we have supplied a Matlab M-file to perform the parameter identification and the linearization of the non-linear equations about the up and down equilibrium points.
    a. Run the script file "id_furuta". This produces a "par" parameter vector.
    b. The "id_furuta" also outputs the linearized A and B matrices for both the inverted (up) equilibrium point and the hanging down (down) equilibrium.
    c. Using the linear model, a linear controller can be designed to stabilize the system. For example, find a full state feedback controller using the "place". (i.e. K = place(Aup,Bup,[-8,-9,-10,-11]))
2. You will find that the encoder cable for link-2 and the friction of the motor can cause a large limit cycle for different control designs. We have supplied a friction compensation algorithm to get ride of some of this limit cycle. You may want to adjust these friction gains to achieve the best performance.
3. Our swing-up controller for the Furuta pendulum implements an energy based swing up control found in the paper "Energy Based Control of Pendulum" by IWASHIRO,FURUTA and Astrom. We have supplied this paper in the directory docs\papers\furuta. There are three gain values that you can tune to get the second link to swing to the inverted equilibrium point slowly enough so that the balancing controller can catch and stabilize the system:
    - Energy proportional gain *Ke*
    - Proportional gain used for initial trajectory *Kp*
    - Saturation value *Sat*.

4. Typically, the controller does not need much tuning but if required try keeping *Ke* and *Kp* constant and adjusting the Saturation Value, *Sat*. Increasing this value will allow the control to swing up the linkage quicker. Too high of a value though will cause the link to run into the cable limit and then not work correctly.

> **CAUTION**: Do not let the motor get stuck in a position say by a wrapped up cable or some other obstacle. If the motor stalls with maximum torque being applied, the motor could get very hot and become damaged. Always take care to power off you system when it is unattended so that an issue like this can be avoided. Also very high gain controllers can cause the motor to heat up. When attempting new control algorithms, monitor the motor's temperature from time to time and power off if the motor gets hot.

5. Perform tuning either by using the VB plugin supplied with the DSP/BIOS II code example or by using CCS watch windows with the use of the bootcode example source files. See the readme.txt files in those respective directories for instructions.

Chapter 5 Hardware Installation

The Mechatronics Control Kit supplies you with the pieces to assemble a DC-motor with load inertia experiment, a Reaction Wheel Pendulum, or a Pendubot. The dynamic equations and control design techniques for these experiments and be found in the **Control Tutorial Manual**. Figure 5-1 gives an overview of many of the different parts of the Kit. Please refer to this figure when reading the procedures below.

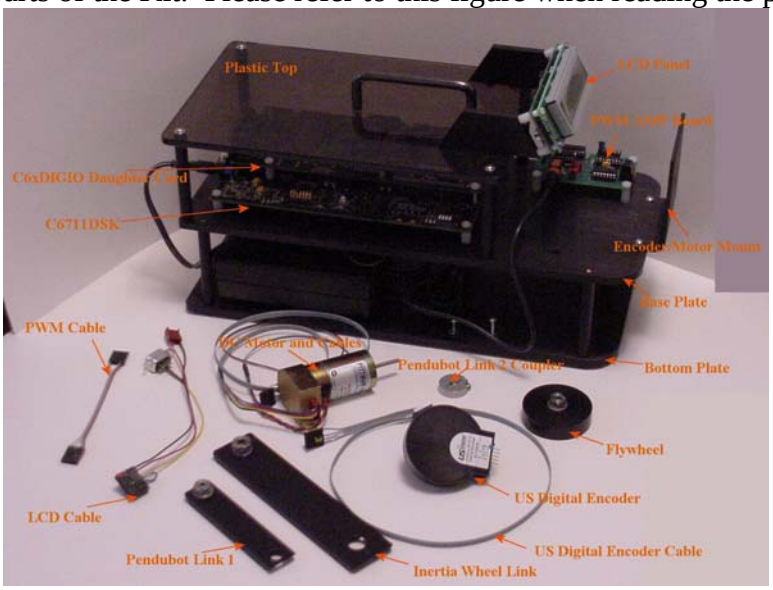


Figure 5-1: Overview of the Mechatronics Control Kit

There are a number of small parts needed to build each of the experiments. Some of the parts are used in more than one experiment and some are used in only one experiment. For example, the flywheel and 5 inch link are not used in the Pendubot experiment. In the same fashion, the Pendubot link1 piece and the Pendubot link2 and link2 coupler and not used in the reaction wheel experiment. To store these parts when using the other experiment, some tapped holes and screws have been supplied to allow you to mount the extra parts on the bottom plate. This way they will stay with the kit and not get lost. Figure 5-2 shows the locations given to store the different parts.

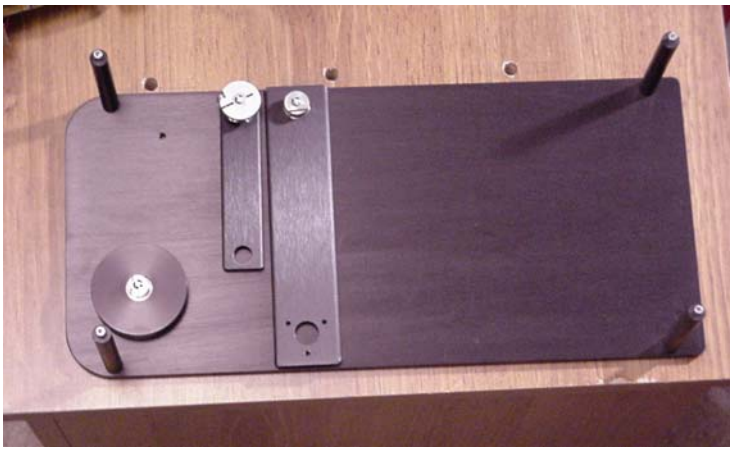


Figure 5-2: Base Plate showing link and flywheel stowage area

1. Building the DC-Motor Load Inertia Experiment

1. First mount the motor to the motor mount on the base plate using three 2-56 screws orientated so that the encoder connector faces straight up. If the motor mount is not already attached to the base plate do so now. You will want to have the M+/- cable and the encoder cable loosely wrapped around the motor body. This will keep these cables somewhat out of the way. Do not wrap the cables too tightly, which creates unneeded stress on the cable wires.
2. Screw the M+(red) and M-(black) leads into the screw terminal of the PWM AMP board.
3. Plug the motor encoder cable into the encoder connector. Make sure to align ground with pin 1.
4. Attach the flywheel using a 5/64" Allen wrench. Slide the flywheel until the motor shaft protrudes slightly from the back end of the flywheel. See Figure 5-3.

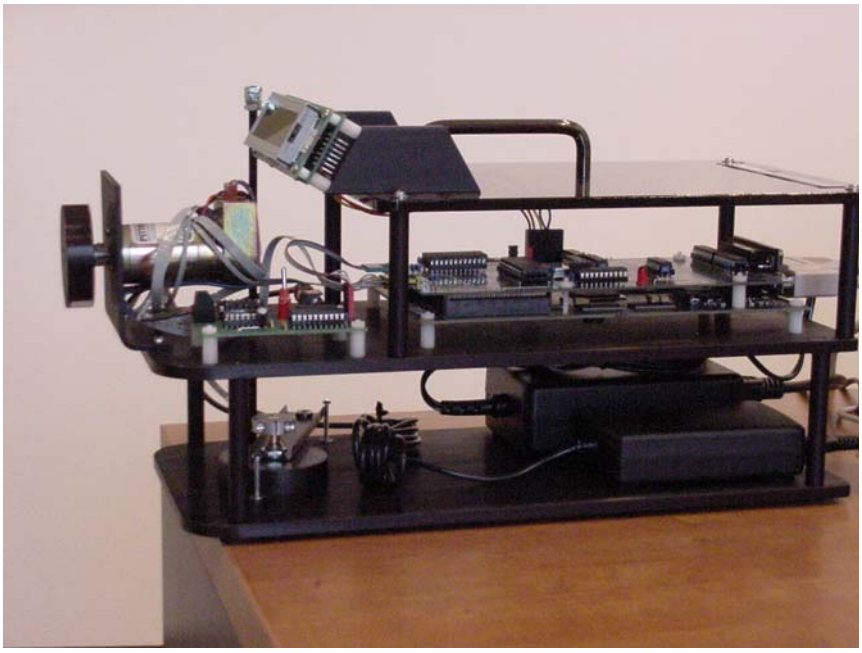


Figure 5-3: DC-Motor Control Setup

5. Plug the motor encoder cable into ENC1 on the Quanser interface board.

2. Building the Reaction Wheel Pendulum

With this experiment you have the highest chance of damaging the US Digital optical encoder unit. This is due to the large moment arm that is created by hanging the motor at the end of the 5-inch link. Just be careful when assembling this experiment and always try to be careful not to allow the motor and link to be bumped into when resting in the experiment configuration. Also avoid unneeded stress on the encoder and link coupling, **ALWAYS assemble the link/motor/flywheel assembly first before sliding the coupling onto the encoder shaft.**

1. Use a ½ inch socket to mount the encoder to the encoder mount. If the encoder mount is not already attached to the base plate do so now. Insert the encoder shaft through the backside of the encoder mount. With the encoder positioned so that the cable extends above the encoder mounting bracket, attach the encoder to the bracket using the 3/8-inch washer and nut supplied with the encoder. Tighten the nut using the ½ inch socket so that the encoder enclosure is not able to rotate but do not over-tighten the nut as this may damage the encoder.

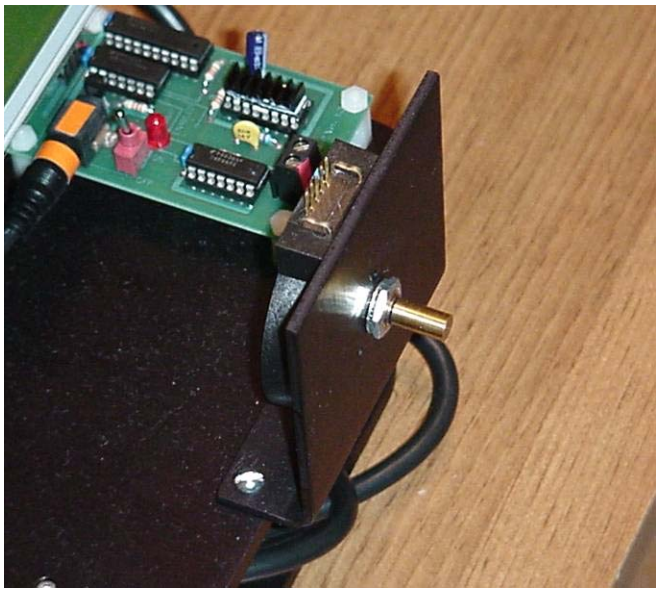


Figure 5-4: US Digital Encoder Attached to Front Bracket

2. Screw the four 2.5” extension feet to the bottom plate of the Mechatronics Kit. These extension feet raise the Reaction Wheel experiment high enough so that the motor does not hit the table when swinging.
3. Mount the motor, with three 2-56 screws, to the reaction wheel link as shown in the diagram. The flanges on the link should point toward the motor.
4. Attach the flywheel using a 5/64” Allen wrench. Slide the flywheel until the motor shaft protrudes slightly from the back end of the flywheel. See figure.

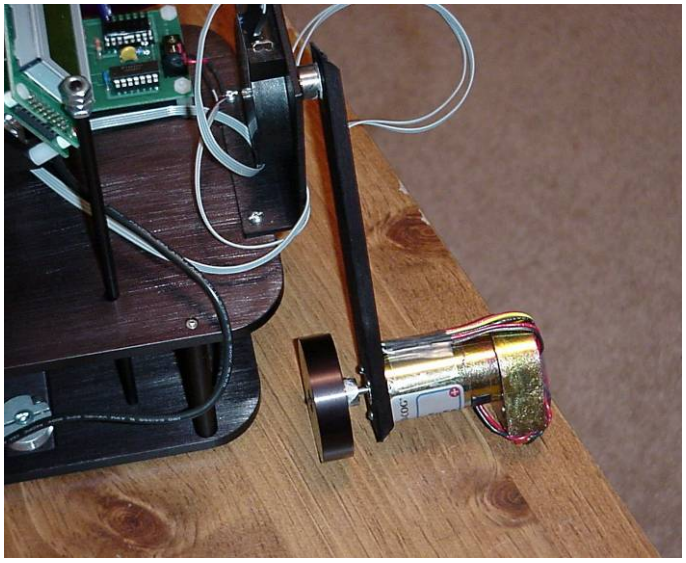


Figure 5-5: Reaction Wheel Showing Cable Routing

5. Tape the cables (encoder and motor) to the link as described in the cable routing section.
6. Slide the completed link assembly onto the encoder shaft as far as possible to minimize the moment arm created by the hanging motor but leaving a slight gap between the coupler and the stationary bearing housing. NOTE: Sliding the link assembly too far onto the encoder shaft will cause the link to rub against the stationary part of the encoder. Be VERY careful not to force the coupling over the shaft. Doing so may damage the encoder bearings. Tighten the coupler socket head cap screw so that the link cannot move on the encoder shaft.
7. Tape the motor and encoder cables to the encoder mount leaving enough slack to permit the link to rotate freely – approximate 2-3 inches. See Cable Routing Section
8. Attach the motor leads to the screw terminal on the PWM Amplifier Board. The red lead goes to M+ and the black lead goes to M-.
9. Plug the encoder connector from the motor into ENC2 on the Quanser interface board. This is the encoder #2 connector. Match ground with Pin1.
10. Plug the connector from the US Digital encoder into ENC1 on the Quanser interface board.
11. Plug the other end of the US Digital encoder cable into the connector on the US Digital encoder. Match ground with pin 1.

3. Building the Pendubot

As with the Reaction Wheel experiment it is very important not to put too much stress on the US Digital encoder when performing the assembly steps. Work gently with the different pieces and NEVER force any thing into position. When the experiment is complete, take steps to not allow the linkage to be bumped into by yourself or other objects. Avoid unneeded stress on the link1 one coupler. ALWAYS assemble the link1/US Digital encoder/link2 coupler assembly first before sliding the coupling onto the motor shaft.

1. First mount the motor to the motor mount on the base plate using three 2-56 screws orientated so that the encoder connector faces straight up. If the motor mount is not already attached to the base plate, do so now. You will want to have the M+/- cable and the encoder cable loosely wrapped around the motor body. This will keep these cables somewhat out of the way. Do not wrap the cables too tight creating unneeded stress on the cable wires.
2. Screw the M+ (red) and M- (black) leads into the screw terminal of the PWM AMP board.
3. Now assemble the linkage. Use a ½ inch socket to mount the US Digital encoder to the flat side of link-1 using the nut and washer supplied with the encoder.
4. Orientate the encoder so that the connector points up towards the link coupler. Rotate and center the encoder by eye and then tighten the nut with the ½ inch socket. NOT too tight, but tight enough so that the encoder housing cannot rotate easily against link-1.
5. Assemble the coupler for link-2 on the encoder shaft. Slide the coupler on the shaft so that the end of shaft is flush with the surface of the coupler.
6. Now JUST holding the coupler (try not to hold onto the encoder), use a 7/64 Allen wrench to tighten the coupler's socket head cap screw. The coupler should not be able to slide on the shaft when tightened.

Figure 5-6: Pendubot Link/Encoder Assembly

7. Plug the US digital encoder cable into the encoder connector. Make sure to match up pin 1 to ground. Tape the cable flat to link-1 just above the connector. See the cable routing section for more detail.
8. Slide the link-1 coupler onto the motor shaft. Slide the coupler just far enough so that the motor shaft barely protrudes out the back of the coupler. The link should sit pretty far out on the shaft so that the "swinging" cable has room to wrap a time or two.

9. Route and tape the US Digital encoder cable as instructed in the cable routing section. Plug the US Digital encoder cable into ENC2 on the Quanser interface board and the motor encoder cable into ENC1.
10. Finally, screw ¼ inch threaded end of link-2 into the link-2 coupler. The ½ inch threaded end of link-2 should have four 8-32 nuts on it.

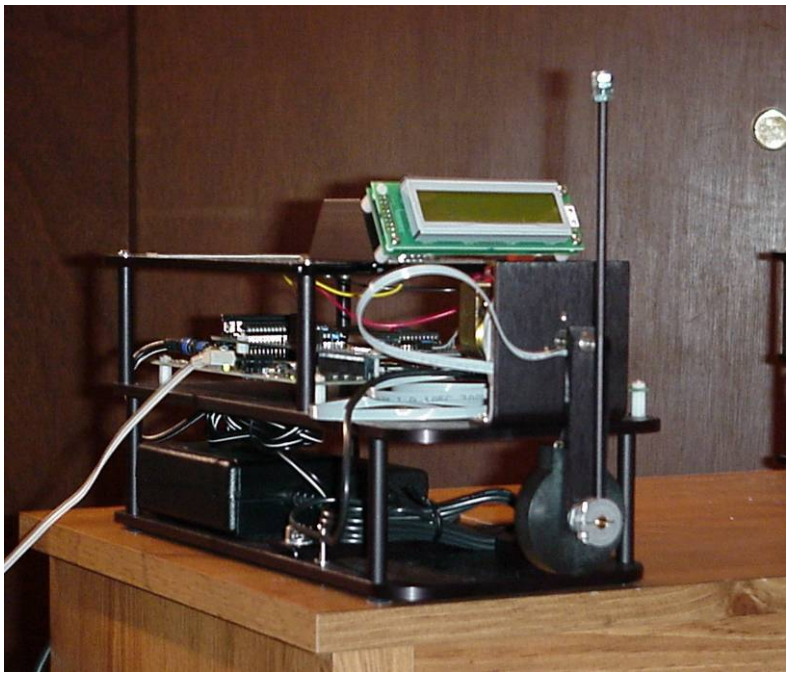


Figure 5-7: Completed Pendubot Setup

4. Building the Furuta Pendulum

As with the Reaction Wheel experiment it is very important not to put too much stress on the US Digital encoder when performing the assembly steps. Work gently with the different pieces and NEVER force any thing into position. Also when the experiment is complete, take steps to not allow the linkage to be bumped into by yourself or other objects. Also avoid unneeded stress on the link1 one coupler. ALWAYS assemble the link1/US Digital encoder/link2 coupler assembly first before sliding the coupling onto the motor shaft.

1. First assemble the Furuta upright assembly as shown in Figure 5-14. Use the two quarter inch aluminum pieces along with the motor mount to assemble the kit in this configuration. There are 4 ½" 6-32 screws that you will use to assemble the furuta's stand. Mount the motor to the motor mount using three 2-56 screws orientated so that the encoder connector faces away from the base. Tape the cables to the upright assembly so they are out of the way of the linkage.
2. Screw the M+ (red) and M- (black) leads into the screw terminal of the PWM AMP board.
3. Now assemble the linkage. Use a ½ inch socket to mount the S1 US Digital encoder to the flat side of link-1 using the nut and washer supplied with the encoder.
4. Orientate the encoder so that the connector points up towards the link coupler. Rotate and center the encoder by eye and then tighten the nut with the ½ inch socket. NOT too tight, but tight enough so that the encoder housing cannot rotate easily against link-1.
5. Assemble the coupler for link-2 on the encoder shaft. Slide the coupler on the shaft so that the end of shaft is flush with the surface of the coupler.
6. Now JUST holding the coupler (try not to hold onto the encoder), use a 7/64 Allen wrench to tighten the coupler's socket head cap screw. The coupler should not be able to slide on the shaft when tightened.

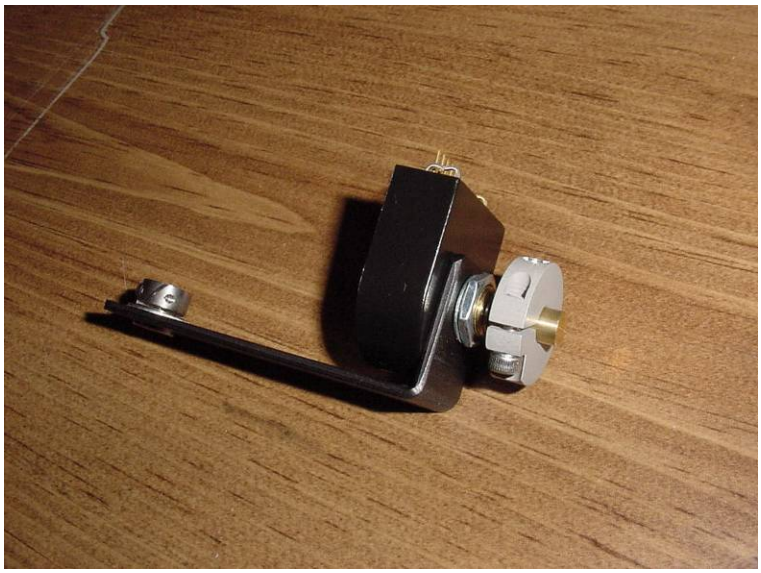


Figure 5-12: Furuta Link/Encoder Assembly

7. Plug the *US Digital Encoder* cable into the encoder connector. Make sure to match up pin 1 to ground. See the cable routing section for more detail.
8. Slide the link-1 coupler onto the motor shaft. Slide the coupler so that about an eighth inch of the motor shaft protrudes out from the coupler.

9. Route and tape the *US Digital Encoder* cable as instructed in the cable routing section. Plug the US Digital encoder cable into the ENC 2 connector on the *Quanser Interface* board and the *Motor Encoder* cable into the ENC 1 connector on the *Quanser Interface* board.

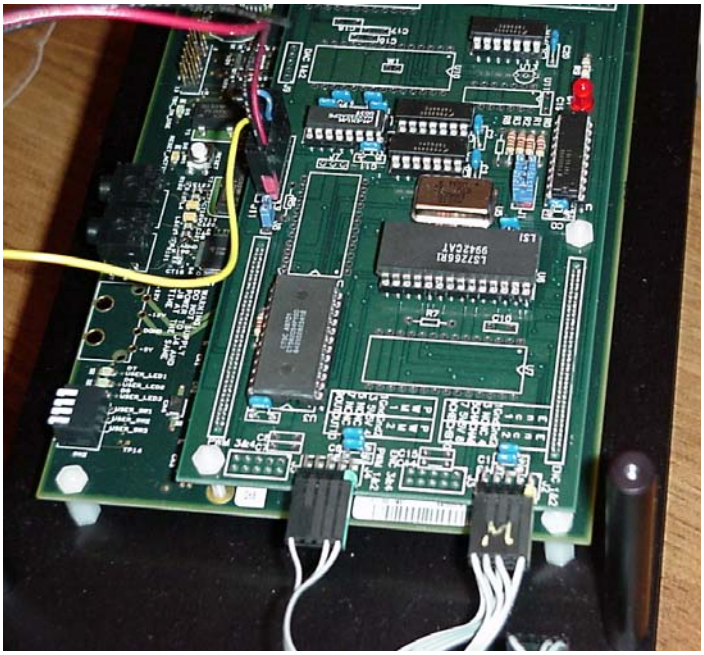


Figure 5-13: Encoder and PWM Amp Connections for the Furuta Pendulum

10. Finally, screw ¼ inch threaded end of link-2 into the link-2 coupler piece. The ½ inch threaded end of link-2 should have four 8-32 nuts on it.

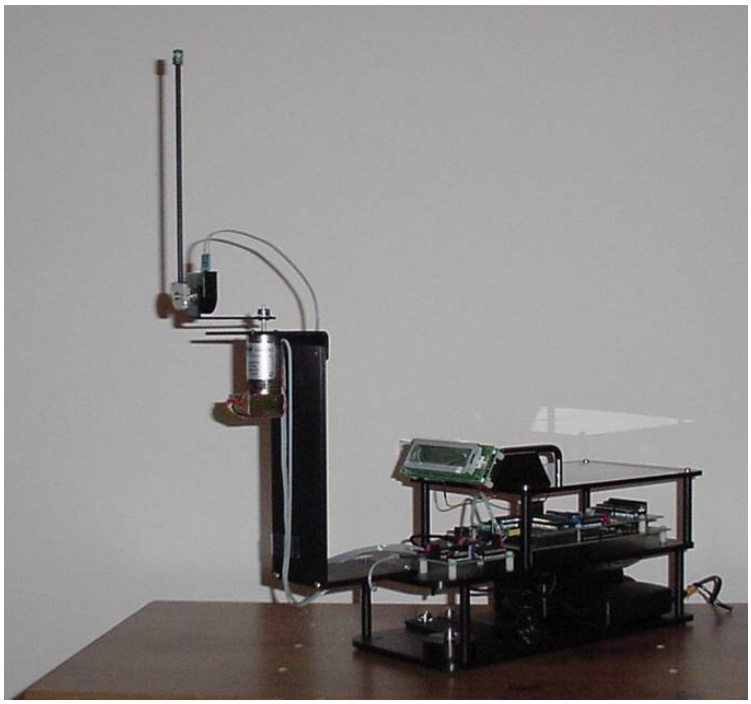


Figure 5-14: Completed Furuta Pendulum Setup

5. Routing the Cables

The Mechatronics Kit includes the following cables:

1. US Digital Encoder Cable. Four wire ribbon cable approximately 22 inches long. One end to mate at the Quanser interface board and the other end to mate at the US Digital Encoder connector.
2. Motor + and – cable (attached to the motor). RED M+, BLACK M-.
3. Motor Encoder cable (attached to the motor). Free end is to mate at the Quanser interface board.

Notice the labels that have been placed on the encoder connectors. Pin 1 is marked as ground. Pin 1 of the encoder headers on the Quanser interface board are denoted by a notch in the silk-screened box surrounding the header. So, when connecting the cables for the different experiments, make sure to match Pin 1. Note that the US Digital encoder is also labeled with Pin 1 as ground.

The two 2.5mm power cables are also marked with colored tape. BLUE is the 5VDC supply connector to be connected to the C6713DSK. ORANGE is the 24VDC supply to be connected to the Quanser interface board. DO NOT plug the 24VDC supply into the C6713DSK. ALWAYS first connect the power connectors to the boards and turn on AC power to the DC supply. Plugging the DC supplies into a dedicated power strip is recommended. This way you have an On/Off switch for the Kit.

You will find that you will spend most of the building time of constructing either the Pendubot or Reaction Wheel experiment in getting the routing of the cables correct. Routing the cables so that they produce a minimum amount of drag on the experiments is very important in getting the controllers to work properly.
You will find that different bends and loops in the cable routing will cause your system to react in different ways. In other words, the cable plays a big role in the system model. In this section, we have supplied pictures and a description of how we recommend routing the cables for the Pendubot and reaction wheel experiments. Feel free to experiment with your own routing, but these have worked best for us.

You will need to be careful with the ribbon cables because with excessive bending and pulling the 28 AWG wires of the ribbon cable can be broken. Take care when inserting the connectors that you are not pinching the cable wires in the process. If you do break a cable though, they are very easy to build. We have supplied a section below giving the parts and pin outs for each cable.

Routing the cables for the Pendubot Experiment

You will want to read this section along with the section that discusses assembling the Pendubot experiment. Before routing the "swinging" cable you should first have most of the Pendubot experiment built. The motor should be mounted and its M+/- cable should be loosely wrapped around the motor and its leads screwed into the terminal block of the PWM AMP board. The encoder should be attached to link-1 of the Pendubot and the link-2 coupler should be tightened on the shaft of the encoder. (Do not attach Link-2 until after the cable has been routed.)

1. Connect the *US Digital Encoder* cable to the connector on the US Digital encoder. Match Pin 1 of the 5-pin female connector with Pin 1 on the encoder housing.
   NOTE: The bent angle of the encoder connector pins is correct. They are angle toward the link in order to keep the connector a little more out of the way when the cable wraps up with multiple revolutions of link-1.
2. With the ribbon cable laying flat on link-1, use a piece of Scotch tape to tape the cable to the link just above the connection, as illustrated in Figure 5-15.

Figure 5-15: Taping the Encoder Cable to Pendubot Link-1

3. Now slide the link1 coupler over the motor shaft and tighten in proper position.
4. Using the figures below as a guide, create a 3 inch loop to the left (when facing the linkage) of link-1 and tape the cable to the back side of the motor mount. There should be no twists in the cable and you may have to shape the cable slightly after it has been taped to the motor mount. You may have to try the routing a couple of times to get it the way you want it. Rotate link-1 up 180 degrees and watch that the cable does not touch any obstacles along the way.

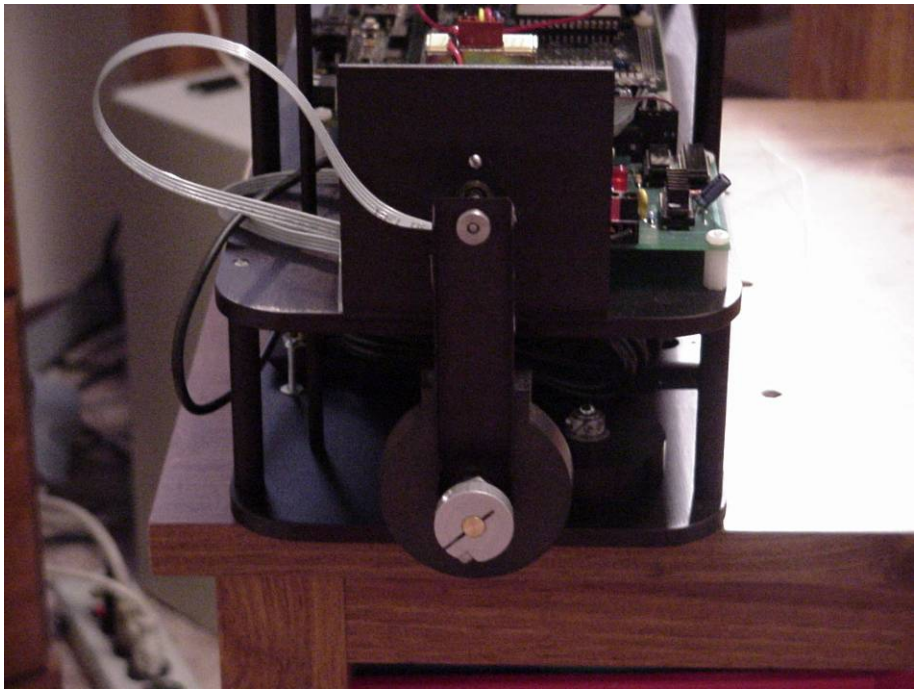


Figure 5-16: Pendubot Cable Routing

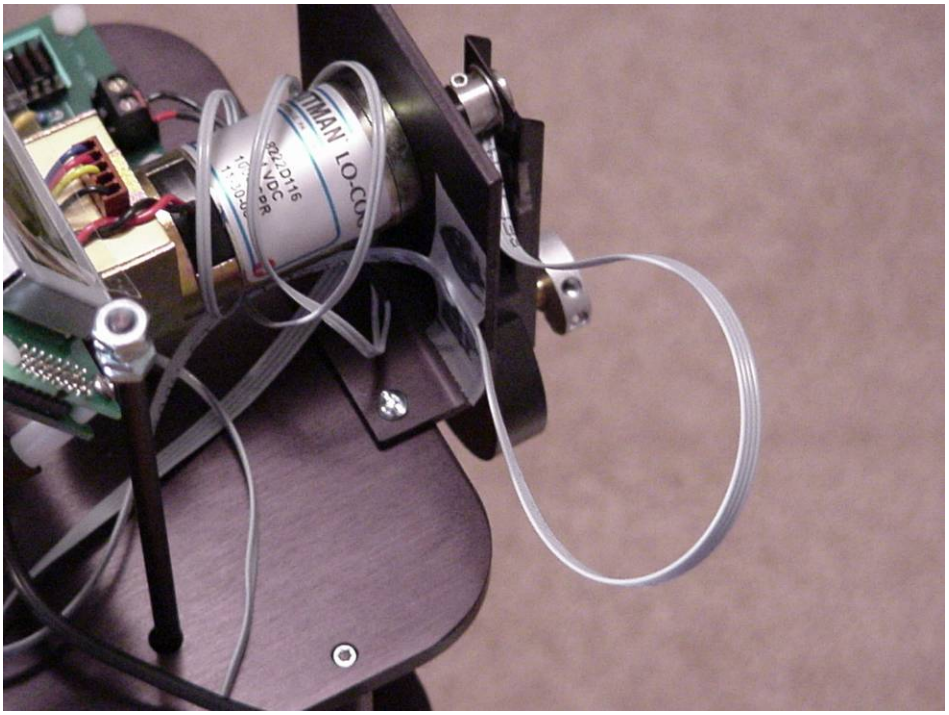


Figure 5-17: Pendubot Cable Routing

Figure 5-18 Pendubot Cable Routing

5. Connect the *US Digital Encoder* cable, which is now routed, to the ENC 2 connector on the *Quanser Interface* board. Ensure that Pin 1 of the connector to Pin 1 on the board are matched.
6. Connect the *Motor Encoder* cable to the ENC 1 connector on the *Quanser Interface* board. Again, match Pin1 of the connector to Pin 1 on the board. Make sure that the excess of this cable is not in the path of link-1.

Routing the cables for the Reaction Wheel Experiment

You will want to read this section along with the section that discusses assembling the reaction wheel experiment. Before routing the motor power and encoder cables, you should first have most of the reaction wheel experiment built. The US Digital encoder should be mounted to the encoder mount. Wait to plug in its cable until after you have the motor encoder cable connected. The motor and flywheel should be assembled as discussed in the assembly instructions.

1. Lay both ribbon cables from the motor flat along the inside of the link. Use a piece of Scotch tape to tape the cables side by side about an inch below the link coupler. See Figure 5-16. Make sure there are no twists in the cables.

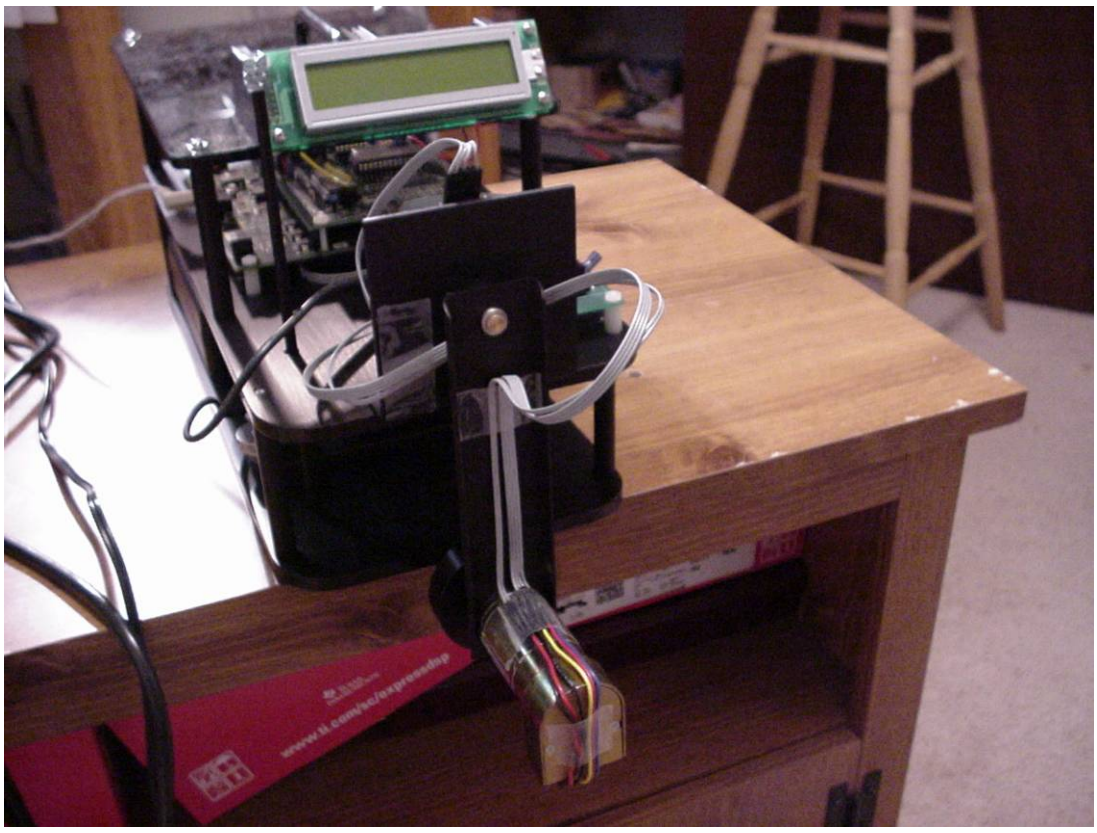


Figure 5-19 Reaction Wheel Cable Routing

2. Now slide the link coupler over the encoder shaft, position in the correct location and tighten. As stated in the assembly instructions, be very gentle when performing this assembly. Forcing the link on could damage the US Digital encoder.
3. Using Figures 5-16 and 5-17 as a guide, create a 3-inch loop to the right (when facing the linkage) of the link and bring them behind the link and tape the cables to the front side of the encoder mount. There should be no twists in the cables and you may have to shape the cables slightly after they have been taped to the encoder mount. You may have to try the routing a couple of times to get it the way

you want it. Rotate the link up 180 and –180 degrees and watch that the cables do not touch any obstacles along the way.
NOTE: The link will hit the loop portion of the cable when you rotate the link positive (Counter Clock Wise).

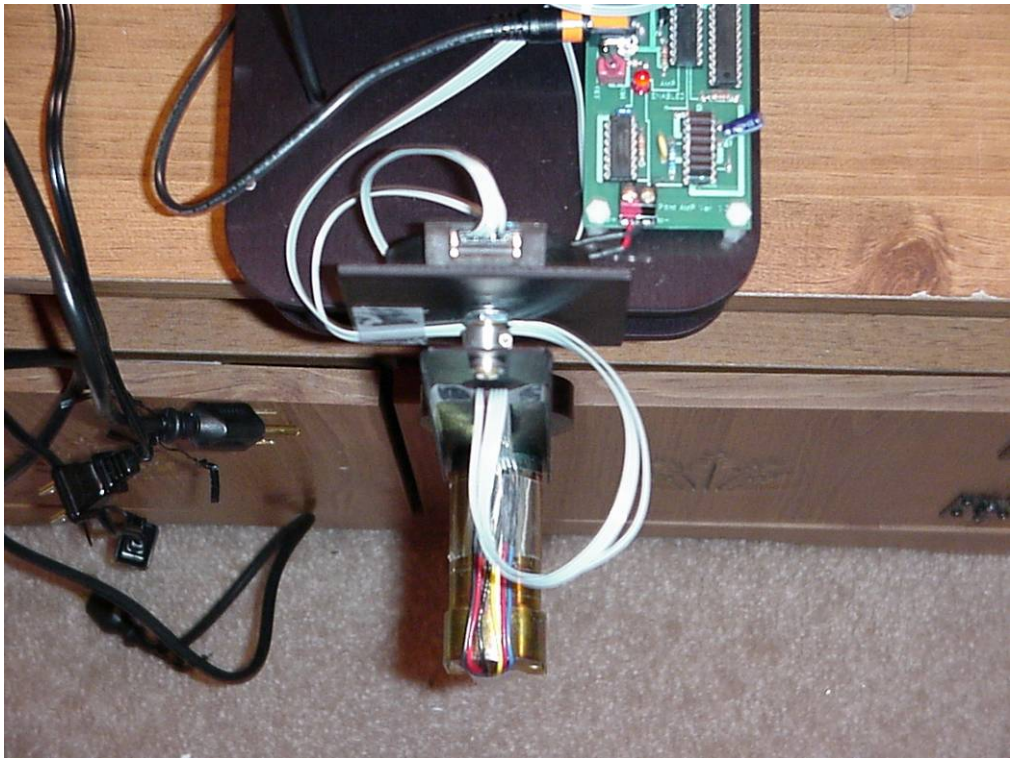


Figure 5-20 Reaction Wheel Cable Routing

4. With the ribbon cables routed, connect the end of the *Motor Encoder* cable to the *ENC2* connector on the *Quanser interface* board. Match Pin 1 of the connector to Pin 1 on the board
   NOTE: This connection is different from all the other experiments. Only in this experiment, the Reaction Wheel, is the *Motor Encoder* connected to the ENC 2 connector (usually always wired to ENC 1).
5. Also screw the M+ (red) and M- (black) leads into the terminal of the PWM Amp Board.
6. Connect the *US digital Encoder* cable to the ENC 1 connector on the *Quanser Interface* board. Match Pin 1 on the connector to Pin 1 on the board. Also, ensure the cable is also plugged into the US Digital encoder (match Pin 1 on the connector to Pin 1 on the encoder) and that the excess of this cable is not in the path of the link's motion.

Routing the cables for the Furuta Pendulum Experiment

You will want to read this section along with the section that discusses assembling the Furuta Pendulum experiment. Before routing the "swinging" cable you should first have most of the Furuta Pendulum experiment built. The motor should be mounted and its leads screwed into the terminal block of the PWM AMP board. The encoder should be attached to link-1 of the Furuta and the link-2 coupler piece should be tightened on the shaft of the encoder. (Do not attach Link-2 until after the cable has been routed.)

1. Connect the US Digital encoder cable to the connector on the US Digital encoder. Match Pin 1 of the connector to Pin 1 on the encoder body.
2. Tape the encoder cable to the upright aluminum piece as shown in Figures 5-21 – 5-23. Make sure to arch the cable as in the figure so link one can turn 180 degrees in each direction without being disrupted by the cable. Shape the cable and make sure there are no twists in it.

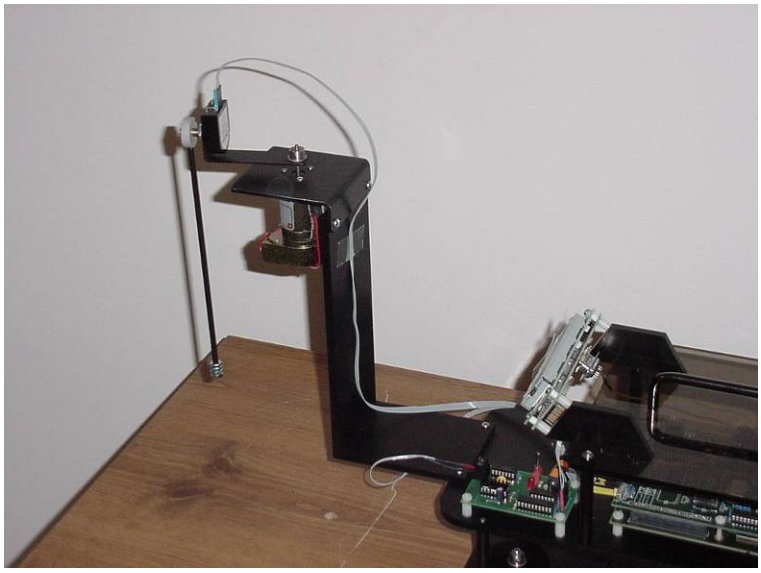


Figure 5-21: Furuta Pendulum Cable Routing

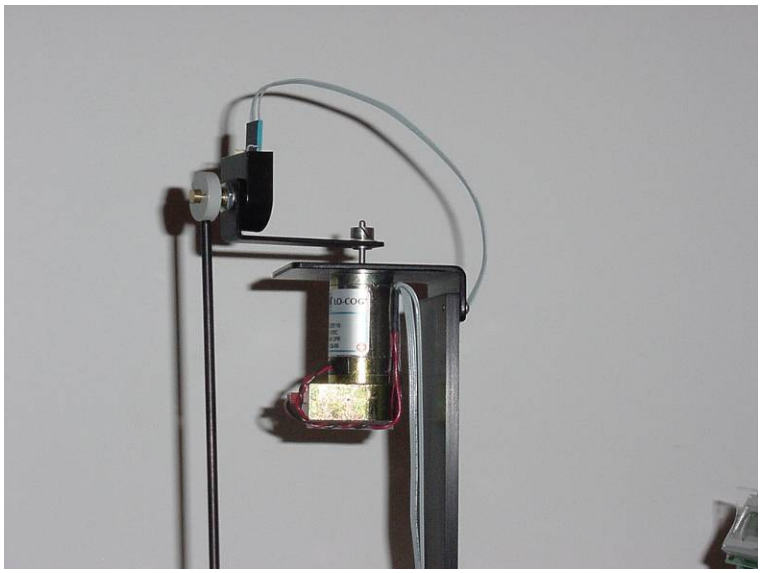


Figure 5-22: Furuta Pendulum Cable Routing

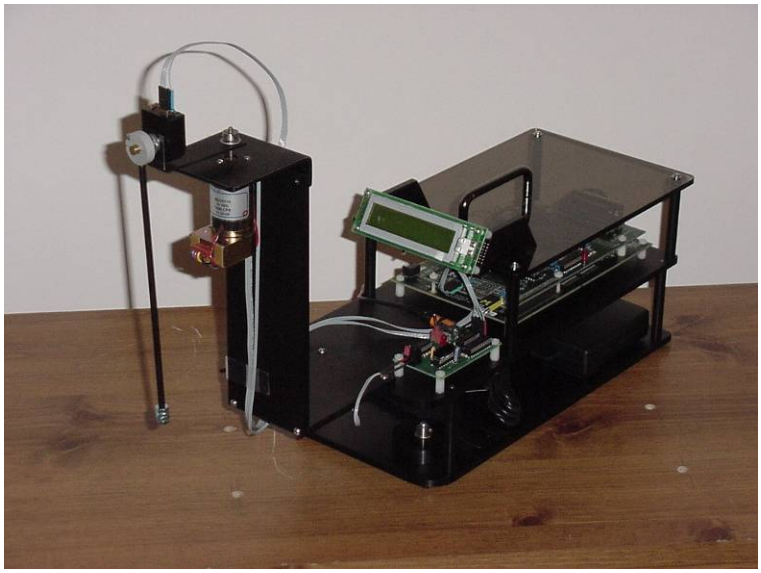


Figure 5-23: Furuta Pendulum Cable Routing

3. Connect the ribbon cable from the US Digital Encoder, which is now routed, to the *ENC2* connector on the *Quanser Interface* board. Make sure Pin 1 on the connector matches with Pin 1 on the board.
4. Connect the *Motor Encoder* cable to *ENC1* on the *Quanser Interface* Board. Match Pin 1 on the connector to Pin 1 on the board. Also, make sure that the excess of this cable is not in the path of link-1.

6. Cable Pin outs and Part List

a. PWM cable:

Quanser Interface 5 Pin		PWM AMP 5 Pin
Pin1	----------	Pin1 (GND)
Pin2	----------	Pin2 (5V)
Pin5	----------	Pin5 (PWM signal)

Part List

Any standard 28 AWG 0.05 in. spacing stranded ribbon cable
Crimp Pins: DigiKey part number: WM2557-ND
Crimp Tool: DigiKey part WM9999-ND
5 pin housings: DigiKey part number: WM2803-ND

2. US Digital encoder cable:

Quanser Interface 5 Pin		Encoder
Pin 1	----------	Pin 1 (GND)
Pin 3	----------	Pin 3 (CH A)
Pin 4	----------	Pin 4 (5V)
Pin 5	----------	Pin 5 (CH B)

Part List

Any standard 28 AWG 0.05 in. spacing stranded ribbon cable
Crimp Pins: DigiKey part number: WM2557-ND
Crimp Tool: DigiKey part WM9999-ND
5 pin housings: DigiKey part number: WM2803-ND

3. Motor Encoder cable:

Quanser Interface 5 Pin		Motor Wires
Pin 1	----------	Black (GND)
Pin 3	----------	Yellow (CH A)
Pin 4	----------	Red (5V)
Pin 5	----------	Blue (CH B)

Part List

Any standard 28 AWG 0.05 in. spacing stranded ribbon cable
Crimp Pins: DigiKey part number: WM2557-ND
Crimp Tool: DigiKey part WM9999-ND
5 pin housings: DigiKey part number: WM2803-ND

4. Motor + and – Leads: Solder 22 AWG wire onto ribbon cable wire for screw terminal ends.
Part List

Any standard 28 AWG 0.05 in. spacing stranded ribbon cable
22 AWG Red and Black stranded wire.

5. LCD cable: Daughter Card 16 Pin Housing (If a pin is not listed in is a No Connect)

Pin 1 is connected to Pin 4 (Gnd) of LCD Power Connector
Pin 3 is connected to Pin 1 (5V) of LCD Power Connector
Pin 8 is connected to Pin 3 (Serial Receive) of LCD 9pin Dsub
Pin 9 is jumpered to Pin 10 of this connector
Pin 10 is jumpered to Pin 9 of this connector

Part List

4 pin LCD Power Connector: DigiKey part number: WM2002-ND
Crimp pins for LCD Power Connector : DigiKey part number:WM2200-ND
9 pin Dsub DigiKey part number: 209M-ND
16 pin Housing: DigiKey part number: WM2525-ND
Crimp pins for 16 pin Housing: DigiKey part number: WM2557-ND
Crimp tool for all pins: DigiKey part number: WM9999-ND

7. C6713DSK, C6xDSK_DigIO and “Quanser Standard” Interface / PWM Amplifier boards

The Mech. Kit includes three different printed circuit boards: The Texas Instruments TMS320C6713DSP Starter Kit, the C6XDSK_DigIO daughter card and the “Quanser Standard” Interface / PWM Amplifier board. This section gives a brief overview of these boards and where to find other resources to better understand how each of these boards work.

1. **Texas Instruments TMS320C6713 DSP Starter Kit Board (C6713 DSK).** We purchase this board from Texas Instruments and supply it with the kit. You can read the details of this board in the “Hardware” section of the “C6713 DSK Help” (i.e. c6713dsk.hlp) that is supplied with the Code Composer Studio software. In brief this board includes the following:
    - **a.** 225MHz TMS320C6713 (floating point) processor
    - **b.** 16 Mbytes of SDRAM
    - **c.** 512 Kbytes of flash
    - **d.** 16 bit Audio Codec
    - **e.** JTAG USB interface to Host PC
    - **f.** Expansion connectors for add-on daughter cards.

Figure 5-24 6713 DSK Board

This DSK board acts as the “motherboard” for our system. All the control routines are run on the C6713 floating point processor and use the external SDRAM supplied on the DSK. The USB

connector on this board is used to communicate with the host PC. JTAG emulation is achieved through this USB port. By reading through TI's documentation and tutorials you will become very familiar with this board and its capabilities. The DSK kit comes included with the DSP board, Code Composer Studio software and DSP/BIOS II. DSP/BIOS II is a new DSP operating system that TI released in 2000. It is a powerful real-time operating system that still allows for relatively small footprint in the DSP's memory. In the CCS "Help" menu, click on "Contents" and see the "DSP/BIOS" item in the left panel for an overview of the DSP/BIOS operating system. For additional information, see the TMS320C6000 DSP/BIOS User's Guide (SPRU423) document. This manual along with many more C6000 documents can be found (after CCS has been installed) under the "Help" menu.
This manual for the Mech. Kit is intended to give you enough information to run the different example programs supplied and get you started with the dynamics models of the two experiments. We have not gone into detail though on the CCS software. We recommend that you take some time to read and perform the different manuals and tutorials supplied by TI with this development kit. Also use TI's web page www.ti.com as a resource for application notes and updated documentation.

2. **Quanser Consulting Inc. C6xDSK_DigIO daughter card for the C6713 DSK:** This is a board that we fabricate for the Mechatronics Kit. This board adds optical encoder feedback and PWM output capabilities to the C6713 DSK system. It also implements a parallel port interface used to communicate with Matlab's Real Time Workshop software. See the RTW section for more details. We have supplied you with the full schematics of this board (docs\schematics\board_schematics.pdf) so the details of how the board operates can be found there along with programming details found in the source file "include\c6xdskdigio.c". In this section we will give a brief overview of the board and a description and pin locations of each connector.

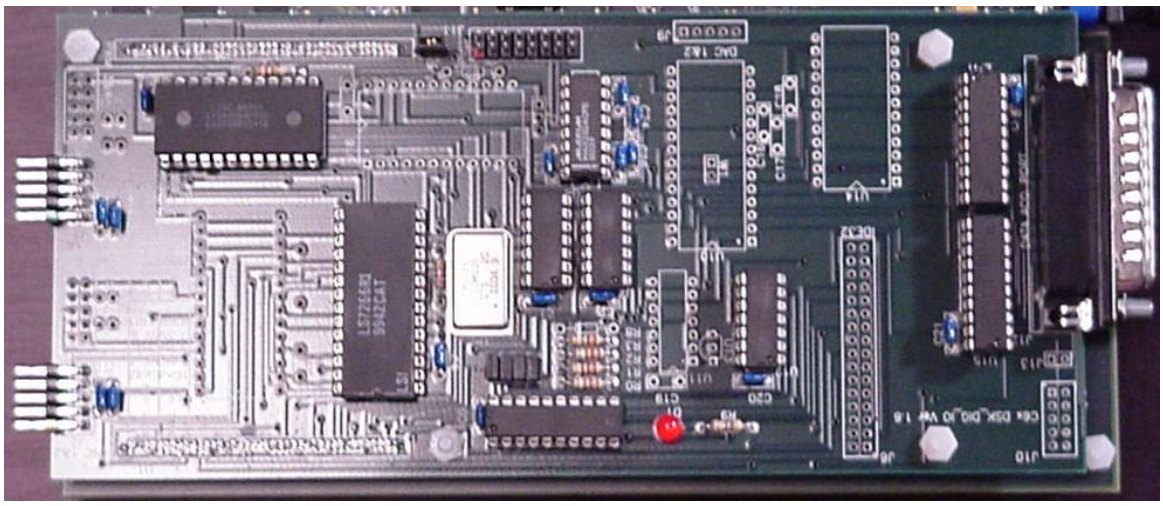


Figure 5-25 C6xDSK_DigIO Daughter Card

As you can see in Figure 5-19, the C6xDSK_DigIO daughter card has not been fully populated. The board has more functionality than the mechatronics kit requires. You may find in the future that you would like to use this DSP for other purposes then just the mechatronics kit experiments. In that case you will want to study the supplied schematics and add the needed components to add additional capabilities to your daughter card. Below we have listed the capabilities of the daughter card and list as "upgradeable" the capabilities not populated on the daughter card. Presently we do not perform the upgrades here at Quanser but will give you the part numbers and suggested distributors for purchasing the need parts. Please contact us for this information.

The C6xDSK_DigIO daughter card adds the following I/O to the C6713 DSK: (see the source file c6xdskdigio.c for details on the supplied functions used to communicate with the daughter card.)

1. Two Channels of Quadrature Encoder Input (LS7266R1 chip) upgradeable to 4 Channels. Brought in at connector J2.
2. Two PWM Output channels (CTS82C54 chip) upgradeable to 4 Channels. Brought out at connector J4.
3. When the DSK system is not being used in the "dummy data acquisition mode" (the Matlab/RTW interface) there are 8 digital input pins and 8 digital output pins brought to the J10 and J12 connectors.
4. Can be upgraded to add 2 channels of +/-10V (12bit) DAC output.
5. A MAX232 chip for converting TTL serial in/out voltages (0 to 5V) to RS 232 standard in/out voltages (-12 to 12V). See schematics and J11 pin out for more details.
6. A 32 Pin expansion header to allow for additional expansion "granddaughter cards" to add other feature to the DSP system. For example a 4 channel 12bit +/-10V ADC granddaughter card has been fabricated for expansion header to add 4 ADC channels to the DSP system. Another example would be to build a granddaughter card that would allow the DSP to communicate with an inexpensive micro-controller programmed to do one specific task.
7. Serial port 1 expansion header giving you access to all the McBSP1 pins. This would allow you to add a serial interface "granddaughter card" to the DSP system. Note: the mechatronics kit presently uses this header to connect to the LCD screen. If you choose to use this header for your own purposes you will have to design another way to communicate with the LCD or not use the LCD.
8. Access to the 4 DSP external interrupt pins found at connectors J6, J10 and J12.
9. Power ON LED for indication of appropriate power.

Below you will find the pin outs and descriptions for the different connectors and jumpers on the C6xDSK_DigIO daughter board. The connectors and jumpers needed for the mechatronics kit will be listed first followed by the connectors left unpopulated.

1. Board Address Selector **J1**. The four jumpers installed or uninstalled on the pins of connector J1 determine the board's address location in the DSP memory space. See the schematics for full details, but in short J1 pins 1,3,5,7 are pulled HI through 2.2 KOhm resistors and connected to an identity comparator chip that compares those pins with the DSP address lines A18-A21 respectively. Pins 2,4,6,8 are connected to ground. When a jumper is installed (i.e. between 1&2,3&4,5&6,7&8) the address pin is being compared with a LOW logic level. When the jumper is not installed a HI logic level is being compared. In this way you can set the address of the board. The default board address is set to all the jumpers installed or 0x000000. The C6xDSK_DigIO daughter card uses CE2 space therefore the base address of the daughter card is 0xA0000000. The only reason you would ever need to change these jumper settings is if would have the need to install another daughter card in between the C6713DSK and the C6xDSK_DigIO daughter card. In that case you may find that you need to change the address so that it does not conflict with the new daughter card's memory space.

2. Optical encoder connector for encoder #1 and #2 **J2**. Use this connector to plug the US Digital encoder and the motor encoder into the DSP system.

Pin#	Signal Name	Description	Pin#	Signal Name	Description
1	DGND	DSP/Daughter Digital Gnd	2	DGND	DSP/Daughter Digital Gnd
3	NC	No Connect	4	NC	No Connect
5	ENC1CHA	Encoder 1 Channel A	6	ENC2CHA	Encoder 2 Channel A
7	5V	DSP/Daughter Power	8	5V	DSP/Daughter Power
9	ENC1CHB	Encoder 1 Channel B	10	ENC2CHB	Encoder 2 Channel B

Connector for Optical Encoder Inputs 1 and 2 (J2)

3. PWM connector for PWM outputs #1 and #2 **J4**. Use this connector to connect the PWM output signal to the PWM amplifier board.

Pin#	Signal Name	Description	Pin#	Signal Name	Description
1	DGND	DSP/Daughter Digital Gnd	2	DGND	DSP/Daughter Digital Gnd
3	5V	DSP/Daughter Power	4	5V	DSP/Daughter Power
5	NC	No Connect	6	NC	No Connect
7	NC	No Connect	8	NC	No Connect
9	PWM1	PWM Output 1	10	PWM2	PWM Output 2

Connector for PWM Outputs 1 and 2 (J4)

4. DB_CLKS1 jumper **J8**. The J8 jumper connects the 10MHz clock of the C6xDSK_DigIO daughter card to the CLKS1 pin of McBSP1. This jumper MUST be installed for the given LCD code to work. The slow (at least compared to the 150MHz clock of the DSP) 10MHz clock is used to generate the slow 19200 Baud rate for the LCD serial port transmissions. The only reason you would need to remove this jumper is if you would like to use the J11 serial port connector for a different purpose then the LCD screen.

5. Serial Port/LCD connector **J11**. Use this connector to connect the serial LCD panel to the DSP's serial port 1 (McBSP1). The McBSP1's pins are not connected to the MAX232 RS232 voltage converter chip on the board. Instead these connections need to be made at connector J11. So to convert McBSP1's data transmit pin to RS232 voltages pin 9 is jumper to pin 10 and pin 8 is taken to LCD. See the schematics for more details. Power is also brought out to this connector to power the LCD. If the LCD screen is not needed, you can use this connector to connect McBSP1 to other peripherals.

Pin#	Signal Name	Description	Pin#	Signal Name	Description
1	DGND	DSP/Daughter Digital Gnd	2	5V	DSP/Daughter Power
3	5V	DSP/Daughter Power	4	RS_Rout1	RS-232 Receive Output
5	DB_DR1	DSP Serial1 Data Receive	6	RS_Rin1	RS-232 Reveive Input
7	DB_FSR1	DSP Serial1 FSR	8	RS_Tout1	RS-232 Transmit Output
9	DB_DX1	DSP Serial1 Data Transmit	10	RS_Tin1	RS-232 Transmit Input
11	DB_FSX1	DSP Serial1 FSX	12	DB_CLKX1	DSP Serial1 CLKX1
13	DB_CLKR1	DSP Serial1 CLKR1	14	NC	No Connect
15	5V	DSP/Daughter Power	16	DGND	DSP/Daughter Digital Gnd

Serial Interface Expansion Header (J11)

6. Parallel Port Interface/General Purpose Digital I/O **J12.** This connector is used for two purposes. If you would like to use the Matlab/Real-Time Workshop interface that we have developed for the mechatronics kit then you would plug your parallel port cable into this connector (instead of the DSK's parallel port connector) and run your Simulink code using either WinCon or Windows Target. See section on RTW. Otherwise this connector can be used to access 8 digital inputs and 5 digital outputs (connector J10 has the additional 3 digital outputs) generated by the daughter card. External interrupt #4 is also brought out to this connector if J13 is installed.

Pin#	**Parallel Port Signal Name&Description**		**General Purpose Signal Name&Description**	
1	NC:	No Connect	NC:	No Connect
2	PD0:	Parallel Port Data Pin 0	DIN0:	Digital Input Pin 0
3	PD1:	Parallel Port Data Pin 1	DIN1:	Digital Input Pin 1
4	PD2:	Parallel Port Data Pin 2	DIN2:	Digital Input Pin 2
5	PD3:	Parallel Port Data Pin 3	DIN3:	Digital Input Pin 3
6	PD4:	Parallel Port Data Pin 4	DIN4:	Digital Input Pin 4
7	PD5:	Parallel Port Data Pin 5	DIN5:	Digital Input Pin 5
8	PD6:	Parallel Port Data Pin 6	DIN6:	Digital Input Pin 6
9	PD7:	Parallel Port Data Pin 7	DIN7:	Digital Input Pin 7
10	PS6:	Parallel Port Status Pin 6	DOUT3:	Digital Output Pin 3
11	PS7:	Parallel Port Status Pin 7	DOUT4:	Digital Output Pin 4
12	PS5:	Parallel Port Status Pin 5	DOUT2:	Digital Output Pin 2
13	PS4:	Parallel Port Status Pin 4	DOUT1:	Digital Output Pin 1
14	NC:	No Connect	NC:	No Connect
15	PS3:	Parallel Port Status Pin 3	DOUT0:	Digital Output Pin 0
16	NC:	No Connect	NC:	No Connect
17	PC3:	Parallel Port Control Pin 3	EINT4	DSP External Interrupt 4 (Jumper J13 Must be installed)
18-25	DGND	DSP/Daughter Digital Ground		

Parallel Port/General Purpose Digital I/O Connector (J12)

Connectors not required for Mechatronics Kit

1. Optical encoder connector for encoder #3 and #4 **J3**. Install this connector after installing the second encoder interface chip (LS7266R1).

Pin#	Signal Name	Description	Pin#	Signal Name	Description
1	DGND	DSP/Daughter Digital Gnd	2	DGND	DSP/Daughter Digital Gnd
3	NC	No Connect	4	NC	No Connect
5	ENC3CHA	Encoder 3 Channel A	6	ENC4CHA	Encoder 4 Channel A
7	5V	DSP/Daughter Power	8	5V	DSP/Daughter Power
9	ENC3CHB	Encoder 3 Channel B	10	ENC4CHB	Encoder 4 Channel B

Connector for Optical Encoder Inputs 3 and 4 (J3) (See Schematics to add the appropriate ICs to the daughter card for the additional Encoder Inputs.)

2. PWM connector for PWM outputs #3 and #4 **J5**. Install this connect after installing the second 82C54 timer chip. With the use of DSP Timer1 and jumper J7 to connect Timer 1 to the second 82C54 chip, the 82C54 can be programmed to control RC Servo motors. Contact us at Quanser for more information.

Pin#	Signal Name	Description	Pin#	Signal Name	Description
1	DGND	DSP/Daughter Digital Gnd	2	DGND	DSP/Daughter Digital Gnd
3	5V	DSP/Daughter Power	4	5V	DSP/Daughter Power
5	NC	No Connect	6	NC	No Connect
7	NC	No Connect	8	NC	No Connect
9	PWM3	PWM Output 3	10	PWM4	PWM Output 4

Connector for PWM Outputs 3 and 4 (J5) (See Schematics to add the appropriate ICs to the daughter card for the additional PWM Outputs.)

3. Granddaughter Card Expansion Connector **J6.** Use this connector to add additional expansion boards to your DSP system.

# Pin #	Signal Name	Description	Pin#	Signal Name	Description
1	DGND	DSP/Daughter Digital Gnd	2	5V	DSP/Daughter Power
3	-15V	Daughter Card -15V	4	+15V	Daughter Card +15V
5	BS	Daughter Card Board Select	6	DB_EINT7	DSP External Interrupt 7
7	DB_D0	DSP Data Line 0	8	DB_AWE	DSP Write Line
9	DB_D1	DSP Data Line 1	10	DB_D2	DSP Data Line 2
11	DB_D3	DSP Data Line 3	12	DB_D4	DSP Data Line 4
13	DB_D5	DSP Data Line 5	14	DB_D6	DSP Data Line 6
15	DB_D7	DSP Data Line 7	16	DB_D8	DSP Data Line 8
17	AGND	Daughter Card Analog Gnd	18	DGND	DSP/Daughter Digital Gnd
19	DB_D9	DSP Data Line 9	20	DB_D10	DSP Data Line10
21	DB_D11	DSP Data Line 11	22	DB_A2	DSP Address Line 2
23	DB_A3	DSP Address Line 3	24	DB_A12	DSP Address Line 12
25	DB_ARE	DSP Read Line	26	DB_A13	DSP Address Line 13
27	DB_EINT6	DSP External Interrupt 6	28	AGND	Daughter Card Analog Gnd
29	+15V	Daughter Card +15V	30	-15V	Daughter Card –15V
31	5V	DSP/Daughter Power	32	DGND	DSP/Daughter Digital Gnd

Parallel Interface Expansion Header (J6)

4. Jumper **J7.** Used in conjunction with the second 82C54 chip, which is also not installed. See schematics for more details.
5. DAC output connector **J9**. The Burr-Brown (Now TI) DAC2815AP chip can be added to the C6x_DSK_DigIO daughter card to add two channels of +/-10V DAC output to the DSP system. Add this connector after installing the DAC2815AP chip.

Pin#	Signal Name	Description
1	AGND	Daughter Card Analog Ground
2	DACOUT1	Digital to Analog Output 1
3	AGND	Daughter Card Analog Ground
4	DACOUT2	Digital to Analog Output 2
5	AGND	Daughter Card Analog Ground

DAC Output Connector (J9) (See Schematics to add the appropriate ICs to the daughter card for the DAC portion.)

6. Digital I/O header **J10**. This connector brings out the 2 digital inputs and 2 digital outputs supplied by the C6713DSK (DB_CNTL0-1 and DB_STAT0-1). It also brings out External Interrupt 5 and the three additional digital outputs not used by the parallel port interface J12.

Pin#	Signal Name	Description	Pin#	Signal Name	Description
1	DGND	DSP/Daughter Digital Gnd	2	DB_EINT5	DSP External Interrupt 5
3	DB_CNTL0	DSP Extra Digital Out0	4	DB_TINP1	DSP Timer1 Input Pin
5	DB_CNTL1	DSP Extra Digital Out1	6	DIGOUT5	Daughter Digital Output 5
7	DB_STAT0	DSP Extra Digital In0	8	DIGOUT6	Daughter Digital Output 6
9	DB_STAT1	DSP Extra Digital In1	10	DIGOUT7	Daughter Digital Output 7

Digital I/O Expansion Header (J10)

7. Jumper **J13**. Install to connect Pin 17 of the 25 pin Dsub connect to the DSP's External Interrupt 4. When connected to a PC's parallel port, Pin 17 is the parallel port's Control Pin #3.
8. Jumper **W1**. Connects AGND to DGND underneath the DAC2815 IC. This is required when installing the DAC portion of the Daughter Card.

3. "Quanser Standard" Interface/PWM Amplifier board.

This board serves three purposes.

1. To amplify the PWM signal from the C6x_DSK_DigIO daughter card to a level capable of driving the 24 V Pittman motor.
2. To route encoder signal inputs back to the C6x_DSK_DigIO daughter card.
3. To allow other Quanser experiments to be controlled by the Mechatronics Kit DSP.

Motor connections are made to the screw terminal block. Terminal 1 is positive and is connected to the red wire of the motor. Terminal 2 is negative and is connected to the black wire of the motor.

Encoders are connected to the polarized, locking headers labeled Encoder 1 and Encoder 2. Connections to these headers vary by experiment, so refer to the appropriate section of this manual for specific details.

The enable switch must be in the 'on' position for the amplifier to operate. The red LED indicates the amplifier is enabled when illuminated.

The RCA jacks and 5 Pin DIN sockets are Quanser proprietary connectors. They mate to analog and encoder cables used on all Quanser control system experiments. The Mechatronics Kit can be programmed to control any Quanser control challenge not requiring more than two analog outputs or two encoder inputs. The two RCA jacks provide +/-5V analog outputs from the Mechatronics PWM outputs. The two 5 pin DIN sockets facilitate two encoder channel inputs. Encoders for the Mechatronics Kit must be disconnected before connecting encoders from Quanser experiments to the 5 pin DIN sockets on this interface board.

The headers that interface with the C6xDSK_DigIO daughter card are connected by standard 10 conductor ribbon cables (ie: DigiKey M3AAA-1006J-ND). The pin outs of these headers are detailed on the next page.

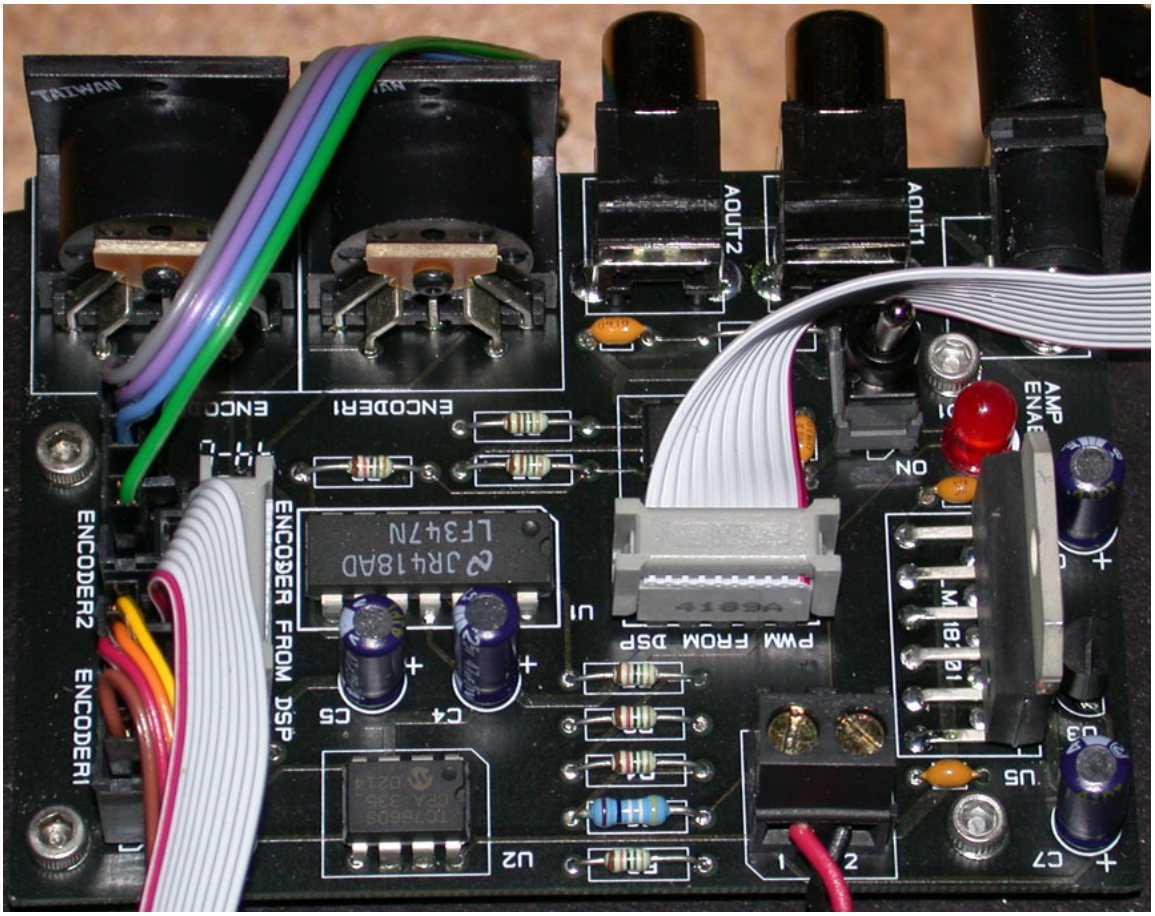


Figure 5-26 "Quanser Standard" Interface / PWM Amplifier

The connections to the encoder headers on the "Quanser Standard Interface / PWM Amplifier are the same as on the C6xDSK_DigIO daughter board and are detailed below.

Encoder 1			**Encoder 2**		
Pin#	**Signal Name**	**Description**	**Pin#**	**Signal Name**	**Description**
1	DGND	DSP/Daughter Digital Gnd	2	DGND	DSP/Daughter Digital Gnd
3	NC	No Connect	4	NC	No Connect
5	ENC1CHA	Encoder 1 Channel A	6	ENC2CHA	Encoder 2 Channel A
7	5V	DSP/Daughter Power	8	5V	DSP/Daughter Power
9	ENC1CHB	Encoder 1 Channel B	10	ENC2CHB	Encoder 2 Channel B

Connections for Optical Encoder Inputs 1 and 2

The PWM connections to the "Quanser Standard" Interface / PWM Amplifier are the same as on the C6xDSK_DigIO daughter board and are detailed below.

PWM1			**PWM2**		
Pin#	**Signal Name**	**Description**	**Pin#**	**Signal Name**	**Description**
1	DGND	DSP/Daughter Digital Gnd	2	DGND	DSP/Daughter Digital Gnd
3	5V	DSP/Daughter Power	4	5V	DSP/Daughter Power
5	NC	No Connect	6	NC	No Connect
7	NC	No Connect	8	NC	No Connect
9	PWM1	PWM Output 1	10	PWM2	PWM Output 2

Connections for PWM Outputs 1 and 2


QUANSER
INNOVATE. EDUCATE.