The HIAPER Microwave Temperature Profiler (MTP) Design Package

MJ Mahoney and Richard Denning

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109-8099

MJM: (818)-354-5584
RFD: (818)-354-2795
FAX: (818)-393-0025
Email: Michael.J.Mahoney@jpl.nasa.gov
Richard.F.Denning@jpl.nasa.gov

Version: 1.05
Date: July 15, 2010

Preface

This is a living document and therefore under constant revision. Its purpose is to document everything that we know about the HAIPER Microwave Temperature Profiler (MTP) so as to facilitate FAA certification of the instrument.

Revision History

Version	Date	Comment
1.00	Jun 15, 2006	Initial release
1.01	Jun 17, 2006	Update after PDR
1.02	Dec 15, 2006	Added appendices with controller board drawings and wiring
1.03	Jan 29, 2007	As presented as CDR
1.04	Jun 06, 2007	Updated SolidWorks Figures and added more of them
1.05	Jul 15, 2010	Updated control/wiring drawings and writing

Table of Contents

Acronyms and Abbreviations

AC	alternating current
COTS	commercial off the shelf
DC	direct current
DFS	Design and Fabrication Services
DIP	dual inline package
DMT	Droplet Measurement Technologies
DSB	double side band
DSM	Data Sampling Module
EOL	Earth Observing Laboratory
FAA	Federal Aviation Administration
FPC	Flat Panel Computer
GHz	gigahertz
HDPE	high-density polyethylene
HIAPER	High-performance Instrumented Airborne Platform for Environmental Research
ID	inside diameter
IF	intermediate frequency
IO	input and output
IWGADTS	Interagency Working Group for Airborne Data and Telecommunications Systems
JPL	Jet Propulsion Laboratory
LED	light emitting diode
LO	local oscillator
MHz	megahertz
MTP	Microwave Temperature Profiler
NASA	National Aeronautics and Space Administration
NCAR	National Center for Atmospheric Research
NTP	Network Time Protocol
OMT	orthomode transducer
PCB	printed circuit board
PIC	programmable integrated circuit
RF	radio frequency
RTD	resistive temperature detector
SPDB	Secondary Power Distribution Box
SPDDB	Secondary Power Distribution Drop Box
SPI	Serial Peripheral Interface
UCAR	University Corporation for Atmospheric Research

1 MTP Description

1.1 Background

The first airborne Microwave Temperature Profiler (MTP) flew aboard a NASA CV-990 in January 1979 and weighed more than 200 kg (Gary, 1981). A ~30 kg instrument was built in 1985 to fly on the NASA ER-2 (Denning et al., 1989). Current MTPs weigh ~10 kg, they occupy locations on the aircraft not normally used by other instruments, they can run unattended, and they make measurements at microwave frequencies which are not affected by the presence of clouds. To date, MTPs have flow on 53 field deployments, and have accumulated 4696 flight hours on 824 flights.

1.2 Principal of Operation

The MTP is a passive microwave radiometer that measures, at several frequencies, the thermal rotational-line emission from oxygen molecules in the Earth's atmosphere. The instrument uses a scanning mirror to view ten elevation angles from the zenith to nadir in the flight direction (Denning et al. 1989). The measured microwave brightness temperatures, or observables, are then converted to a vertical temperature profile along the flight track by using a statistical retrieval procedure (Strand and Westwater 1968) with Bayesian aspects. To do this, a forward radiative transfer calculation must first be made. Hundreds of radiosondes, which are representative of the season and location of the actual measurements, are used to calculate the expected brightness temperatures for each radiosonde temperature profile. These calculations must be done for all flight levels. A linear multiple regression is then used to statistically relate the expected brightness temperatures (from the forward radiative transfer calculation) to physical temperature profiles (from the hundreds of radiosondes). This results in a set of retrieval coefficients that can be used to convert actual measured brightness temperatures to a physical temperature profile. Retrieval accuracy is improved by using radiosondes that are flown by as templates to select the others needed to calculate a set of retrieval coefficients. An information-theory-based metric then compares the measured observables to the average calculated observables for each retrieval coefficient set, and determines which set (or pair of sets) of retrieval coefficients to use for the temperature profile retrieval. A temperature profile has a vertical resolution of ~150 m near flight level, and degrades with distance from the aircraft. More information on the theory of MTP measurements can be found in **Appendix A**.

Figure 1-1. The NASA DC-8 MTP mounted in an unused window.

1.3 Hardware Description

It was originally proposed that the HIAPER MTP would be a close copy of the current DC-8 MTP (**Figure 1-1**). All of the electronics for this instrument are mounted on the back of a window blank. The MTP is controlled by a rack-mounted, flat panel computer

located on a bulkhead behind the forward starboard Exit door. Previously, all MTPs consisted of two components: a Sensor Unit, which contained the receiver and related electronics, and a Data Unit, which controlled the Sensor Unit and recorded data from it (**Figure 1-2**). The DC-8 MTP simplification of having the hardware in a single enclosure was made possible by the use of several micro-controllers and "smart" peripheral chips to replace many standard logic and analog integrated circuits. For example, the integration timer/counter that reads the output of the radiometer voltage-to-frequency converter previously required nine 74HC DIP packages of 14 to 16 pins. It was replaced by a single 18-pin dual inline package (Microchip 16F628) and a ceramic resonator.

Figure 1-2. The MTP Sensor Unit (left) and Data Unit (right) used on all platforms except the DC-8.

The HIAPER MTP shown in **Figure 1-3** weighs 18.2 kg (including the fairing and Droplet Measurement Technologies (DMT) canister, but not the cabin control computer and cables.) We had proposed to mount the HIAPER MTP in a window similar to the DC-8 MTP shown in **Figure 1-1**. However, concerns about the cost and time involved for anti-icing certification led us to consider other options. The one that we decided on was to mount the MTP on the front of a DMT canister, far enough forward to have the scanning mirror beam clear the leading edge of the GV wing. The canister itself is attached to one of three wing hard points by a strut. The MTP requires a fairing with a microwave-transparent high-density polyethylene (HDPE) window to protect it from the elements.

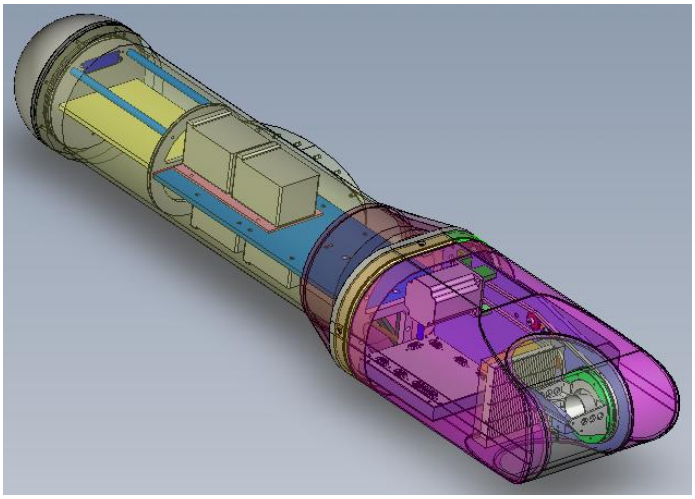


Figure 1-3. The HIAPER MTP is mounted on the front of a 6-inch diameter (ID) DMT canister.

Figure 1-4 shows a block diagram of the current DC-8 MTP on which the HIAPER instrument is based. A flat panel computer (FPC) running under Windows 2000 controls the operation of the radiometer, records the data, performs the analyses, and displays the temperature profiles as they are retrieved. Between the MTP and the FPC there is an RS-422 serial connection allowing the computer to be located anywhere in the aircraft cabin. Since all control functions are in one computer with access to and from the Ethernet, the

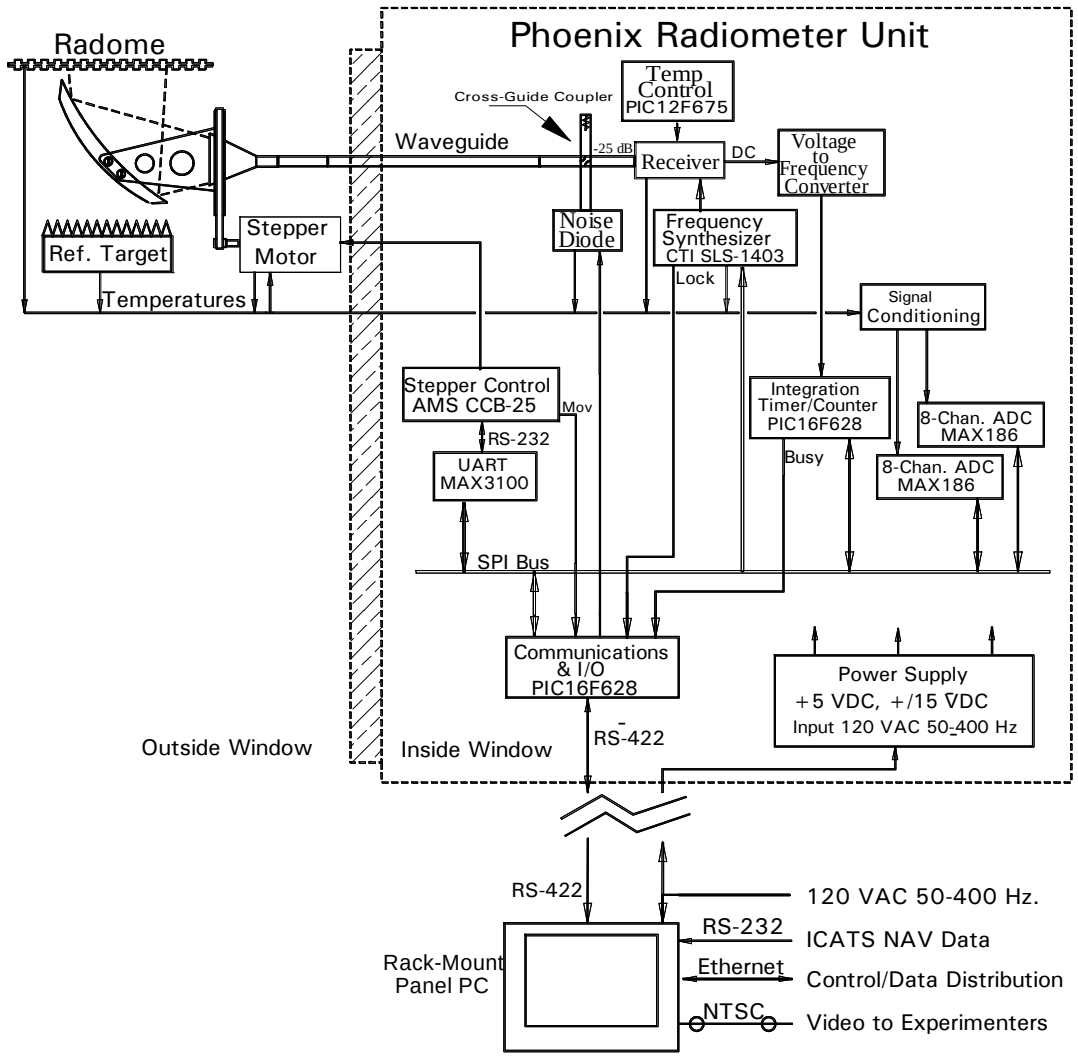


Figure 1-4. A block diagram of the current DC-8 MTP. The HIAPER MTP is very similar to this, but it is mounted on a DMT canister instead of a window.

entire scan and frequency tuning sequence is available to the MTP experimenter, if present or if there is a satellite connection from the ground, for modification during flight. Parameters such as mounting offsets, integration time, frequencies and scan angles are contained in a configuration file that can be changed at any time. Other experimenters can obtain numerical data or temperature profile images during flight over the GVs display and user data traffic network. If an MTP experimenter is on board or has access via a satellite connection, he/she can control the instrument or download data files, or even edit programs on the FPC.

There are three minor changes that were implemented for the HIAPER MTP compared to the DC-8 MTP. Because it was a prototype, the current DC-8 MTP digital circuitry was implemented on a wire-wrap board rather than a printed circuit board (PCB). The DC-8 wire-wrap design has been converted to a multi-layer PCB. This will allow us to inexpensively make spare PCBs. The PCB design was optimized for the HIAPER MTP.

The second minor change was the following. The HIAPER MTP uses the Network Time Protocol (NTP), which has a latency of 1 ms, to synchronize the MTP measurements to the other measurements on the aircraft. There was no time synchronization on the DC-8.

The third minor change involved changing the double side band (DSB) receiver to operate with the IF at base band instead of 320 MHz (as is case for all our existing instruments). This simplifies the data analysis and improves the quality of the retrievals.

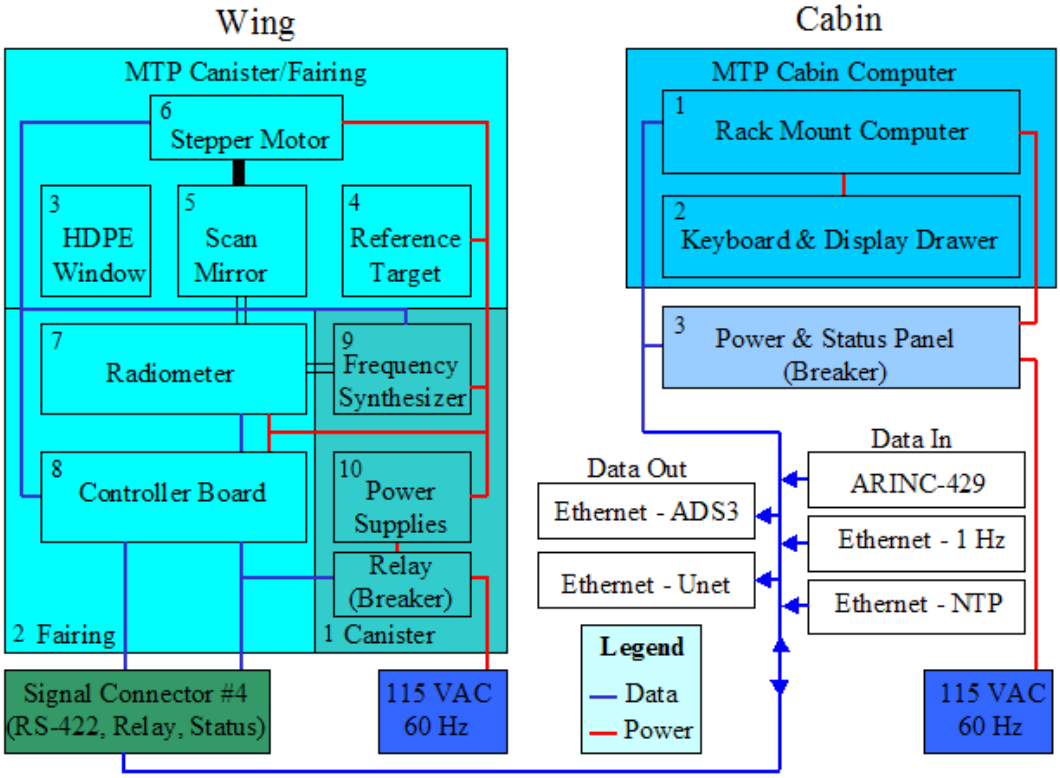


Figure 2-1. Block diagram of the HIAPER MTP, which shows major components, and data and power connectivity. Numbers indicated in the upper or bottom left hand corner of boxes map to sub-sections in Section 3 of this document.

2 The HIAPER MTP

2.1 Overall Architecture

The overall architecture of the HIAPER MTP is shown in **Figure 2-1**. As shown on the left, most of the MTP components will be located inside a fiberglass *Fairing (2)* on the front of a six-inch-inside-diameter aluminum *Canister (1).* The fairing will contain a *HDPE Window (3)* for viewing through using a *Scan Mirror (5)* inside the fairing, which is driven by a *Stepper Motor (6).* In addition to viewing ten different elevation angles from near-zenith to near-nadir in the flight direction, the *Scan Mirror* also views a *Reference Target (4)* for gain calibration purposes. The RF signal received by the *Scan Mirror* connects to the *Radiometer (7),* which is mounted on a thermally isolated temperature-controlled plate. A microwave signal from a *Frequency Synthesizer (9)* is fed into the *Radiometer* to down-convert the RF signal to baseband. The *Frequency Synthesizer* and *Power Supplies (10)* are located in the *Canister* itself, which is mounted on a wing hard point strut.

The detected baseband signal is processed by a voltage-to-frequency converter (VFC), and that signal is sent to the *Controller Board (8),* which communicates with the MTP

Cabin Computer (MCC). The *Controller Board* also receives and processes commands from the MCC and sends data back to the MCC. The MCC records this data and retrieves real time temperature profiles. The raw data is also be recorded on the ADS3.

Small PIC-based *Temperature Controller Boards* are used at four locations in the MTP to maintain stable temperatures and to prevent condensation on descent after cold soaking. These locations are the *Reference Target*, the *Radiometer Plate*, *Power Supply/ Synthesizer Plate*, and the *Controller Board* enclosure. The controllers all operate near 40 C.

2.2 Connectivity

2.2.1 Wing

The connection between the MTP and MCC is done via wing **Signal Connector #4**, with one of the ASCII cables used for RS-422 communications, and two pairs of the other conductors used for a power relay signal and power status from the instrument. The MTP is powered from one of the wing stores SPDBs using 115 VAC 60 Hz power via a relay controlled by a toggle switch on the cabin *Power and Status Panel (3)*. The MCC is also powered by 115 VAC 60 Hz power from either a SPDB or a SPDDB. MTP cabling is provided to connect power from the wing stores, and to connect to wing **Signal Connector #4**. Cabling will also be provided to connect to a cabin SPDB power source, and from the cabin end of **Signal Connector #4**. Section 4 provides a detailed power load analysis. We now provide a more detailed description of the MTP cabin and instrument connectivity shown in **Figure 2-1**.

2.2.2 Cabin

As shown on the right side of **Figure 2-1**, the MTP is controlled by a 1U rack-mounted *MTP Cabin Computer (1),* which has a user-interface provided by the *Keyboard and Display Drawer (2).* Power for the MCC is provided from the *Power and Status Panel (3),* which receives power from a Secondary Power Distribution Box (SPDB) or Secondary Power Distribution Drop Box (SPDDB).

The *Power and Status Panel* has one breaker to turn the computer on, and a toggle switch to operate a relay to turn the instrument on. (It is done this way because there may be a need to access to the computer, for example to offload data, without turning the instrument on.) Power connectivity is shown by red lines on the right side of **Figure 2-1**.

An LED on the *Power and Status Panel* indicates that power is available to the rack. Another LED indicates that power is available at the MTP canister. The condition in which power is available in the canister, but the 'Operate' toggle switch has not been turned on, is the 'Standby' mode. In Standby mode only canister temperature controllers are operating.

The cabin MTP non-power-connectivity is shown by the blue lines on the right side of **Figure 2-1**. Control signals from the computer are sent and data returned via an RS-422 interface between the MCC and the MTP. Data from the MTP and other on-board sources are combined into data files recorded on the MCC for real-time analysis and display. The raw data is also sent to the ADS3 to be recorded for backup. The MCC performs real time

temperature profile retrievals. Images of these profiles (and other data products) are available on the user data traffic network.

To properly point the MTP, real time pitch and roll are required. These are acquired from the 1 Hz IWGADTS ASCII data stream. Its 2-3 second latency is much better than the current 10-15 second latency of other existing MTPs, which have acceptable pointing performance. If there is a need to reduce the latency this could be done by receiving an ARINC-429 signal from the Honeywell Laseref IV. Time synchronization is obtained from the Ethernet NTP signal.

2.2.3 Instrument

Top level instrument power connectivity is shown as red lines on the left side in **Figure 2-1**, while the control/data connectivity as shown as blue lines. In addition to wiring, there is waveguide connectivity between the receiver components, and from the frequency synthesizer (local oscillator) to the mixer in the receiver. Detailed information on the instrument wiring can be found in the appendices.

There are two classes of measurements in the MTP: (1) radiometric, that is, data used directly in calculating brightness temperatures, which includes the radiometer output as well as the temperatures of certain key components, and (2) engineering values such as power supply voltages, motor temperature, power supply temperature, etc. that are used to evaluate and diagnose the health of the instrument.

3 HIAPER MTP Hardware Component Details

3.1 MTP Canister Hardware

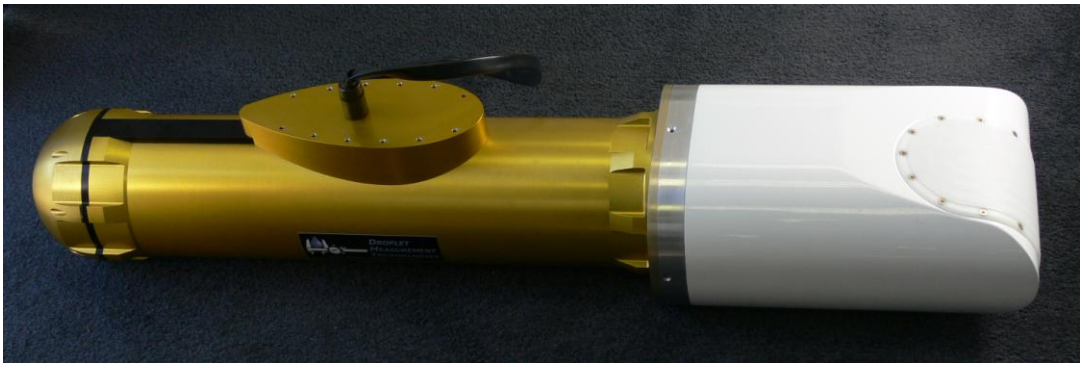


Figure 3-1. HIAPER MTP fairing (white), DMT canister (gold) and HDPE window (white).

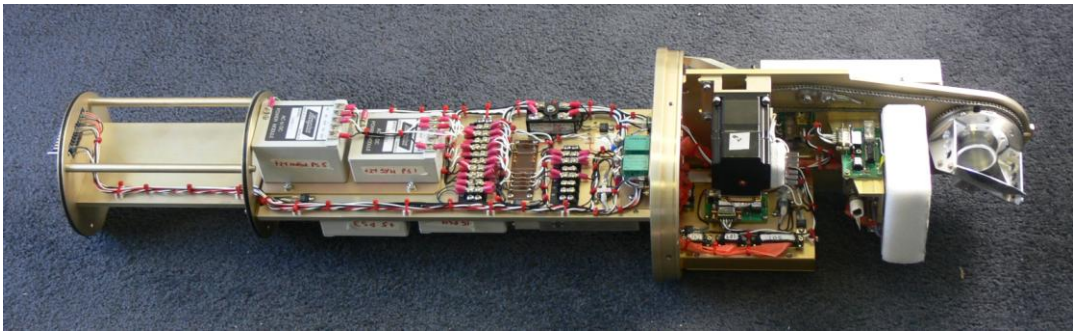


Figure 3-2. The HIAPER MTP without the DMT canister and fairing.

3.1.1 MTP Canister

Source: Droplet Measurement Technologies, Inc.

Material: Aluminum

Description: A Droplet Measurement Technologies (DMT) 6-inch diameter (ID) canister, but with a simplified construction process to reduce cost. The completed HIAPER MTP is shown in **Figure 3-2**.

3.1.2 MTP Fairing

Source: Zivko Aeronautics
Material: Fiberglass
Description: The fairing protects the MTP hardware from the environment.

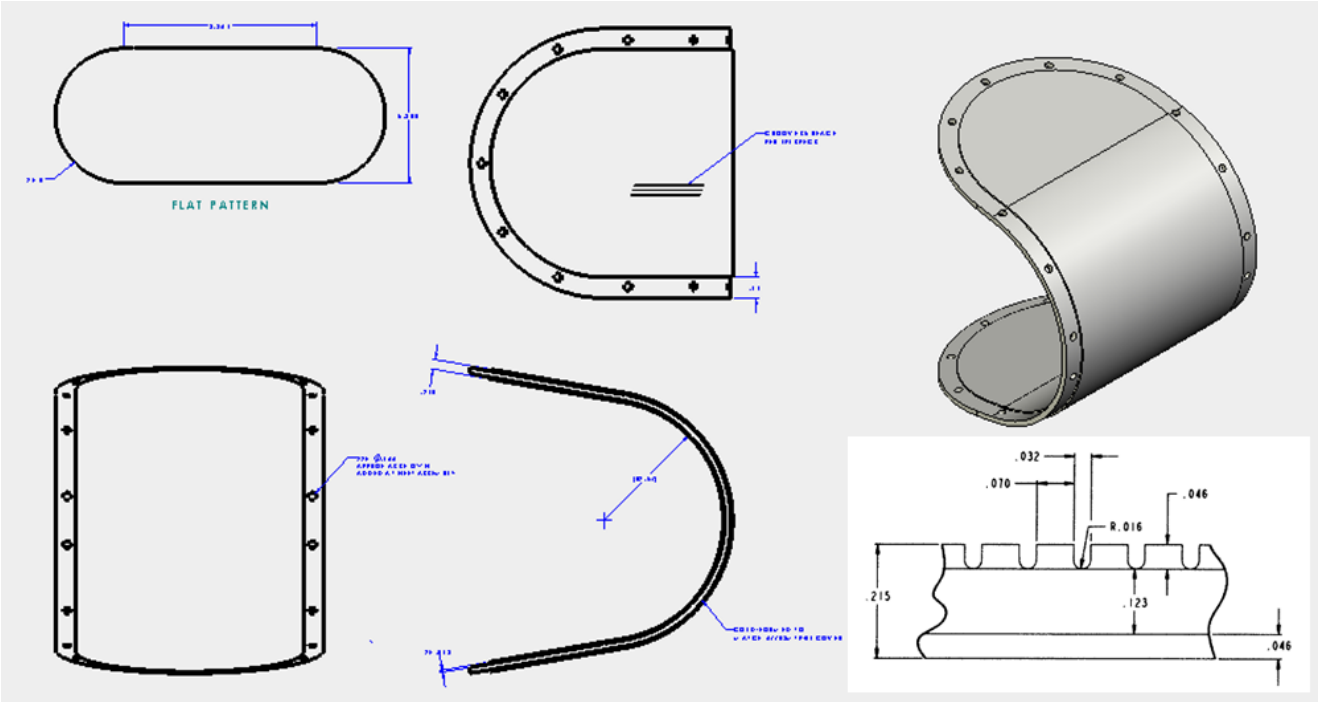


Figure 3-3. The HIAPER MTP window with orthogonal anti-reflection groove detail inset.

3.1.3 HDPE Window

Source: NCAR/EOL/DFS
Material: High-density polyethylene (HDPE)
Description: HDPE is microwave transparent and allows the MTP scan mirror to view outside the fairing. The window has groves machined on each side which are orthogonal, to serve as an anti-reflection coating. The groves are separated by 0.102 inches, are 0.032 inches wide and are 0.046 inches deep. The thickness of the machined HDPE sheet is 0.215 inches.

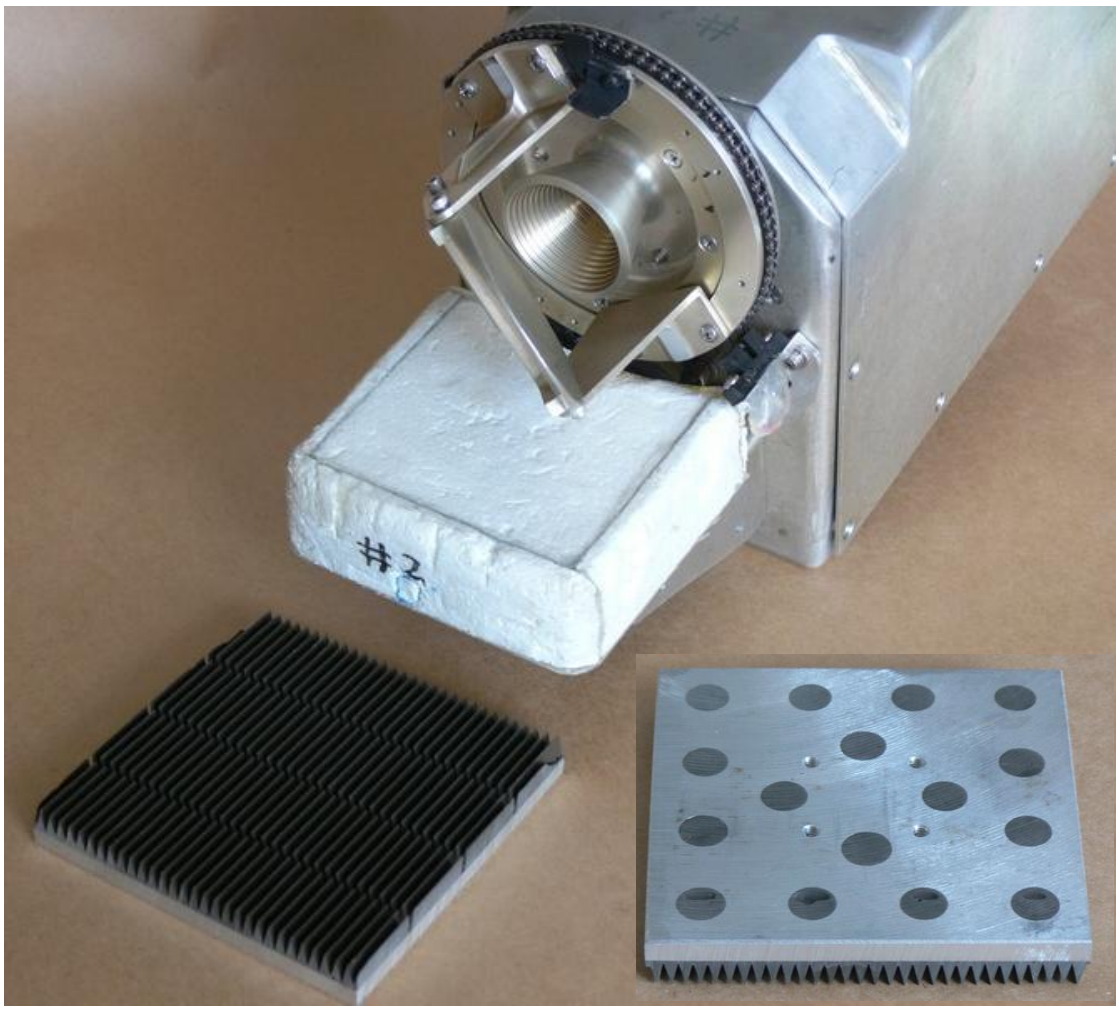


Figure 3-4. ER-2 Sensor Unit #2 is shown enclosed in ¼" thick Styrofoam to control thermal gradients and to provide a long thermal time constant. A top view of the target is shown in front of the enclosed target, and the metal backing is shown to the right.

3.1.4 Reference Target

Source: JPL
Material: Carbon-ferrite mounted on an aluminum plate
PLASTAZOTE LD15 and ROHACELL foam insulation
Power: 115 VAC 60 Hz 0.5 A max for heater
Description: The reference target temperature is used in combination with the outside air temperature to calculate the gain of the MTP. The target is 4 inches x 4 inches and about 1 inch deep, and it is surrounded by ¼ inch foam insulation. All our existing MTPs use polystyrene foam (Styrofoam) to thermally insulate the target. As discussed in Section 11 we considered other materials for insulation because of the flammability issues with using polystyrene. We found a new polyethylene foam (PLASTAZOTE LD15) which has better electrical properties than Styrofoam, and initially used it on the front face of the target. However, it did not hold up and was replaced with Styrofoam. ROHACELL foam insulation was used on the other faces as it meets FAA flammability requirements.

Figure 3-5. Scan mirror, corrugated feed horn, chain drive, and orthomode transducer. Note that the sprocket for the chain drive is mounted outside of the original belt drive sprocket. This was done so that the reference target could be mounted closer to the antenna. Previously, it was mounted in place of the original belt drive sprocket.

3.1.5 Scan mirror

Source: JPL Microwave Sounder Unit (MSU) satellite spare

Material: Aluminum

Description: The scan mirror is used to view ten different elevation angles from near zenith to near nadir in the flight direction. It also scans to a reference target for calibration purposes. The scan mirror has a beam width of 7.5° FWHM, and re-directs the beam 90° into a conical, corrugated feed horn.

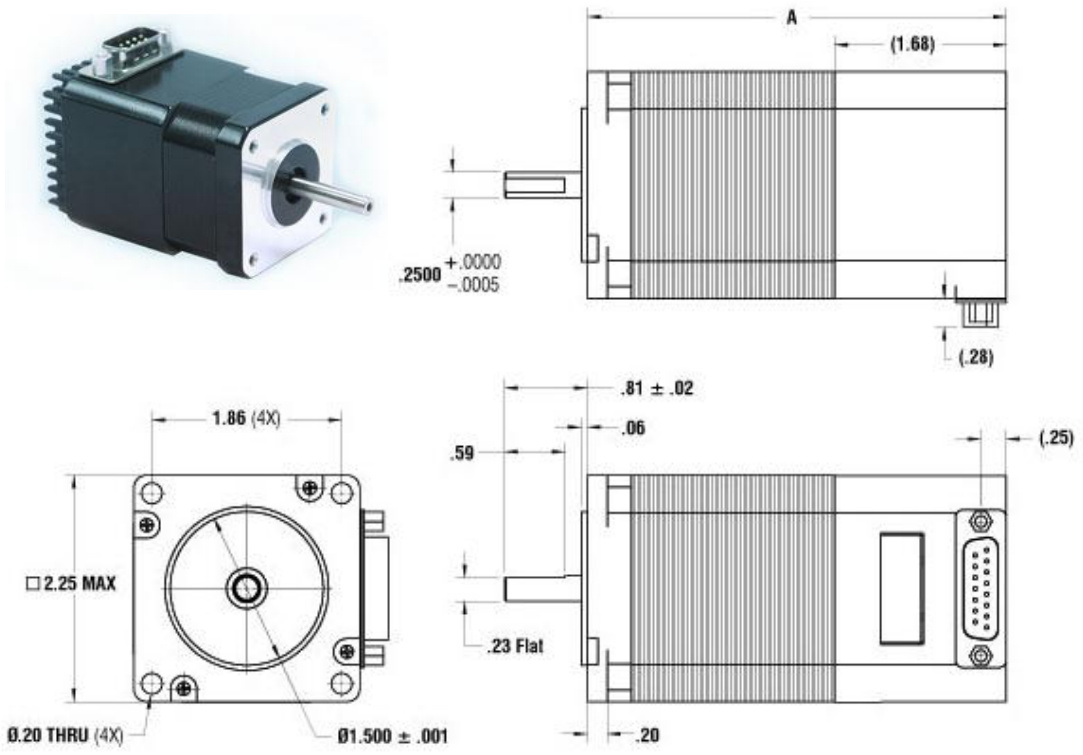


Figure 3-6. Lin Engineering CE-5718M stepper motor.

3.1.6 Stepper motor

Source: COTS from Lin Engineering
Part Number: CE-5718L-01PD-20 with 200 cpr encoder
Power: 12-40 VDC @ 3 A max, will operate at +24 VDC, 2 A max
Description: The stepper motor is used to point the MTP scan mirror. It has a shaft encoder to verify position, and an integrated controller that operates via an RS-485 interface.

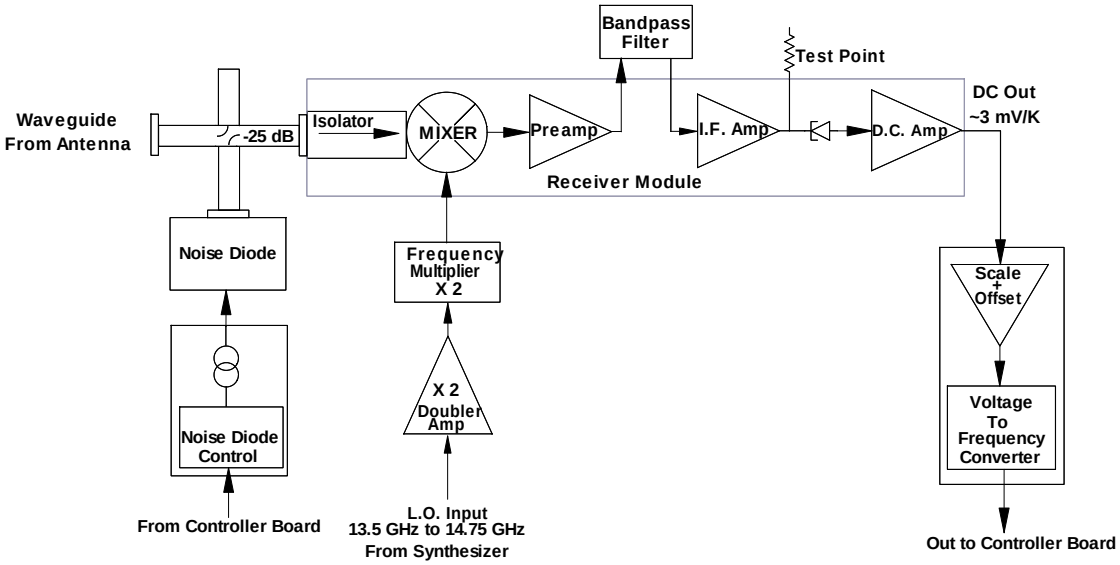


Figure 3-7. Schematic layout of the hardware on the HIAPER MTP radiometer plate.

3.1.7 Radiometer

Source: Custom built by Spacek Labs, Inc, Santa Barbara, CA
Material: Mostly metal
Power: 15 VDC @ 200 mA max
Description: The signal received from the corrugated feed horn on the scan system is converted into orthogonal linear polarizations in an orthomode transducer (OMT) attached to the base of the feed horn. A load is put on one linear polarization and the other is conveyed in waveguide to the radiometer. The radiometer consists of a cross-guide coupler for injection of a noise diode calibration signal, followed by an isolator to prevent local oscillator (LO) signal leakage. Next is a double-side-band biased mixer, which mixes the incoming radio frequency (RF) signal with the LO signal. The LO signal is derived from the frequency synthesizer, by first doubling the signal in an active doubling amplifier, which is followed by a passive doubler. Unlike our current mixers, the HIAPER mixer down converts to base band with the lowest frequency ~10 MHz. This is followed by amplification and an intermediate frequency (IF) filter to select the pass band which is nominally ~200 MHz, but can be larger or smaller depending on the application simply by changing the IF filter. The IF signal is detected and processed by a voltage-to-frequency converter, and then sent to the *Controller Board* for counting. **Section 9** provides more detail on how we arrived at this particular radiometer architecture, which is essentially the same as what we currently employ in our other instruments. We had wanted to use another architecture, which had definite advantages, but the performance would not have been as good.

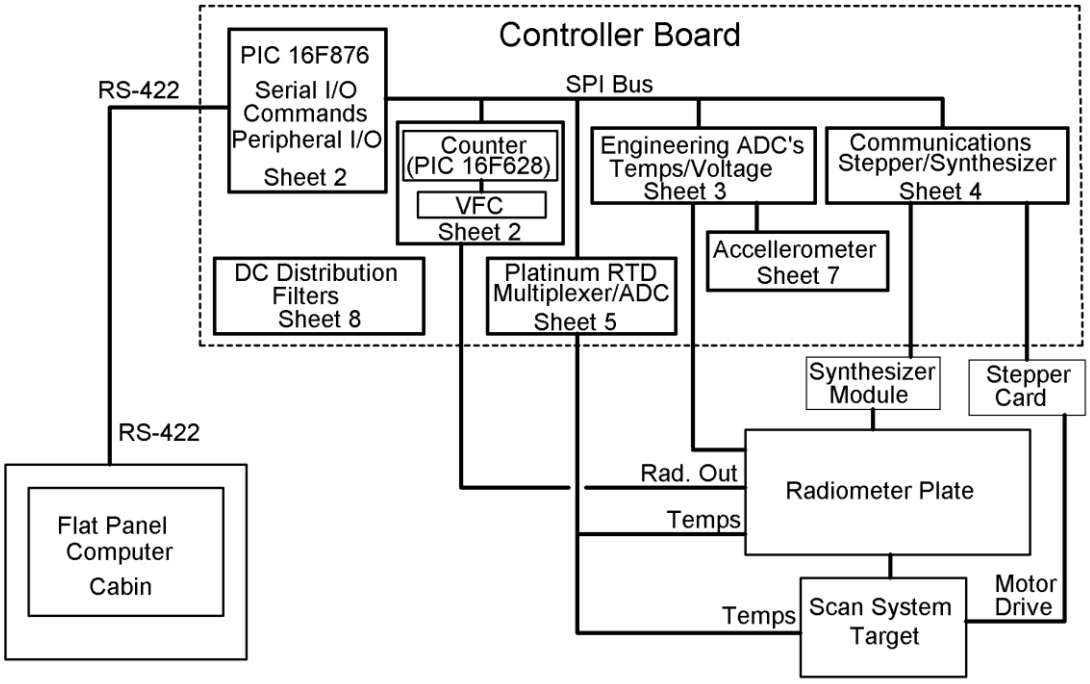


Figure 3-8. HAIPER MTP controller board with attached hardware.

3.1.8 Controller Board

Source: Multilayer PCB board designed by JPL and manufactured by a third party.
Description: A single multilayer printed circuit board is used to control all of the MTP hardware. It is 4 inches by 5 inches in size, and uses surface mount devices. The *Controller Board* has two programmable integrated circuits (PICs) performing different functions. As shown in **Figure 3-8**, one PIC controls the asynchronous RS-422 IO with the MTP Cabin Computer, and communicates with the other hardware over a serial peripheral interface (SPI) bus. On the SPI bus, a second PIC counts the signal from the *Radiometer* voltage-to-frequency converter (VFC).

The *LO* frequency synthesizer is controlled directly as a device on the SPI bus, and the stepper motor is controlled via an RS-485 interface through an SPI UART. The SPI bus also controls the multiplexing and A/D conversions needed to read resistance data from six platinum resistive temperature devices (RTDs). The RTDs are placed on instrument components requiring stable temperature measurement for data processing purposes, such as the *Reference Target*, *HDPE Window*, and components in the *Radiometer*. (Eight channels are available on the RTD multiplexer of which two are used for precision calibration resistors, which have an extremely low temperature coefficient.) Less critical temperatures and voltages are measured on a separate 16-channel engineering data multiplexer. This multiplexer also receives data from a vertical accelerometer. **Section 13** in the appendix shows a complete layout of the *Controller Board*.

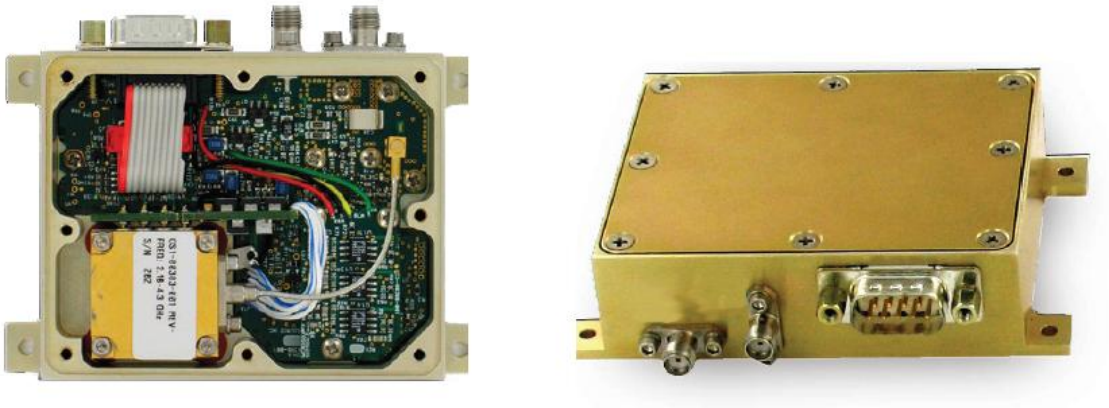


Figure 3-9. Giga-tronics/Microsource, Inc., Frequency synthesizer.

3.1.9 Frequency Synthesizer

Source: COTS from Giga-tronics/MicroSource, Inc.
Part Number: Ultra-Low Noise Synthesizer, 12-16 GHz, MSI
Model No. SNP-1216-520-01
Material: Various
Power: 24 VDC @ 310 mA (500 mA max)
Description: This wide band YIG-tuned synthesizer has 1 Hz resolution and can be tuned for an output frequency from 12.0 to 16.0 GHz. The power output across the band is 17 dBm. The output from the synthesizer is amplified in a doubling amplifier. It in turn drives a passive doubler, which generates the final local oscillator (LO) signal needed for the MTP heterodyne receiver. The size of the package is 5.0” x 5.5” x 1.6”.

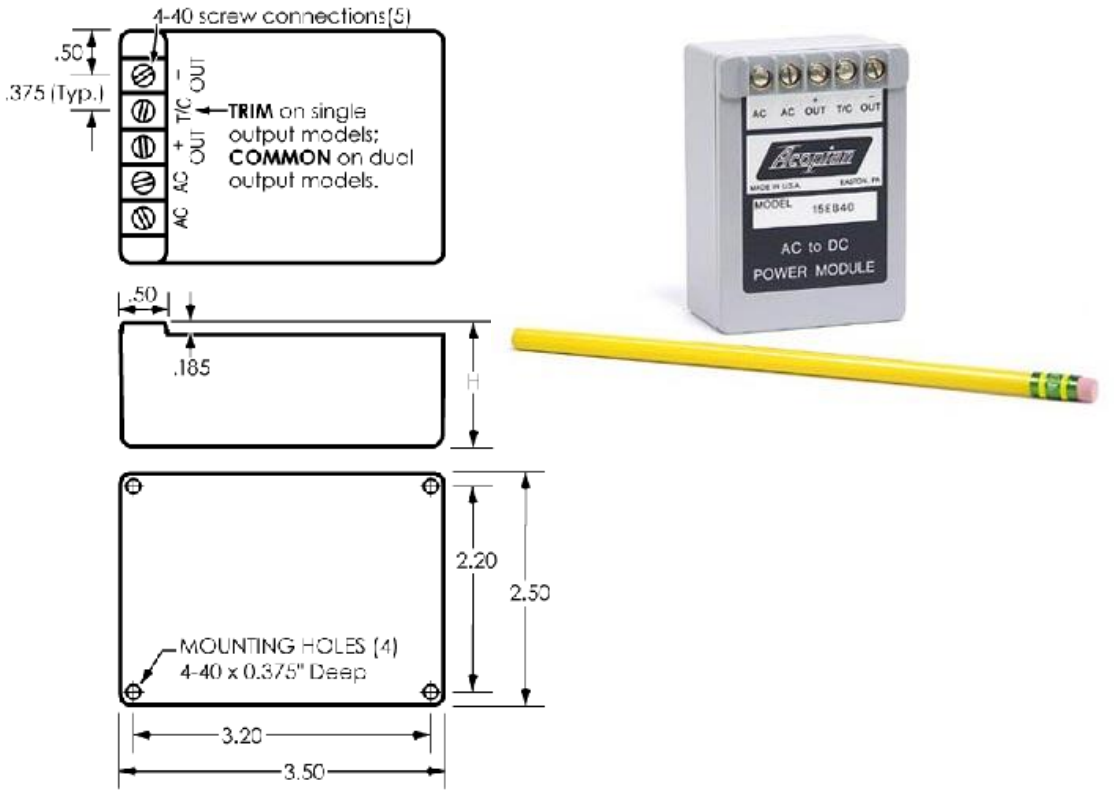


Figure 3-10. A typical Acopian power supply. All the DC power supplies have the same footprint (lowest drawing), but they have different heights (H).

3.1.10 Power Supplies

Source: COTS from Acopian

Part Numbers	Detail	PS Number
24WB210	+24 V @ 2.10 Amp max, switching regulated	5
15WB330	+15 V @ 3.30 Amp max, switching regulated	1
15WB200	+15 V @ 2.00 Amp max, switching regulated	2
15EB40	–15 V @ 0.40 Amp max, linear regulated	4
5EB100	+5 V @ 1.00 Amp max, linear regulated	3
5EB100	+5 V @ 1.00 Amp max, linear regulated	6

Description: Six (6) modular power supplies are used to convert aircraft 115 VAC 60 Hz aircraft power to the DC voltages needed for the MTP. The PS Number above is the power supply number on the wiring diagram on page 66. The +24 V switching regulated supply is used for the *Stepper Motor*. There are two +15 V switching regulated supplies: one is used for the *LO* frequency synthesizer and the noise diode, and the other is used for the IF and LO amplifiers. A -15V linear regulated supply is used for analog circuits in the receiver and the *Controller Board.* There are two +5V linear regulated supplies: one for the *Controller Board*, and a separate one for the *Temperature Control System* to provide logic power when the MTP is in Standby mode. All these supplies can handle 60 or 400 Hz, and they all have the same footprint, but vary in height.

In addition to the DC power supplies just described, the MTP also uses 115 VAC 60 Hz for heater power. As mentioned above, heaters controlled by *Temperature Controller Boards* are used to control the temperature of the *Radiometer* box, the *Reference Target*, the *Power Supplies/Synthesizer Plate* and the *Controller Board* enclosure. The heaters improve performance by maintaining temperature stability and they prevent condensation on descent after cold soaking. These heater loads represent another 4 x 115VAC @ 0.5 Amp (max) added to the total power requirement. A detailed power load analysis is provided in **Section 4**.

3.2 MTP Cabin Hardware

In order to minimize the amount of space that the MTP control computer would require, we used a rack-mounted computer with a display included in a separate keyboard drawer. As a result only 5.25 inches of rack space (3U) are required to mount the *MTP Control Computer* (1U), the *Keyboard and Display Drawer* (1U) and *Power and Status Panel* (1U).

Figure 3-10. MTP General Technics Cabin Computer 1U enclosure.

3.2.1 Rack-mounted Computer

Source: COTS from General Technics.
Model: 1U ATX-style case with MB796 Intel SE3000AH system board
Description: This computer controls the MTP in the DMT canister, records the data from the MTP, and performs real time temperature profile retrievals. The computer is rack-mountable and is 1.75" high (1U). It has a front accessible CD-DVD RW drive, and two front and two back USB-2 interfaces. There are two internal 160 GB hard drives configured as RAID 1 (exact images). There is a 300 W ATX power-supply.

Figure 3-10. HIAPER MTP 1U keyboard and display drawer.

3.2.2 Keyboard and Display

Source: COTS from I-Tech Company
Model: BJK117-8Le
Description: A rack mountable keyboard tray with a foldaway display. The keyboard is a 104-key Windows keyboard with short travel keys. The display is 17" diagonal with 1280x1024 resolution and has a 120 degree viewing angle with 350:1 contrast ratio. Dimensions are 19" wide x 1.75" (1U) high x 24.6" deep (442 mm x 43 mm x 625 mm), and it weighs 20.9 lb (9.5 kg). The enclosure is heavy-duty steel with ball bearing rack slides. The drawer is lockable.

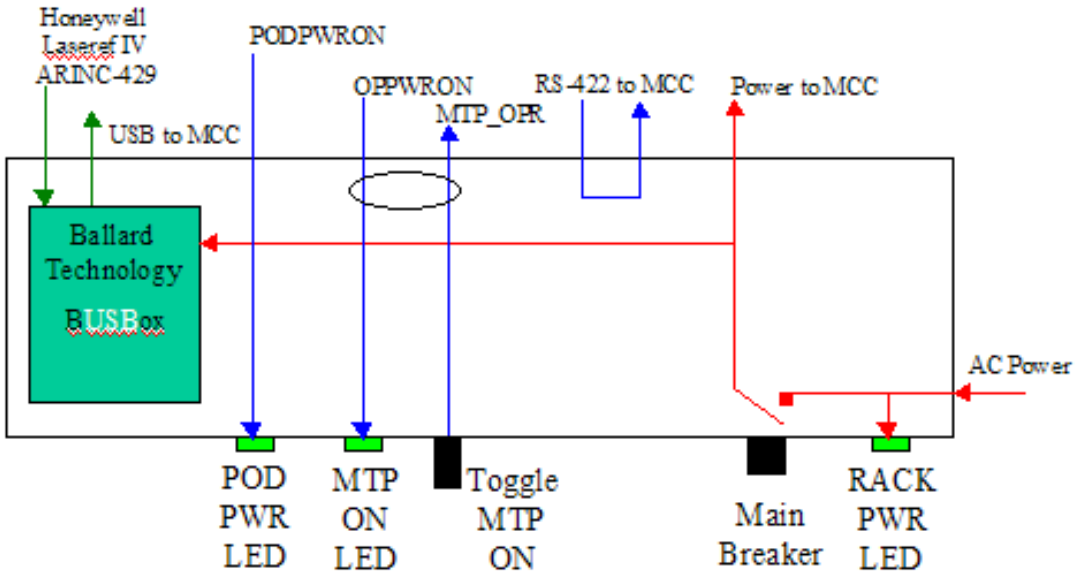


Figure 3-11. Layout of the Power and Status Interface

3.2.3 Power and Status Panel

Source: JPL build

Description: The power and status panel operates as follows. When AC power is available to the rack, the RACK PWR LED will come on. At this time the Main Circuit Breaker can be depressed (engaged) to provide power to the rack-mounted MTP Cabin Computer (MCC). (This action could also provide power to the Ballard Technology ARINC-429 to USB converter, which can provide data (pitch and roll) from the Honeywell Laseref IV inertial reference system to the MCC. As mentioned above, we decided to use the 1 Hz IWG feed instead of ARINC-429.) At this point only the MCC is operational.

When wing power from the wing store is available to the MTP, the pod power on (PODPWRON) signal line will turn on the POD PWR LED. The MTP has a 5-volt power supply that is activated by the presence of wing power. Although the instrument is not yet on, wing power does turn on the four temperature controllers in the instrument. The controller logic is run from one of the two 5-v supplies although the heater resistors use the AC power. The presence of the 5-v power is what energizes the PODPWRON signal. When the POD PWR ON LED is lit, the instrument can be turned on. This is done by switching the MTP ON toggle switch. This switch sends the MTP operate (MTP_OPR) signal to a relay on the *Power Supply and Synthesizer Plate* in the instrument. When the relay turns on, it returns the operate power on (OPPWRON) signal back to the Power and Status Panel, which turns on the MTP ON LED. If for any reason the instrument fails to cycle properly, the MTP ON LED will turn off.

3.3 Summary of MTP Components Details

The **Table 3.1** lists the major MTP components, referring to the section (Sec) in which they were discussed above, whether they are commercial off the shelf (COTS), and their source.

Sec	Component	COTS?	Source
3.1.1	DMT Canister	N	DMT, Boulder, CO
3.1.2	Fairing	N	Zivko Aeronautics, Guthrie, OK
3.1.3	HDPE Window and Mounting Hardware	N	EOL/DFS build
3.1.4	Reference Target	N	JPL build
3.1.5	Scan Mirror	N	JPL MSU Spare
3.1.6	Stepper Motor	Y	Lin Engineering, Santa Clara, CA
3.1.7	Radiometer	N	Spacek Labs, Inc., Santa Barbara, CA
3.1.8	Controller Board	N	JPL build
3.1.9	Frequency Synthesizer	Y	Giga-tronics, San Ramon, CA
3.1.10	Power Supplies	Y	Acopian, Inc., Easton, PA
3.2.1	Rack-Mount Computer	Y	General Technics, Ronkonkoma, NY
3.2.2	Display & Keyboard	Y	i-Tech Company, Freemont, CA
3.2.3	Power & Status Panel	N	JPL build

Table 3-1. MTP COTS components.

4 Electrical Description

4.1 Circuit Protection

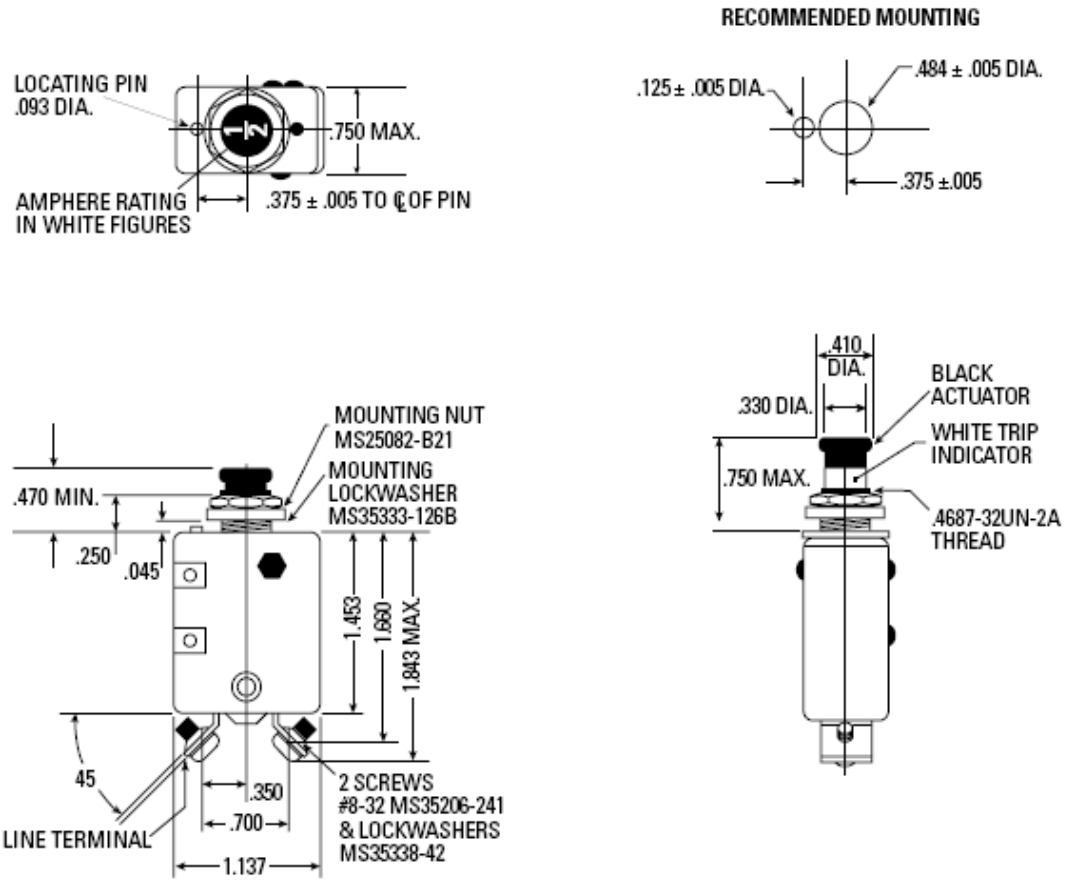


Figure 4-1. Mil Spec 3320 circuit breaker.

4.1.1 Wing

There will be two circuit breakers in the wing for the MTP 115 VAC power. One is for the heater circuit whenever wing power is available, and the other is for the main instrument power which comes on when the MTP Power On toggle switch is turned on at the MTP cabin *Power and Status Panel*. Both of these circuit breakers are Mil Spec. MS3320 5 A breakers (e.g., Eaton part number 1500-052-5) as shown in **Figure 4-1**.

4.1.2 Cabin

Power for the *MTP Cabin Computer* (1U Rack-mounted Computer, and 1U Keyboard and Display Drawer) is provided from the *Power and Status Panel*. 115 VAC power to the *Power and Status Panel* comes from a SPDB or SPDDB, and is protected by Mil Spec. MS3320 5 A breaker (e.g., Eaton part number 1500-052-5).

Acopian Model	PS No.	DC Voltage (volts)	DC Current (amps)	DC Power (watts)	115 VAC 60 Hz (amps)*
24WB210	5	+24	2.10	50.40	0.58
15WB330	1	+15	3.30	49.50	0.57
15WB200	2	+15	2.00	30.00	0.34
15EB40	4	-15	0.40	6.00	0.07
5EB100	3	+5	1.00	5.00	0.06
5EB100	6	+5	1.00	5.00	0.06
Total				145.90	1.67

*Assumes typical Acopian power-supply efficiency of 76%
PS No. is the Power Supply number from the wiring diagram on page 66.
Table 4-1. Maximum DC power-supply load summary and equivalent 115 VAC current.

4.2 Load Analysis

4.2.1 Wing

As is shown in **Table 4-1** there are six DC power supplies for the MTP mounted in a DMT canister using 115 VAC 60 Hz power. They are used as follows, where the leading PSn is the Power Supply designation in the wiring drawing on page 66:

PS5, the large +24V supply powers the *Stepper Motor*,

PS1, the large +15V supply powers the *Frequency Synthesizer*,

PS2, the +15 V supply powers the IF amplifier and the frequency doubling amplifier, as well as +15 V analog circuits on the *Controller Board*,

PS4, the -15 V supply powers the IF amplifier and –15 V analog circuits on the *Controller Board*,

PS3, one +5 V supply powers logic circuits, and the other

PS6, +5 V supply powers circuitry on the *Temperature Controller Boards*. It operates in both “Standby” and “Operate” modes.

When the instrument is turned on from the cabin *Power and Status Panel*, the total 115 VAC draw from the DC power supplies will be 1.67 A (max). Measurement of the actual AC power draw from the power supplies indicates that the draw is only 0.34 A (max) and 0.21 A (typical).

In addition, there will be a 2 A (max) draw at 115 VAC 60 Hz to power four temperature-controlled heaters, which will be on whenever wing power is on. Therefore, the total 115 VAC 60 Hz load will be less than 3.7 A. Actual measurements of the total 115 VAC power indicate that it is 2.1 A (max) in Standby mode and 2.4 A (max) in Operate mode. The typical values are 0.9 A and 1.3 A, respectively, when measured on the ground. These can be expected to approach the maximum values when the instrument cold-soaks in flight and the heaters have a longer duty cycle. The maximum measured AC current is

completely dominated by the heaters, which draw 2.0 A when on, since the DC power supplies use a maximum AC current of 0.34 A.

4.2.2 Cabin

The *Power and Status Panel*, which provides all the 115 VAC 60 Hz to run *MTP Cabin Computer* and itself draws 1.6 A (max) and 1.2 A (typ). The *Keyboard and Display Drawer*, which is plugged in separately because it has a shared 8-port KVM switch and could be elsewhere in the cabin, draws x.x A (max) and y.y A (typ).

4.3 Wiring Used for MTP-H

All wires are from the RAF approved wiring list dated November 15, 2007. Per the RAF recommendation, MIL-W-22759 wire is used for all single conductor wire since it is approved by the FAA in Advisory Circular (AC) 43.13-1B Table 11-11 and 11-12, from any manufacturer, and does not need to be burn tested. The difference between Table 11-11 and 11-12 is that the former applies to "open wiring" and the latter to "protected wiring". Open Wiring is defined as "Interconnecting wire [that] is used in point to point open harnesses, normally in the interior or pressurized fuselage, with each wire providing enough insulation to resist damage from handling and service exposure. Electrical wiring is often installed in aircraft without special enclosing means." "Airborne wire that is used within equipment boxes, or has additional protection, such as an exterior jacket, conduit, tray, or other covering is known as protected wire." The specifications for protected and and open wiring are Mil-W-22759/11 and /16, respectively. They have the following properties:

Mil-W-22759/11

Insulation:	Polytetrafluoroethylene (PTFE)
Conductor:	Silver Plated Copper
Voltage Rating:	600 Volts
Temperature:	- 55^{o}C to + 200^{o}C

Mil-W-22759/16

Insulation:	Ethylene-Tetrafluoroethylene (ETFE)
Conductor:	Tin Plated Copper
Voltage Rating:	600 Volts
Temperature:	150^{o}C

For the MTP we have followed Mil-W-22759/11 since all the wires are protected.

4.3.1 115 VAC Wiring On Power Supply and Synthesizer Plate

Line	M22579/11 20 AWG Black
Neutral	M22579/11 20 AWG White
Ground	M22579/11 20 AWG Green

4.3.2 Temperature Control bus: 120 VAC and 12 VDC

Wire:	M22579/11 20 AWG
Thermal Cut-Out:	Cantherm R20 60 C Open-On-Rise
Heaters:	4-Ohmite TCH-35, 220 Ohm in series/parallel on each assembly. Each heater circuit is fused at 2 A (AGC-2).

4.3.3 Main Wiring Harness

Connectors: Conec Industrial D-sub IDC for Flat Ribbon

Wire: Temp-Flex F2807S-050-55

4.3.4 DC and Signal Wiring in Sub-Assemblies

Unshielded: M22579/11 24 AWG

5 Mechanical Description

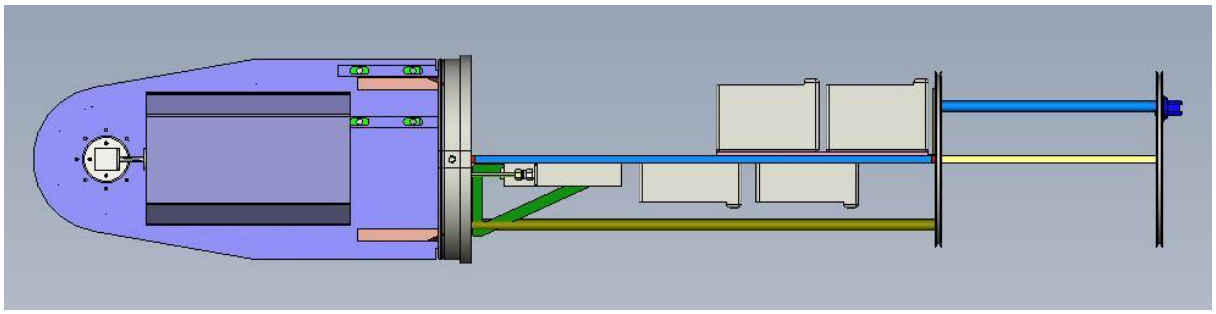
5.1 Wing

Four views of the MTP with canister, fairing and window removed are shown on the following page. NCAR/EOL/RAF (Mark Lord) performed the necessary mechanical load analysis for certification.

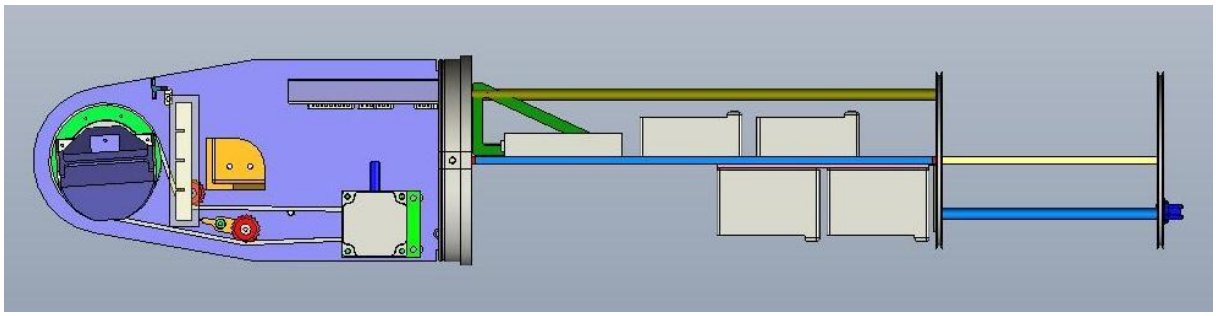
5.2 Cabin

Section 3.2 described the *Rack-Mounted Computer*, *Keyboard and Display Drawer*, and *Power and Status Panel* that are mounted in 3U (5.25 inches) of rack space in the cabin.

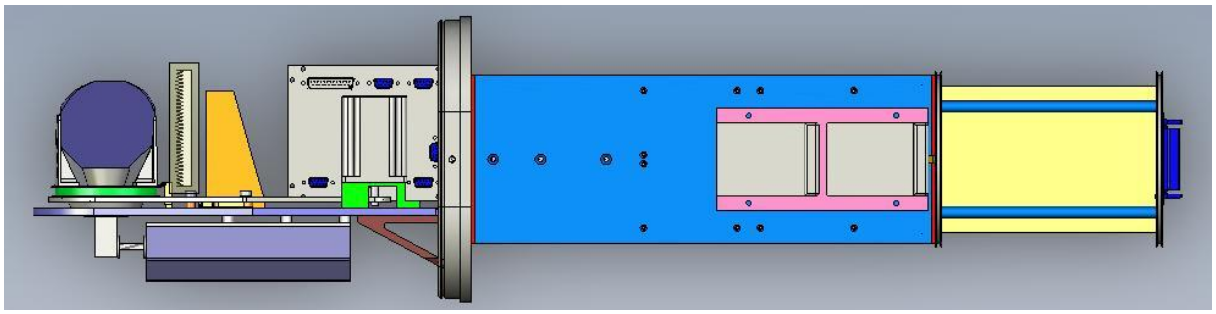
Fairing Removed



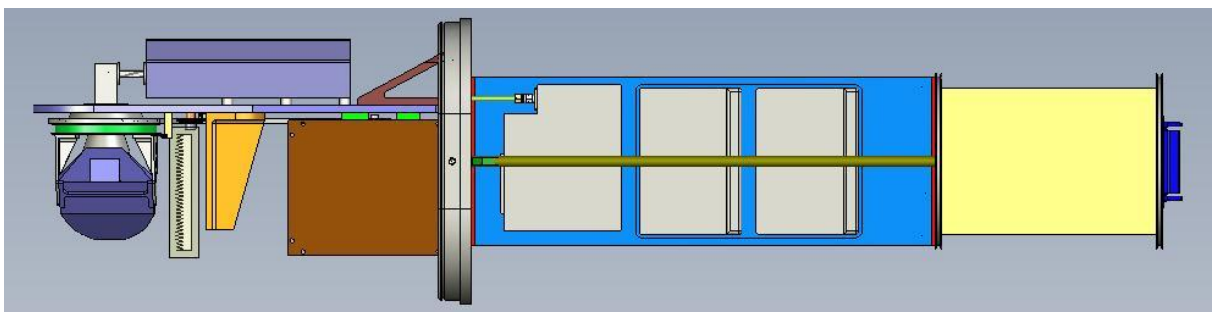
Front



Back (Front Rotated 180)

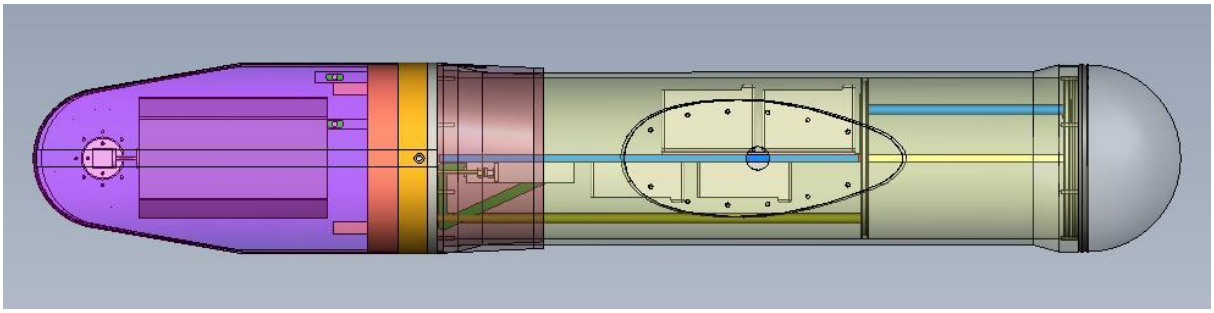


Top

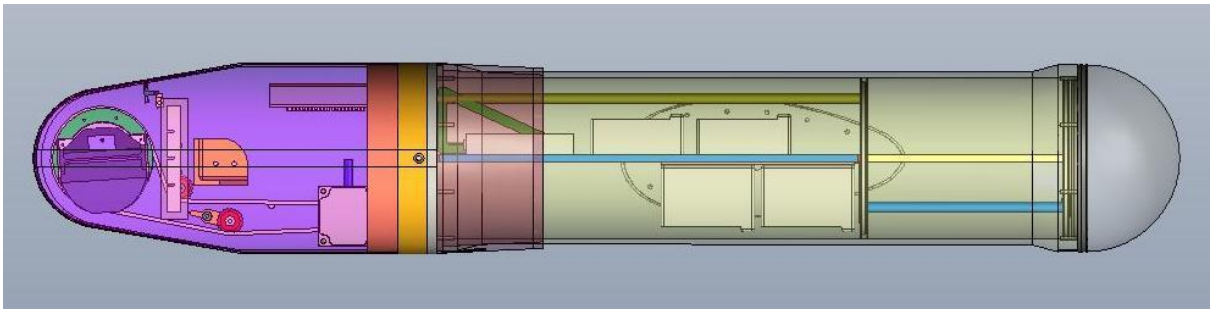


Bottom (Top Rotated 180)

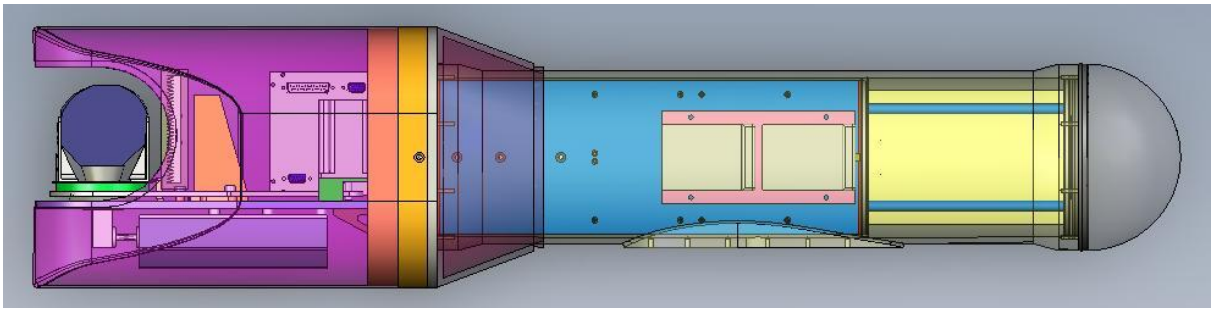
Transparent



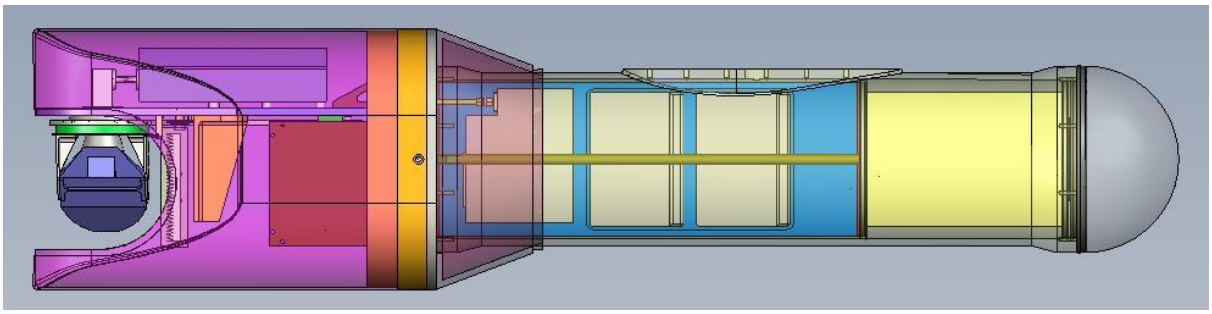
Front



Back (Front Rotated 180)

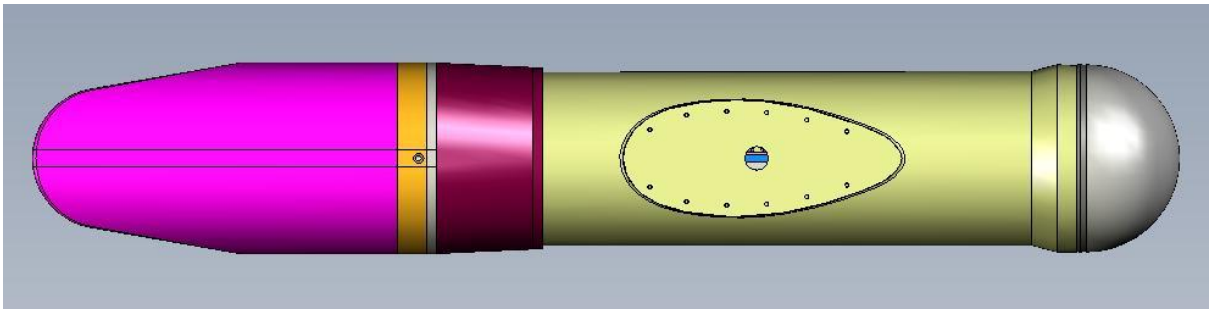


Top

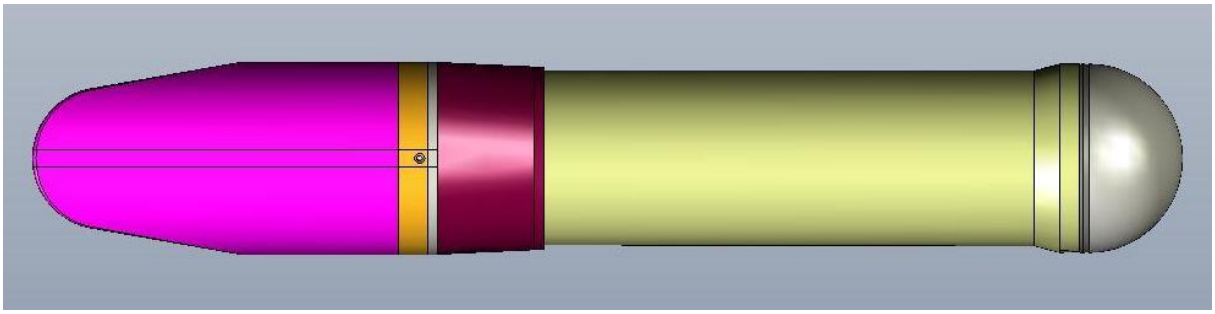


Bottom (Top Rotated 180)

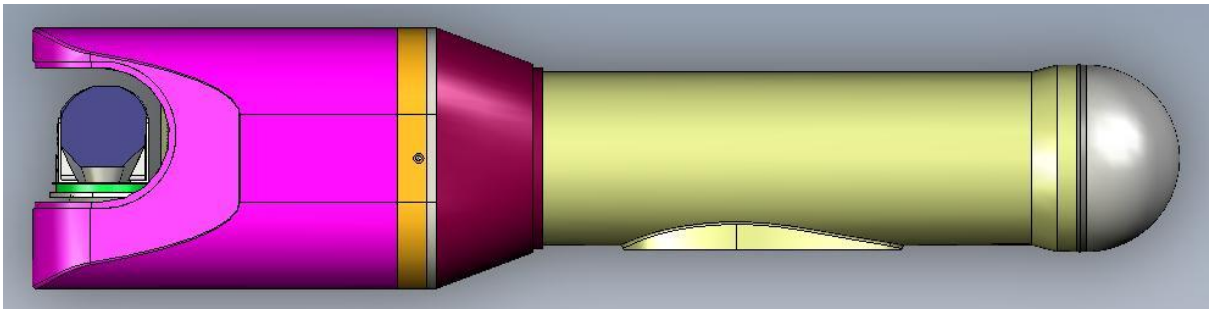
Solid



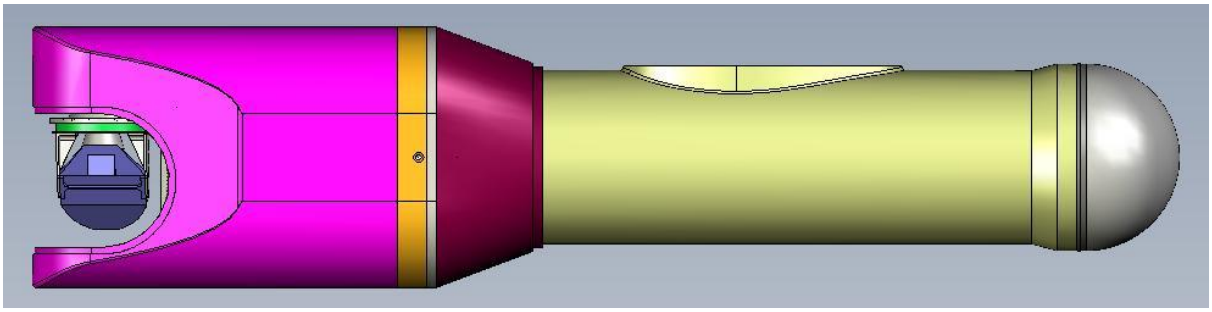
Front



Back (Front Rotated 180)

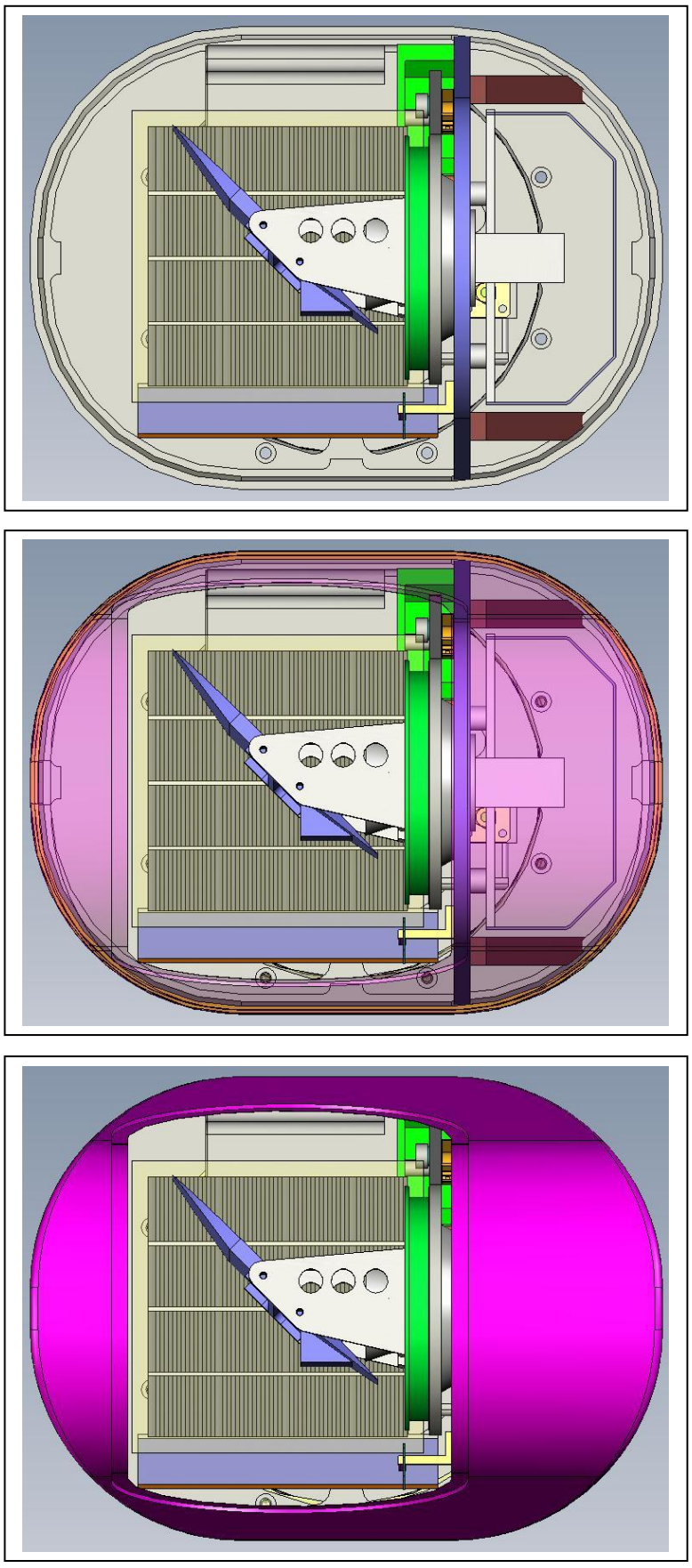


Top

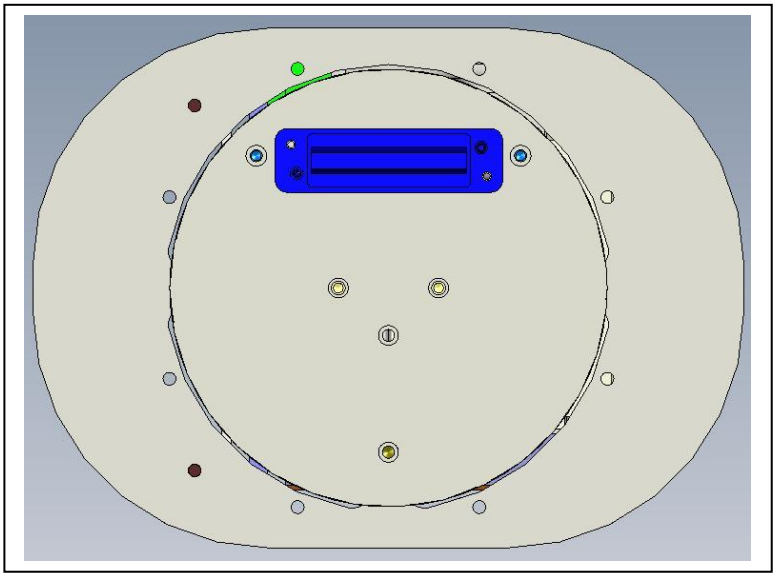


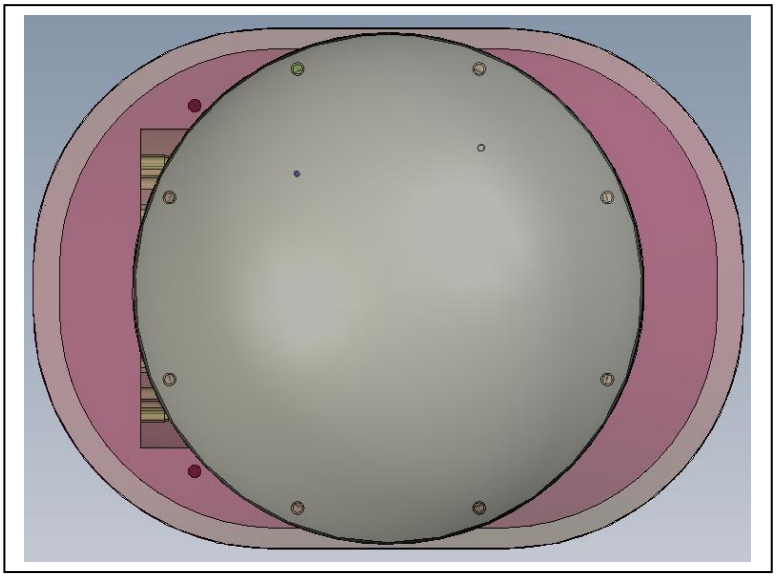
Bottom (Top Rotated 180)

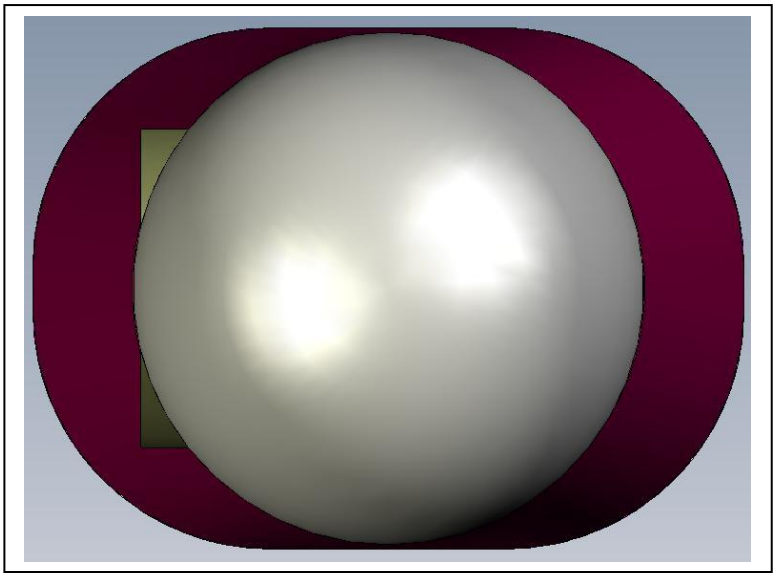
Nose



Tail







6 Software Description

The MTP software can be categorized into three broad categories:

- software that is used in the programmable integrated circuits (PICs) that are used in the instrument (the Controller Board has two and the Temperature Controller Boards have one),
- software that runs in the MTP Cabin Computer to control the MTP and collect data from it, and
- software that is used to perform retrievals and other data analysis-related steps.

These categories are described below.

6.1 PIC Software

There are three Microchip PICs used in the MTP-H. The Integration Timer and Counter PIC on the *Controller Board* uses native Microchip assembly language for programming. A listing of the assembly language code is provided in **Section 18**, and will also be provided electronically. The Control and IO PIC, also on the Controller Board, and the Temperature Controller Board PIC are both programmed using Hi-Tech C in MPLAB IDE, which compiles into native Microchip assembly code and is burned onto the Flash memory in the PIC. A C-language listing for the Control and IO PIC is given in **Section 19**, and for the Temperature Controller Board PIC in **Section 17**.

6.2 MCC Control Software

Visual Basic 6.0 software on the *MTP Cabin Computer* (MCC) is used to control the MTP in the canister and record data from it. This software will be provided electronically.

6.3 Data Analysis Software

Visual Basic 6.0 software is used for all the MTP data analysis software. It falls into five broad categories: the main retrieval program, a simulation program, several RAOB management programs, a program to calculate retrieval coefficients, and many utility programs. The main retrieval program is also used in the MCC to generate real time temperature profiles. All of the MTP data analysis software has already been provided to NCAR, and it is updated as revisions and upgrades are developed.

7 References

Brown, Shannon, Giovanni DeAmici, Alan Tanner, William Wilson, *CMIS Calibration Target Radome Measurements at 50 and 183 GHz*, JPL Internal Report, 2006.

De Amici, Giovanni, Ryan Layton, Shannon Brown, and David Kunkee, *Stabilization of the brightness temperature of a calibration warm load for space-borne microwave radiometers*, accepted by Transactions on Geoscience and Remote Sensing (TGARS), 2006.

Denning, R. F., S. L. Guidero, G. S. Parks and B. L. Gary, *Instrument Description of the Airborne Microwave Temperature Profiler*, J. Geophys. Res., 94, 16,757-16,765, 1989.

Gary, B. L., An Airborne Remote Sensor for the Avoidance of Clear Air Turbulence, AIAA Paper, AIAA-81-0297, 1981.

Gary, B. L, Observational Results Using the Microwave Temperature Profiler During the Airborne Antarctic Ozone Experiment, J. Geophys. Res. **94**, 11,223-11,231, 1989.

Lamb, James W., Miscellaneous data on materials on millimetre and submillimetre optics, International Journal of Infrared and Millmeter Waves, Vol. 17, Num. 12, pp. 1997-2034, December 1996. http://www.ovro.caltech.edu/~lamb/ALMA/Receivers/mmMaterialProperties2.pdf

Strand, O. N., and E. R. Westwater, Statistical estimation of the numerical solution of a Fredholm integral equation of the first kind, Journal of the ACM, 15(1):100-114, January 1968.

8 Appendices

9 Theory of MTP Measurements

In the main text we mention that the MTP measures the natural thermal emission from oxygen molecules in the Earth's atmosphere. A little more insight might help as it is not possible to measure a temperature profile using other molecules in the Earth's atmosphere. Molecular oxygen has some (almost unique) properties. First, molecular oxygen is a well-mixed gas in the atmosphere. Therefore, the amount of emission or absorption does not depend on geometric altitude, only frequency, pressure, temperature and to a small extent water vapor density. This could not be said, for example, for water vapor and other trace gases. Second, in order to measure a temperature profile, you want to observe at several frequencies with different absorption so that you can "see" different distances, and thus improve the information content of the measurements. (Measurements at several frequencies with the same absorption coefficient would only reduce the noise of the measurements but not improve the information content needed to make retrievals at larger distance from the airplane.) If molecular oxygen (O_2) had a simple diatomic-molecule rotational spectrum, like carbon monoxide (CO) for example, this would not be possible. For most diatomic molecules, the energy levels depend only on the rotational angular momentum quantum number (J) and the moment of inertia (I) of the molecule. The energy difference between energy levels increases as 2*(J+1)*B, where B is the rotational constant for the molecule. (B is inversely proportional to the moment of inertia of the molecule.) Thus the J=1-0 transition has 2B units of energy; the J=2-1 transition, 4B units of energy; the J=3-2 transition, 6B units of energy and so on. So if the J=1-0 transition was at 60 GHz (near the center of the oxygen band), the J=2-1 transition would be at 120 GHz, the J=3-2 transition at 180 GHz, etc. If O_2 was like CO, this is how the rotational lines would progress, and several receivers would be needed to measure several lines. Fortunately, Molecular Orbital theory (correctly) predicts that molecular oxygen has two unpaired electrons, which makes it paramagnetic. This produces a much more complicated set of spectral lines, and therefore several lines with different absorption can be observed using a single receiver.

Not just the MTP, but also many satellite-borne microwave temperature sounders, take advantage of these properties of molecular oxygen, but there are important differences. Satellite sounders generally observe in the nadir at a number of frequencies that define a number of averaging kernals over which a temperature is retrieved. In addition to observing a number of frequencies, the MTP also has a scanning mirror, which moves from near-zenith to near-nadir in the flight direction. This allows the MTP to obtain very good vertical resolution near flight level (see below).

As shown in the top left of **Figure 9-1**, when the MTP scan mirror is pointed at a particular angle Θ above the horizon, it is basically integrating the emission from oxygen molecules along the line of sight. The emission is weighted quasi-exponentially W(r) with range (r) because the emission further and further from the measurement location is absorbed by the intervening oxygen molecules. Now it is possible to show that in the case where the temperature changes linearly with altitude that the brightness temperature measured by the MTP is actually equal to the physical temperature at the e-folding distance Ra along the line of sight. (This assumes that the energy levels of the oxygen molecules are populated in thermodynamic equilibrium at the kinetic/physical temperature of the molecules, which they are.) The height above the aircraft

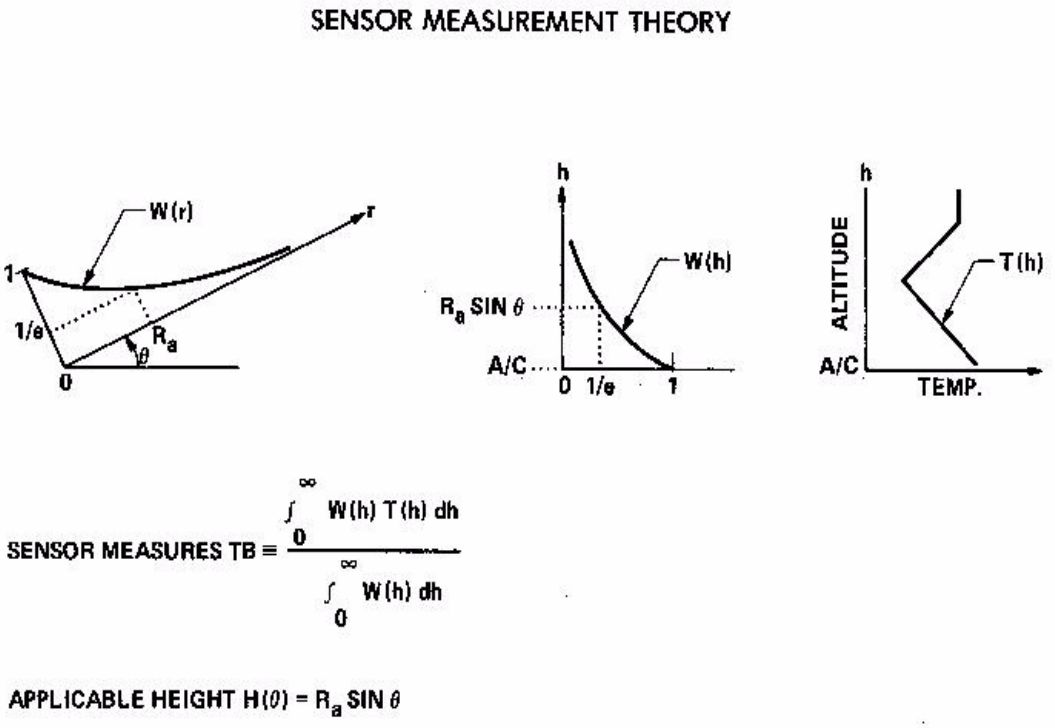


Figure 9-1. MTP measurement illustration.

corresponding to the e-folding distance is called the applicable height H(Θ) = Ra sin(Θ). Under the assumption of a constant lapse rate with altitude, measurements at a number of elevation angles could be used in this manner to construct a temperature profile. Note that when Θ is small the applicable height is also small. This is why the MTP can measure with very good vertical resolution near flight level. It should be noted however that the measurement angle is not the limiting factor here, because the MTP has a 7.5° FWHM beam-width. That's what determines the ultimate resolution and the Nyquist-sampling strategy. The reader should be aware that the vertical resolution of the MTP is elevation-angle, and therefore altitude, dependent. The horizontal resolution is also scan-angle/altitude dependent, and this is complicated by the fact that the aircraft is moving.

Of course the real atmosphere does not always have a linear lapse rate. Think of what would happen near the tropopause, for example, where there is a temperature inversion. As a result more powerful retrieval techniques need to be used. As mentioned in the main text, we use a statistical retrieval procedure with a Bayesian component. Many other retrieval techniques are possible such as physical retrievals or neural network retrievals. We have tried some of these and basically find that <5% improvements in accuracy are possible over a simple statistical retrieval (without Bayesian help). Since we have a Bayesian component to our retrievals, our approach is likely as good as what is possible.

10 Receiver Architecture Considerations

Figure 10-1 shows the microwave absorption spectrum for molecular oxygen from the ground (green) to 20 km (purple). For HIAPER we will be predominantly interested in the absorption from 10 km (blue) to 15 km (red). Notice that on the ground (green) that the pressure broadending of molecular oxygen is so strong the individual rotational lines are not apparent. However, even at 10 km the lines are starting to distinguish themselves.

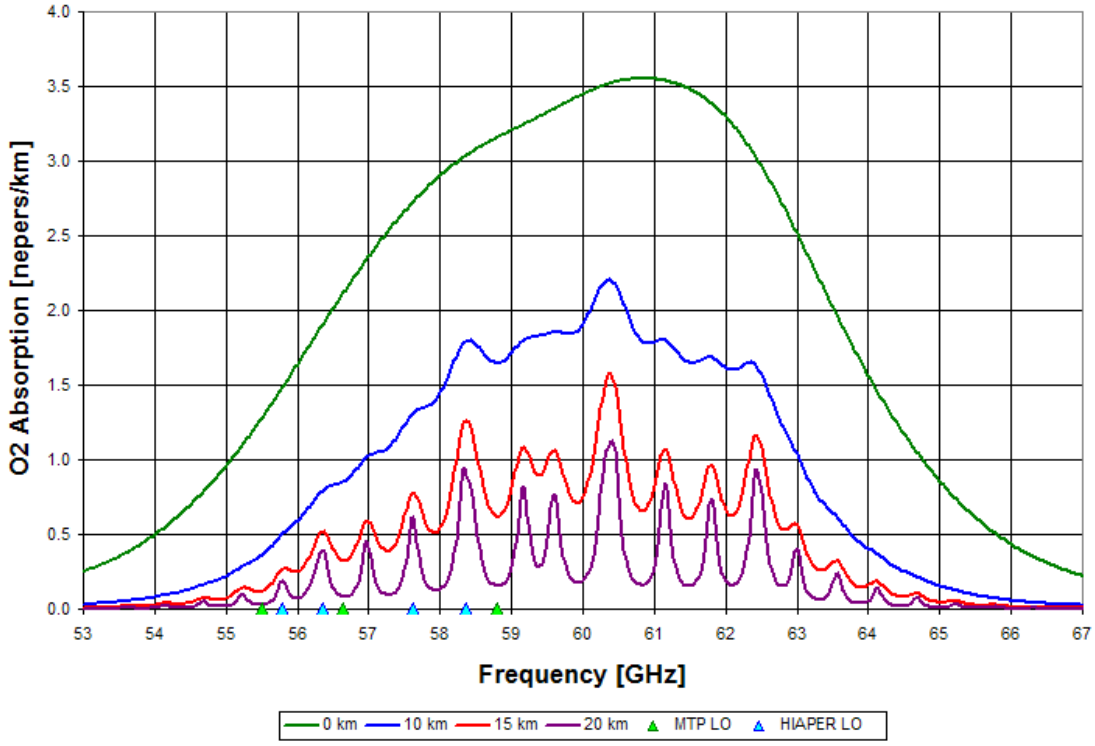


Figure 10-1. The molecular-oxygen absorption spectrum at altitudes from 0 to 20 km.

This is an important consideration for airborne MTPs.

The MTP instruments employ double side band (DSB) biased mixers. That is, they mix the received broadband radio frequency (RF) signal with a nearly monochromatic local oscillator (LO) signal to form an intermediate frequency (IF) signal from both the upper (USB) and lower (LSB) side bands (frequencies RF+/-LO). The green triangles at the bottom of the figure show the local oscillator (LO) frequencies for the three existing MTP instruments. Notice that the LO is located between pairs of lines so that the DSB receiver can receive signal from both side bands. The advantage of having two sidebands is a root two improvement in signal to noise. However, since both side bands don't have exactly the same absorption coefficient, the received signals in the two sidebands are not coming from exactly the same distance. This means that if there is temperature structure in the direction being viewed, there will be temperature smearing in the observables, and hence

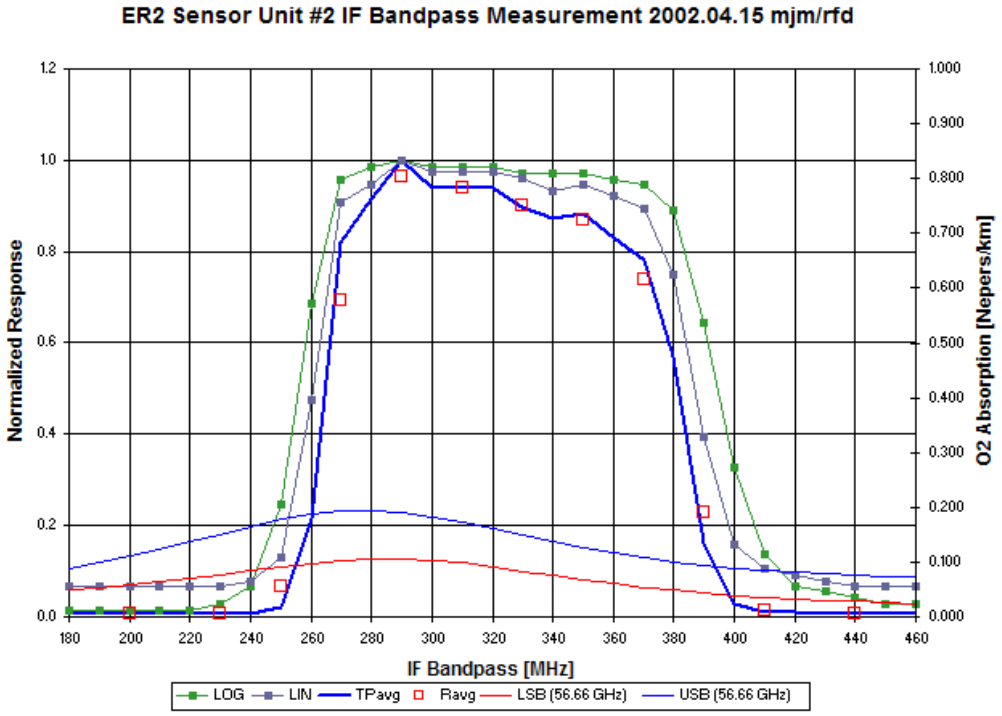


Figure 10-2. NASA ER-2 IF filter measurement and model points, with upper and lower sidebands superimposed.

de-graded retrievals. Another (not so obvious) issue is that the three pairs of lines don't have exactly the same frequency separation. Since the IF filter after the mixer is centered at a fixed frequency (320 MHz with 100 MHz bandwidth), this means that the IF filter will not be equally well matched to all three pairs of lines. This is illustrated in **Figure 10-2**. The thin blue line is the RF USB and the thin red line is the RF LSB superimposed on the IF filter shape (heavy blue line) for an LO frequency of 56.66 GHz. Not only is the absorption not the same for both sidebands, but they don't peak at the same IF frequency and they are not centered on the IF at 320 MHz. They peak closer to 280 MHz. For another pair of lines the match is better, but there is always some compromise because the line pairs are not equally spaced in frequency.

The issue of matching the IF filter to the upper and lower sidebands is obviously more important at high altitudes where line shape becomes more important. An way to avoid this problem all together is to use a single side band receiver and center the LO on lines of interest. This is an approach that we seriously considered for the HIAPER MTP. As shown in **Figure 10-3** a 3-7 GHz IF is used with individual filters for each of 4 observing frequencies. This has the advantages of observing all frequencies simultaneously and of being able to use a single fixed-frequency LO source (which would be cheaper and simpler than using a frequency synthesizer). On the down side, a high pass filter must be included in front of the mixer to avoid unwanted signals in the lower side band. Two bad things happen: you are only receiving half the signal because you are detecting only one sideband, and the high-pass filter adds additional insertion loss, which degrades the noise figure of the receiver.

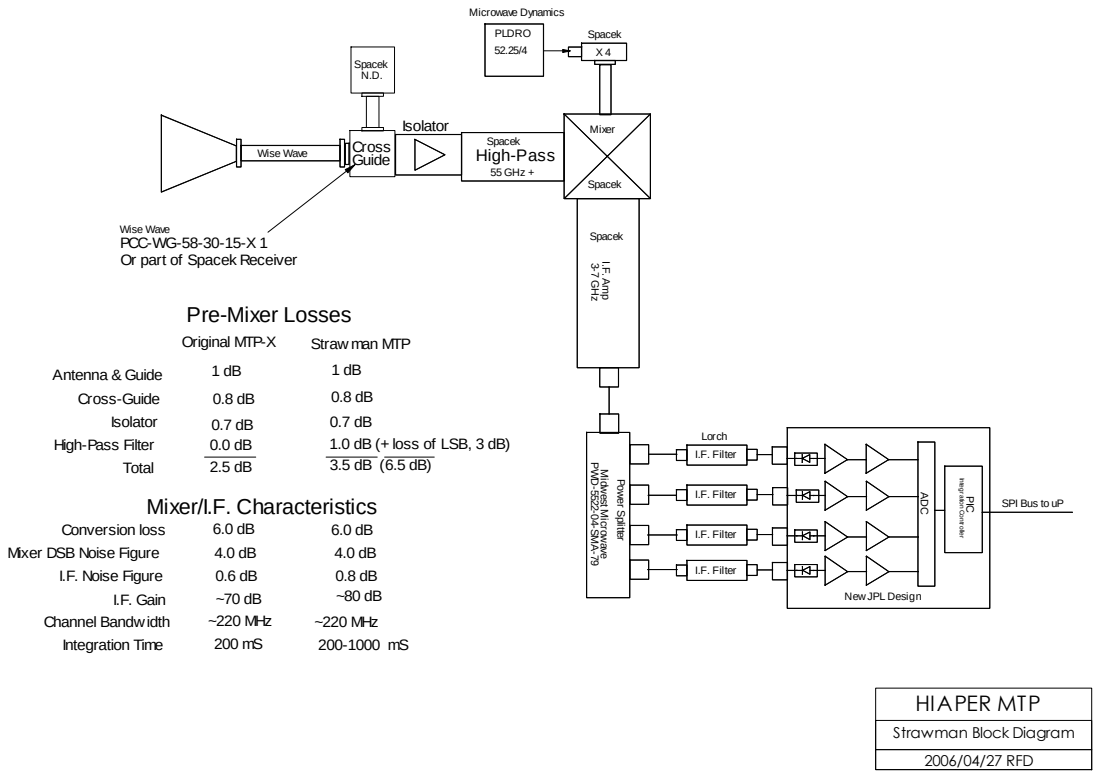


Figure 10-3. A single side band (SSB) receiver considered for HIAPER.

To overcome the degraded noise figure, we considered putting an RF amplifier in front of the mixer This has its only set of issues. Low noise InP RF amplifiers unfortunately do not have good 1/f noise performance. To get around this problem, a Dicke switch would have to be included in front of the amplifier. That has issues too. First you are only observing the signal half of the time, and second you are subtracting two signals; the net result is that a Dicke switched receiver is only half as sensitive as an unswitched receiver. After all of these considerations, we concluded that our current receiver configuration was the best choice (with one small change).

There is a trade off between mixer noise figure and IF frequency coverage. One way to improve the current upper and lower sideband mismatch issue is to put the IF at baseband with the LO centered on the lines of interest. This starts the IF as close to DC as possible with the LSB measuring the lower half of the line, and the USB measuring the upper half of the line. When we bought our current mixers, the noise figure of baseband mixers was not nearly as good as at 320 MHz (a frequency dicated by line pair separation). Today it is possible to get good mixer noise figure performance with a baseband IF starting at 10 MHz. That is what we are going to do. **Figure 10-4** shows an ideal rectangular IF filter superimposed on the 57.615 GHz oxygen line. The filter has a bandwidth of 100 MHz in this figure. The actual bandwidth is important because it is one of the parameters determining the receiver's sensitivity. Below 10 km **Figure 10-4** shows the absorption

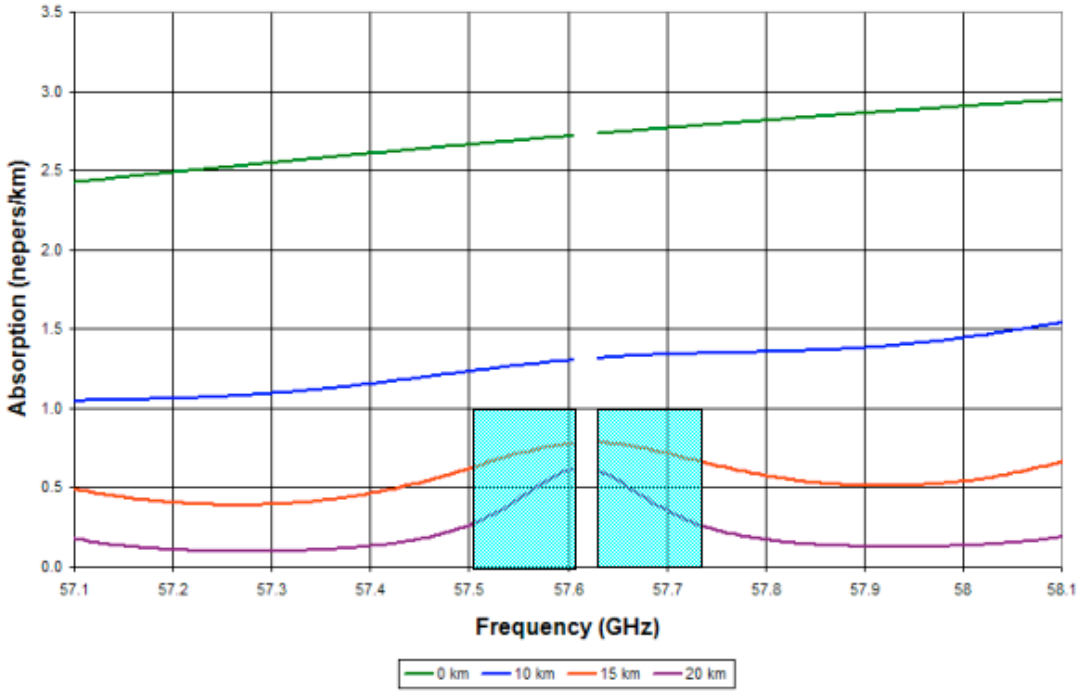


Figure 10-4. Upper and lower sideband IF with 100 MHz bandwidth superimposed on the 57.615 GHz oxygen line with a baseband IF starting at 10 MHz (20 MHz

slowly increasing with altitude so much larger bandwidths could be used. The rms variation in the absorption at 10 km, expressed as a percentage is 1% for a 100 MHz wide IF, and only 2% for a 200 MHz wide IF. This means that over the IF pass band there is not very much variation in absorption and there will not be much temperature smearing in the retrievals. However, by 15 km (close to HIAPERs ceiling) these percentages increase to 5% and 13% respectively, which is starting to become significant. Since the IF filter is easily replaced and not expensive (~$200), it might be important to exchange the filter depending on the planned flight profile.

11 Investigation of Target Cover Materials

In order to calculate the gain of a microwave receiver it is necessary to have at least two reference temperatures. These could be obtained by using hot and cold (or ambient) targets, or as in the case of all MTPs built to data, an ambient reference target and the outside air temperature (OAT). Reference targets are ussually made from pyramidal-shaped microwave-opaque materials that have a very large emissivity (as close to unity as possible); they are generally mounted on a metal substrate into which are embedded one or more precision platinum resistive thermal devices (RTDs) to measure the physical temperature of the target. For these targets to be useful (i.e., accurate) it is necessary that there are neither axial nor planar temperature gradients across the target. This is particulary challenging in the space environment where solar insolation can easily produce thermal gradients if the target is not thermally insulated. However, even in the more protected thermal environment of the MTP fairing, temperature gradients in the reference target can still be an issue. To provide a benign thermal environment, the MTP reference target has to date been surrounded by a one-quarter inch thick layer of Styrofoam – a Dow Chemical Company expanded polystyrene (EPS) foam. An important property of an insulating material (in addition to having very small thermal conductivity) is that it not absorb microwave energy as this would effectively reduce the emissivity of the target and lead to a disagreement between the physical temperature (measured by the RTD) and the brightness temperature (measured by the radiometer). (This relationship can be affected by many more issues than will be discussed here.) Using one-quarter inch thick Styrofoam results in a thermal time constant of about one-half hour. Until recently this was the best material available: it was essentially microwave transparent, or equivalently, it had a dielectric constant (or relative permittivity, ε_r - see below) whose imaginary part was very small.

Before proceeding it will be useful to write down a few equations relating the dielectric constant to other common physical parameters, as the relationship between them will be useful in identifying material other than Styrofoam that might be useful as microwave-transparent target-insulating materials. The reason for needing to look for an alternative material to Styrofoam is that Styrofoam is flammable, and it was suggested during the PDR that, even though the MTP target uses a very small amount of Styrofoam (4” x 4” x 0.25”) an effort should be made to find a less flammable material.

So we move on to a discussion of the relationship between common terms used to describe the microwave properties of material such as the refractive index, the dielectric constant, and the permittivity. First off, we note that the complex refractive index, m:

$$m = \sqrt{\varepsilon_r \cdot \mu_r}$$

is the square root of the product of the relative permittivity (ε_r) and the relative permeability (μ_r), where the term *relative* means relative to free space. The relative permittivity is just another name for the dielectric constant, and is given by:

$$\varepsilon_r = \frac{\varepsilon}{\varepsilon_o} = \varepsilon_{re} + i \cdot \varepsilon_{im}$$

where ε is the permittivity, ε_o is the permittivity of free space, and ε_{re} and ε_{im} are the real and imaginary parts of the relative permittivity or dielectric constant. A similar expression can be written for the relative permeability. However, if a material is non-

$$m = \sqrt{\varepsilon_r}$$

magnetic, the relative permeability is unity, and the complex refractive index is simply the square root of the relative permittivity:

Now the microwave energy loss, or absorption, in any material is simply proportional to the imaginary part ε_{im} of the dielectric constant (as well as the electric field strength squared and the frequency). Therefore, to evaluate different insulators for the reference/ calibration target, we need to know the real and imaginary parts of the dielectric constant of any potential material.

Sometimes microwave engineers use the loss tangent, tan(δ), instead of ε_{im} to represent the absorption. In terms of the loss tangent, ε_{im} can be written: $\varepsilon_{im} = \varepsilon_{re} \tan(\delta)$.

Finally, in terms of the complex refractive index, m = n + i k, where n and k are the real and imaginary parts, it is easy to show that $\varepsilon_{re} = n^2 - k^2$ and $\varepsilon_{im} = 2nk$. Armed with this knowledge we can scour the literature looking for complex refractive index, complex permittivity, complex dielectric constant, and loss tangent information to determine which materials will work best for the reference target window.

A comprehensive list of the microwave and submillimeter wave properties of materials was published in 1996 (Lamb). In the decade since then, many new (and better) microwave materials have been developed. In particular, colleagues at JPL have been very involved in testing these materials for both spaceborne (De Amici et al. 2006) and airborne (Brown et al., 2005) platforms. These recent tests show that one material stands out amongst all others for use in insulating microwave calibration targets. This material is Plastazote LD15, a low-density polyethylene (LDPE) foam made by a British company, Zotefoams PLC. The JPL tests showed that Plastazote LD15 had a value for ε_{im} of 4.5 $10^{-6} \pm 5.0\ 10^{-6}$ at 50 GHz (and only 50% worse at 183 GHz). As is clear in Figure C-1,

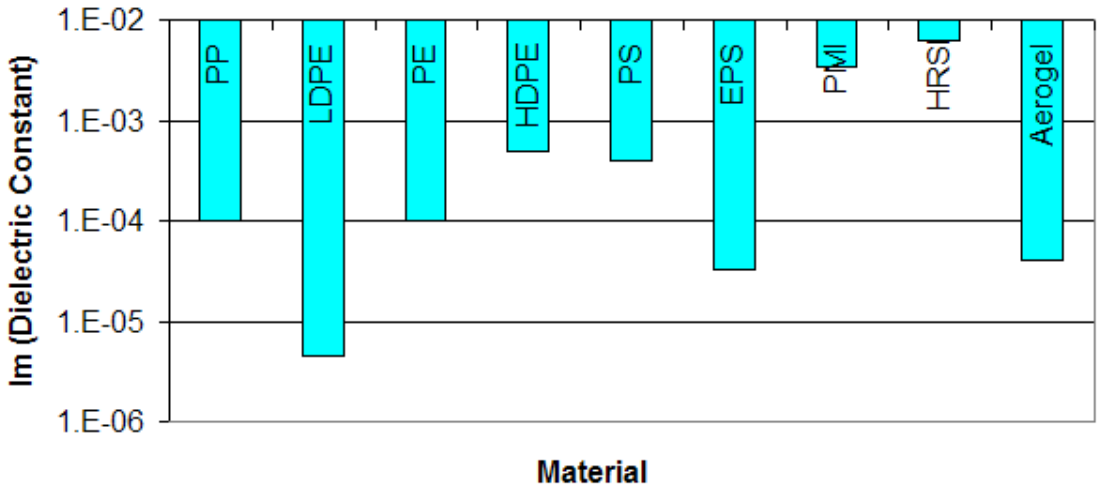


Figure 11-1. Materials considered for the HIAPER MTP reference target insulation.

Material abbreviations

PP (polypropolene) LDPE (low density polyethylene) PMI (polymethacrylimide)
PS (polystyrene) HDPE (high-density polyethylene) EPS (expanded polystyrene)
PE (polyethylene) HRSI (high-temperature surface insulation)

this makes Platazote LD15 nearly an order of magnitude less absorbing than any other known material. In fact it is so transparent that the value of the imaginary part of the relative permittivity (dielectric constant) is smaller than the accuracy of the measurement. It should be little surprising that virtually every group that uses microwave calibration targets uses this material.

We contacted Zotefoam shortly after the PDR to discuss other fire retardant foams that they make so that we could test them. We were promised four additional samples to test, but repeated phone calls have not been returned. Obviously they realize that we only need a small quantity and are not interested in pursuing the matter.

One material suggested at the PDR for use on the MTP target (because it has acceptable flammability characteristics and is used by RAF) was ROHACELL -- a polymethacryimide (PMI) foam made by Rohm GmbH. As can be seen in **Figure 11-1**, the imaginary part of its dielectric constant is the second worst of the nine materials considered.

Before we were aware of ROHACELL's properties we ran tests in our laboratory to compare it to Styrofoam. We set up an MTP to look at a 90 K hot target and adjusted the gain so that the analysis software indicated 90 K. We then inserted the ROHACELL foam sample and noted that the temperature dropped to 72 K! We then inserted a similar thickness of Styrofoam and saw that there was a <0.2 K temperature change.

ROHACELL has terrible microwave electrical properties. In hind site this could have been predicted. Although its dielectric constant is excellent (1.03, low-density foams are

mostly air), it has a large loss tangent (0.003) – nearly three orders of magnitude poorer than Plastazote LD15! The properties are from the Rohm web site: http://www.rohacell.com/en/performanceplastics.html?content=/en/performanceplastics/rohacell.

Space Shuttle tiles (HRSI) have been suggested in the past as an insulator, and as can be seen in **Figure 11-1**, it is the poorest of the materials considered. JPL makes aerogels for various space probes and we considered it as an option. It's imaginary part of the dielectric constant is slightly worse than Styrofoam, and it is very expensive, absorbs water vapor, and requires special mounting. Being silicon based however, it does not burn and is an extremely good insulator.

To summarize, it is clear that Plastazote LD15 is by far the best material to cover the reference target. We in fact plan to recover all of our existing MTP targets with this material instead of Styrofoam. However, to address the flammability issue, we will use Plastazote LD15 only on the face of the target facing the scan mirror to minimize the amount of flammable material. (We could find no literature on the flammability of this Zotefoam, but we were told by Zotefoams PLC that it is comparable to Stryofoam.) All other surfaces surrounding the target will be covered with ROHACELL which is certified for aircraft use and also has a slightly lower thermal conductivity (0.031 W/mK) than Plastazote LD15 (0.039 W/mK). Both of these material are slightly more conductive than Styrofoam (0.025 W/mK).

12 Temperature Control Circuits – Small_TC

Four assemblies in the MTP-H canister are temperature-controlled, either for temperature stability during a measurement, or to reduce condensation after cold-soaking at altitude. The four controllers (see **Figure 12-1**) are implemented in PIC16F675 flash micro-controllers on small PCBs mounted on or near the temperature-controlled assembly. A schematic drawing of the temperature controller board is shown in **Figure 12-2**.

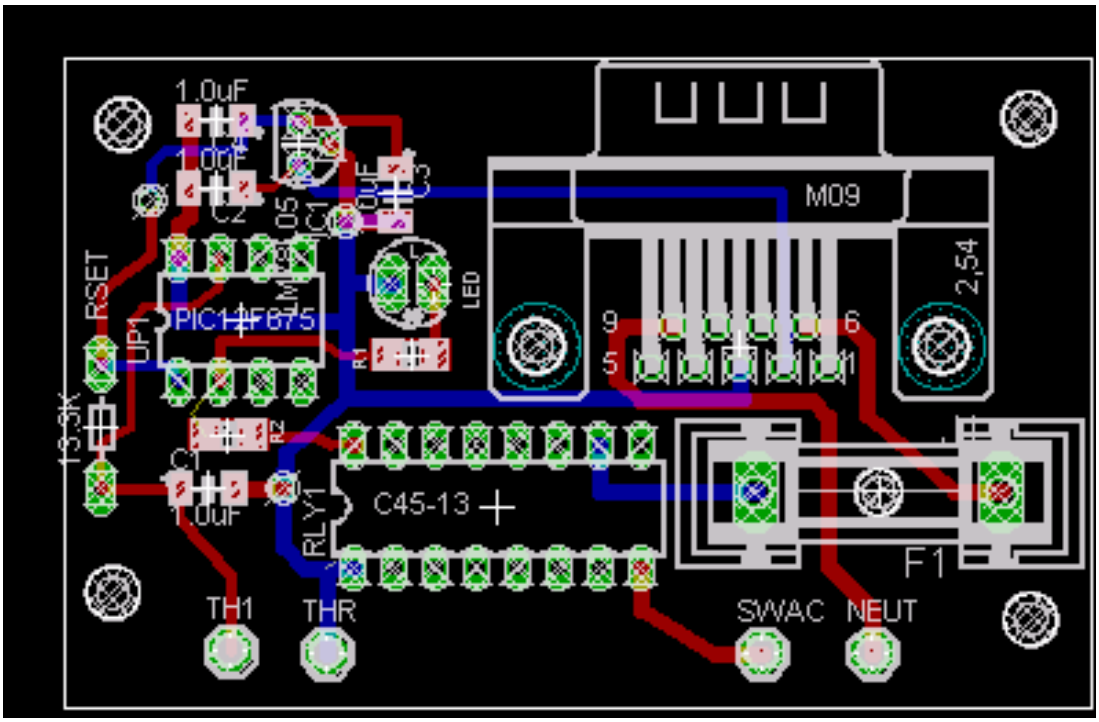


Figure 12-1. The MTP-H Temperature Controller board (show much oversized).

The control temperature is set by selecting a resistor that equals the resistance of the control thermistor (44008) at the desired set point. Typically, all the controllers will operate at about 40 C using a set point resistor of 16.2 K. The micro-controller drives a zero-crossing solid state relay -- a Teledyne C45-11, which switches 120 VAC to a string of four 220 ohm, 35W resistors in series-parallel configuration, producing a maximum heating capacity of about 65W for each heating circuit. The resistors are attached to the parts that they are heating by screws and thermally conducting compound.

The control program pulse-width modulates the relay at about a 2-Hz rate, and the zero-crossing relays switch the power near the zero-crossing of the AC waveform, to minimize switching noise. The temperature control system will operate with power line frequencies from 47 to 440 Hz.

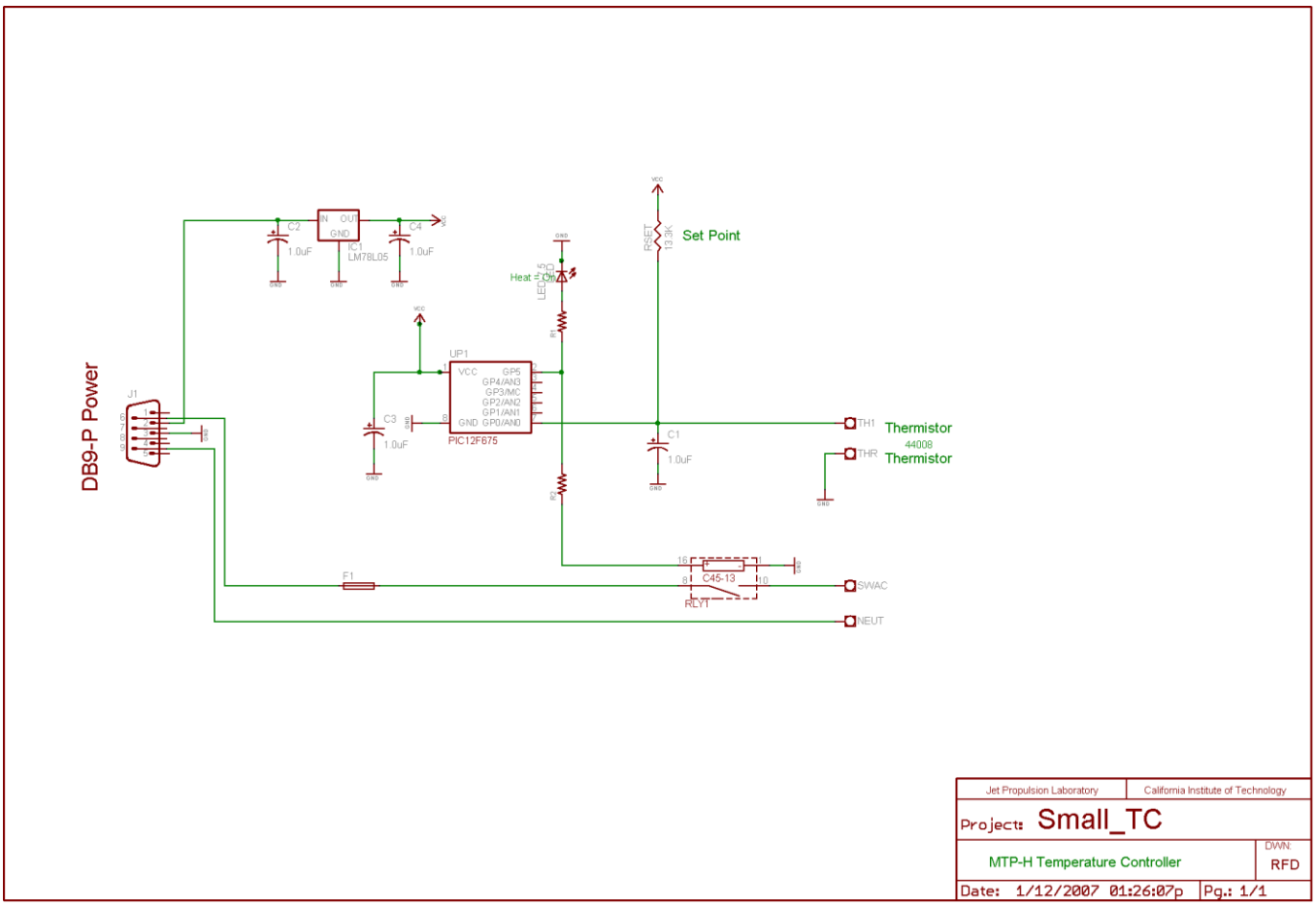


Figure 12-2. Schematic drawing of the temperature controller.

13 The MTP-H Controller Board Drawings

13.1 Sheet 1 - Overall architecture

The MTP for HIAPER is comprised of two major systems: the *MTP Cabin Computer* (MCC), and the MTP itself in a DMT-style canister. The MCC controls the MTP by sending commands to, and receiving data from, the MTP-H *Controller Board* in the canister over an RS-422 full-duplex serial bus.

The schematic drawings and printed circuit design were created using EAGLE 4.16. Sheet 1 of the *Controller Board* schematics is a block diagram showing how the various functions have been divided into logical subsystems on separate sheets. Each sheet has a manageable number of input and output lines, which makes it easy to see how the circuits work.

The MTP-H *Controller Board* consists of a combination of flash-programmed micro-controllers and data acquisition integrated circuits along with analog components. While the internal workings of the controller is quite complex, it can be treated as a 'black box' that interprets a small number of high-level commands and returns a correspondingly small set of data types.

For example, there is a command to read all the engineering ADC channels, which is an "M" followed by a carriage return. The response is a string of ASCII characters starting with "m" containing 16 channels of ADC data. Similarly, there are commands to read the status word, to start an integration cycle, move the scan system, and so forth.

Because of their "Flash-based" memory, the micro-controllers on the MTP-H Controller PCB are "in-circuit programmable," meaning that by connecting a programmer to a small header on the PCB, the program in any of the micro-controllers may be replaced without removing them.

13.2 Sheet 2 - PIC Control & IO

A pair of 75176 differential transceivers, U1 & U2, convert between the RS-422 received and transmitted signals to and from the *MTP Cabin Computer* and the TTL levels used in UC1, a Microchip PIC16F876 flash-based micro-controller. UC1 acts as the gateway to the various circuits in the controller that perform the control/data acquisition functions via the Serial Peripheral Bus (SPI).

While the SPI bus handles almost all of the data traffic on the *Controller Board*, certain time-critical events (such as the ITG_BSY line that indicates whether the Integration Counter is busy or done), are read directly by UC1 on an I/O port to avoid delay in transmission on the SPI bus. For similar reasons, other lines controlled directly by UC1 turn on/off the Noise Diode, or flash an LED, for instance.

The SPI bus is organized as three lines (SDO, SDI, and SCLK) that are common to all peripheral devices, and several "Chip Selects" (CS0-9). Only one CS line is active at a time to select to which circuit the communication is directed.

MTPH controller Chip Select (CS) assignments:
CS0: Integrator (UC2)
CS1: Engineering Mux (ADC1)
CS2: Engineering Mux (ADC2)
CS3: Platinum RTD (ADC3)
CS4: Platinum RTD MUX (SW1, SW2, SW3)
CS5: Accelerometer Processor (UC3)
CS6: Frequency Synthesizer (CTI SLS-1403, Off-board)
CS7: Stepper Motor (UART1, Lin Engineering Silverpak 23CE off-board))
CS8: Auxiliary UART (UART2)
CS9: Spare CS

The control program in the PIC16F876 is called MTPH_Control.C and is described in Section X.

13.3 Sheet 3 - VFC and Integration Counter

The basic data produced by the MTP-H is a DC voltage proportional to the brightness temperature in front of the antenna. This voltage is converted into a pulse train whose frequency is proportional to the input signal by VFC1, an Analog Devices AD654. IA2 and OP3A provide for gain and offset adjustments during the initial setup of the MTP.

This pulse train, in turn, is counted for a known time, typically 200 mS by UC2, a Microchip 16F88 micro-controller, programmed to do the communications, timing and counting functions that perform the integration. It also signals via the ITG_BSY line when an integration is in progress and then answers a request to send the resulting values back to UC1 for transmission to the *MTP Cabin Computer*.

The integration time is controlled by a command on the SPI bus to UC2, which sets values from 1 to 256 increments of 20ms each, or from 20ms to 5.12 s. The value chosen is determined by a trade-off between receiver noise figure, receiver 1/f noise, and the time it takes to complete one scan cycle.

The integration program in the PIC16F88 is called “F88_integ_IRQ.ASM” and is listed in Section X.

13.4 Sheet 4 - Engineering Mux and ADCs

Sixteen (16) channels of engineering analog-to-digital conversion are provided by ADC1 and ADC2, both MAX186 low-power serial 8-channel analog-to-cigital converters. Certain of these voltages require a scaling and/or polarity change to fit into the ADC range of 0 to +4.096V. In the case of measuring engineering temperatures by thermistor, a resistor to the reference voltage VTREF is used to provide bias. Typically, this resistor has a value near the thermistor resistance at the expected operating temperature of the device being monitored, so that mid-scale on the ADC represents the normal operating point. In practice, these resistors are set to a “typical” mid-scale value of 34.8 K, which corresponds to a temperature of +22 C. Thermistors are highly non-linear (actually exponential), so a linearization algorithm is used in the data analysis to produce the corresponding temperature value.

The MAX186’s generate an internal reference voltage, which is accessible externally. The reference from ADC2 is buffered by OP3B and used for the thermistor bias mentioned previously.

13.5 Sheet 5 - Platinum Multiplexer and ADCs

Radiometric measurements are generally made by measuring the difference in radiometer output between a known “Reference” and an unknown temperature such as, in the case of the MTP, the air temperature. This comparison cancels out most of the variability in the

receiver itself, and simplifies several independent variables down to a single parameter: the physical temperature of the reference, which can be directly measured. Conversely, this means that the accuracy of the radiometric observation literally hinges on the determination of this one parameter.

All physical temperatures in the MTPH, which are used in the radiometric calculations, are determined from platinum Resistance Temperature Devices (RTDs), which provide excellent long-term stability. One of the penalties of the platinum sensors, however, is a low resistance (500 ohms at 0 C for the ones used here), which requires a relatively high measurement current (about 1 mA) and causes vulnerability to resistance in the leads and connectors (1 ohm ~ 0.5C).

In the MTP-H, a more-or-less standard 4-wire technique is used to remove the sensitivity to conductor resistance, and an internal calibration method borrowed from space-flight designs provides very high confidence levels, as well as a way to monitor the changes in the circuits without having to do frequent calibrations.

The circuitry used to make these measurements is comprised of:

The current source, which generates a voltage across the selected RTD or calibration resistor, comprised of OP1A, OP1B, and OP2A.

The current multiplexer (SW3, a MAX349 8-channel switch), which sends the current source to one of the six RTDs, or two calibration resistors.

The differential voltage multiplexer (SW1 and SW2, both MAX349s), which sample the voltage across the selected resistor or RTD and send it to:

An instrumentation amplifier (IA1, an Analog Devices AD620), which converts the differential voltage from the resistor or RTD to a single-ended input for:

A 12-bit A-to-D Converter (a MAX1272), which makes the A/D conversion and sends it to the controller (UC1).

The overall gain of the measurement circuit can be set with the gain resistor RG, on IA1, and the scale offset is set by the resistors associated with OP2B.

By use of this 4-wire circuit, resistance of 100 ohms or more in any or all conductors to the RTD can be tolerated without causing measurable error.

13.6 Sheet 6 - Stepper and Synthesizer Communications

The local oscillator frequency synthesizer -- a CTI SLS-1403 -- is tuned by sending the desired channel number directly to it over the SPI bus at CS6.

The stepper motor -- a Lin Engineering SilverPak 23CE -- has an intelligent controller built in, and needs only high-level commands to be sent to it on an RS-485 half-duplex

(sending and receiving at separate times) bus. A MAX3100 UART and a 75176 transceiver convert the commands and responses to and from the SPI bus using CS7.

Another UART/Transceiver circuit provides a second RS-485 bus for future expansion, at SPI address CS8.

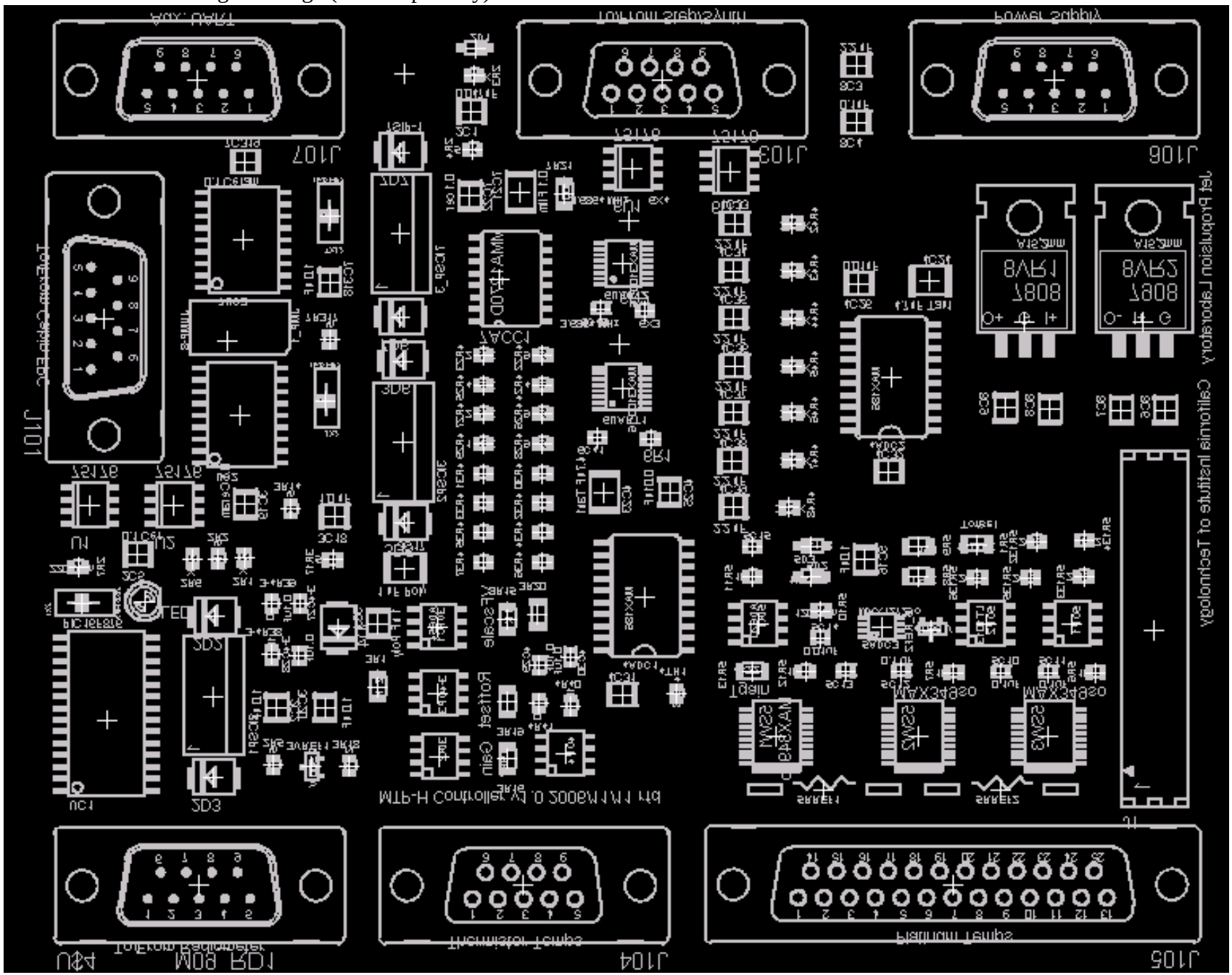
13.7 Sheet 7 - Accelerometer

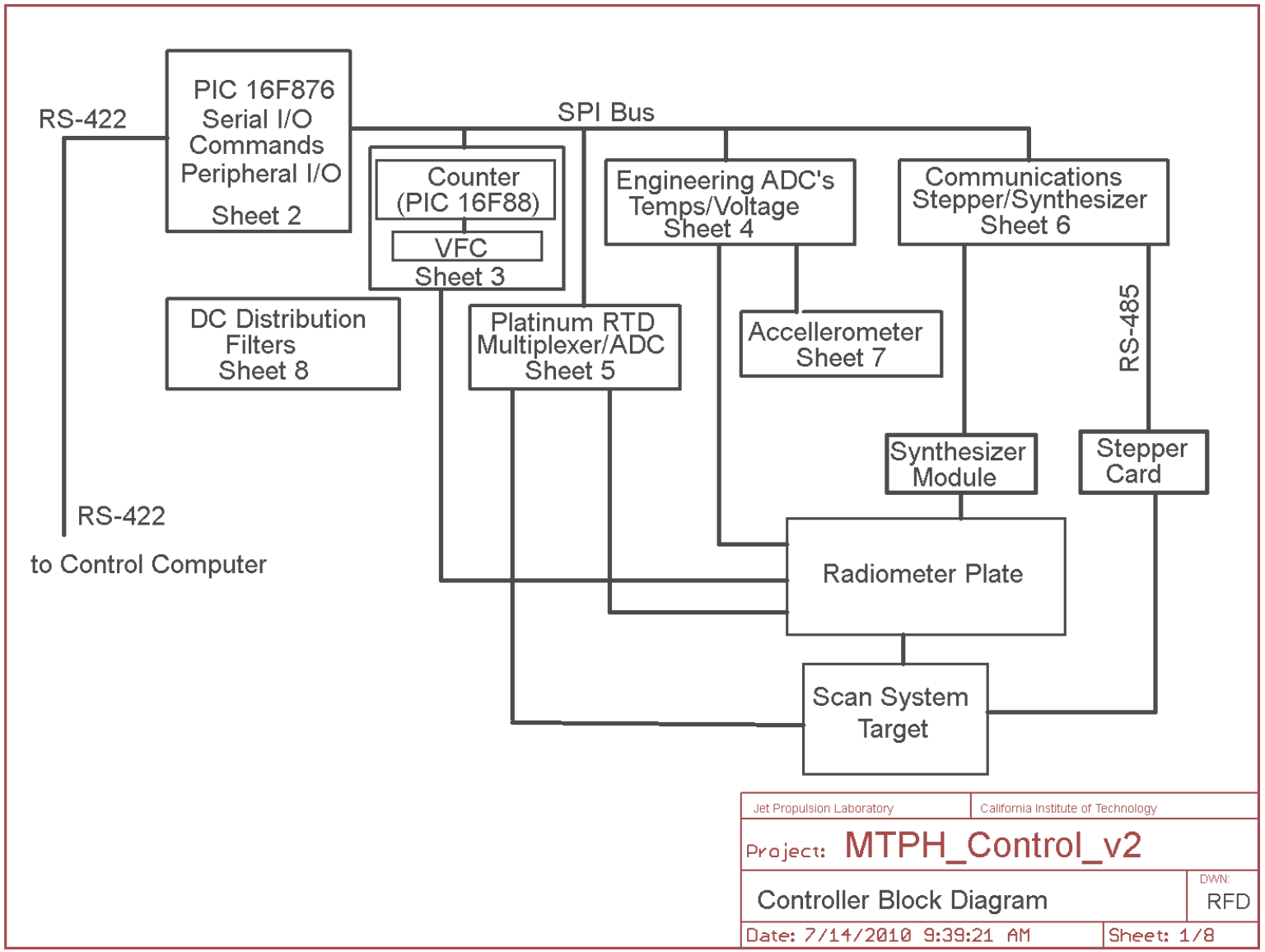
An accelerometer measuring aircraft vertical acceleration has traditionally been included in MTP designs as a means of studying the relation between the shape of the temperature profile and air turbulence. An MMA1270D MEMS accelerometer converts the acceleration to a voltage which is sent to the engineering mux.

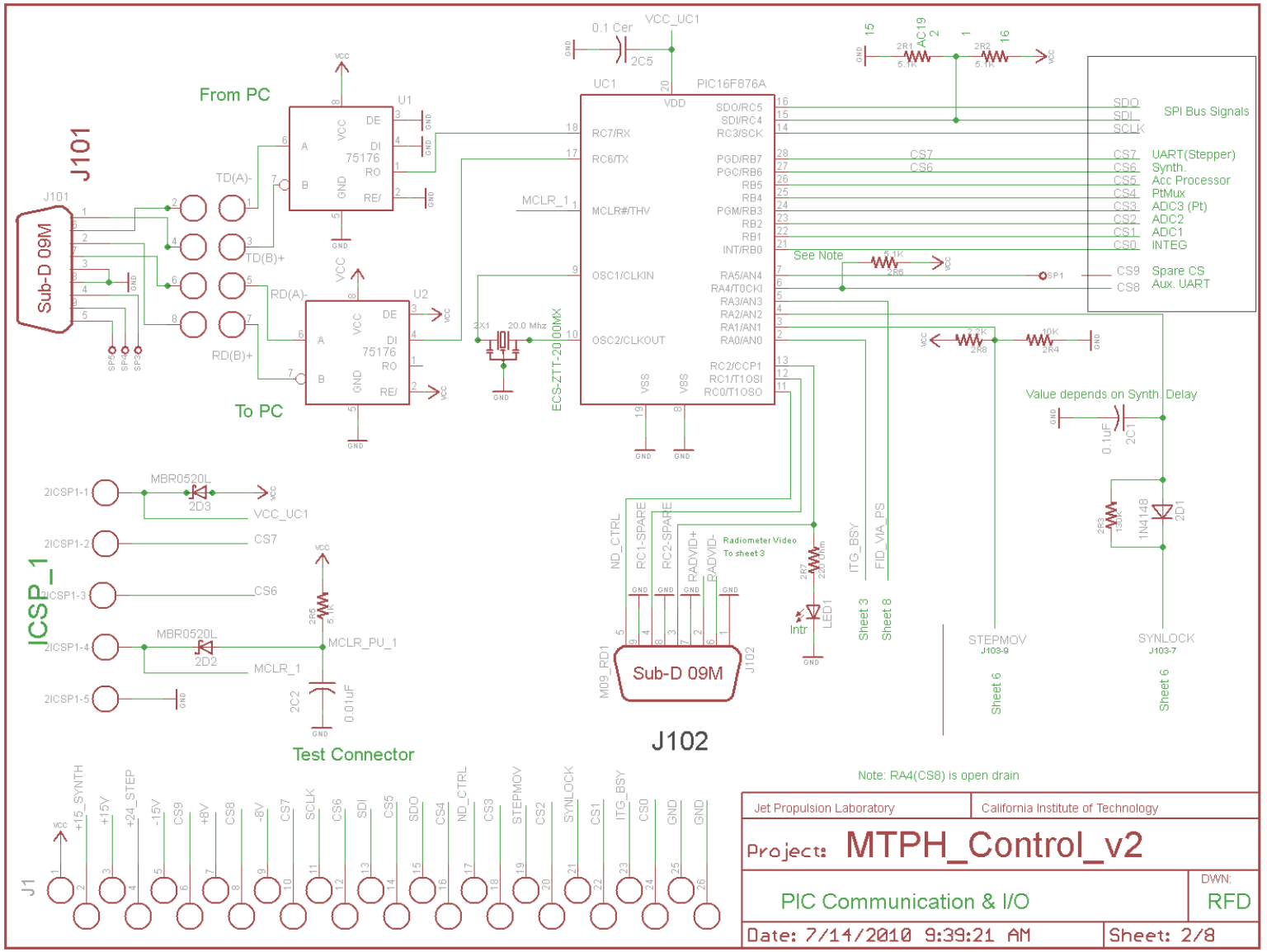
More refined acceleration monitoring may be added later by the use of UC3, a PIC16F88 microcontroller, addressed on the SPI bus as CS5.

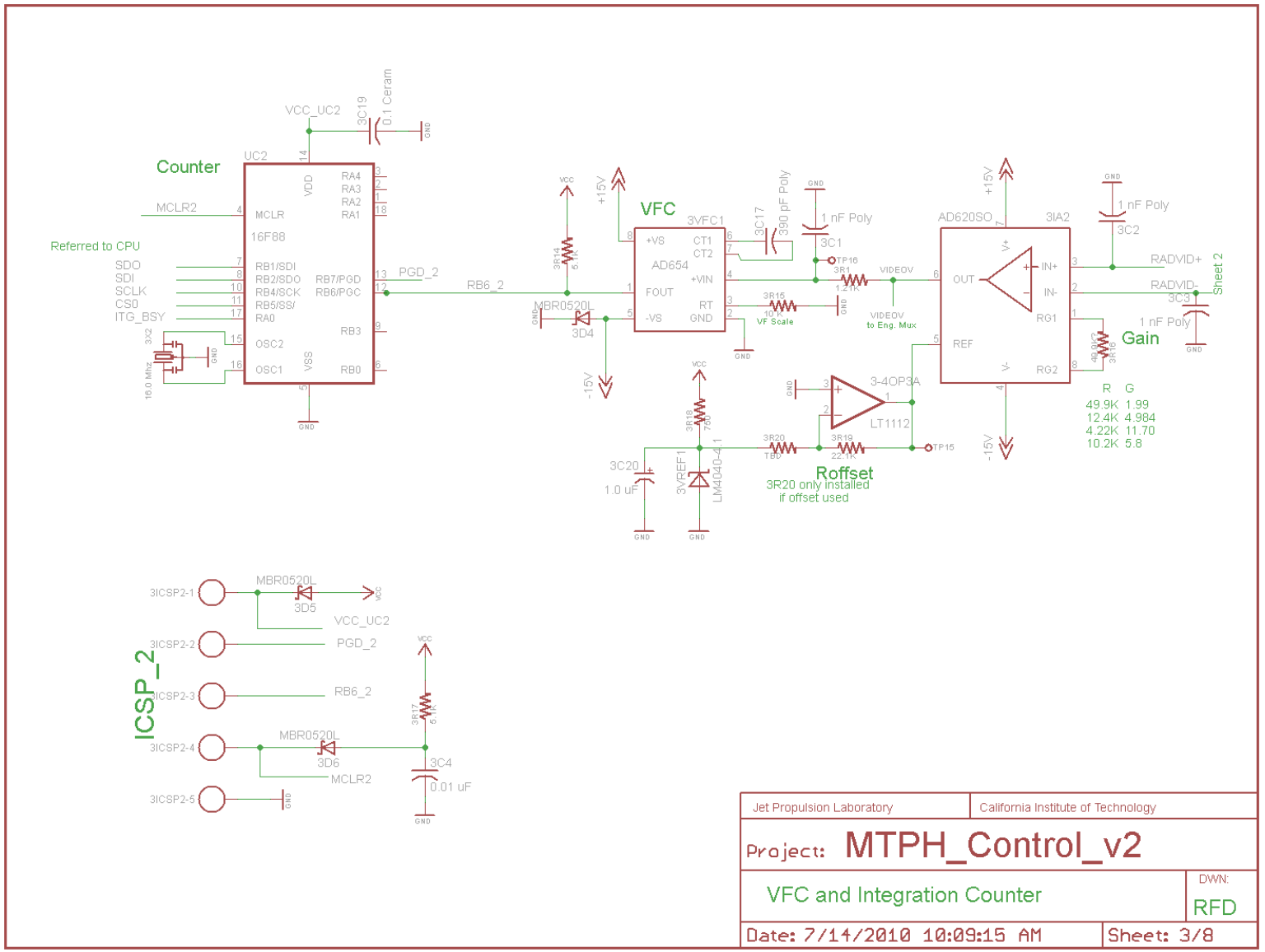
13.8 Sheet 8 - Power Distribution

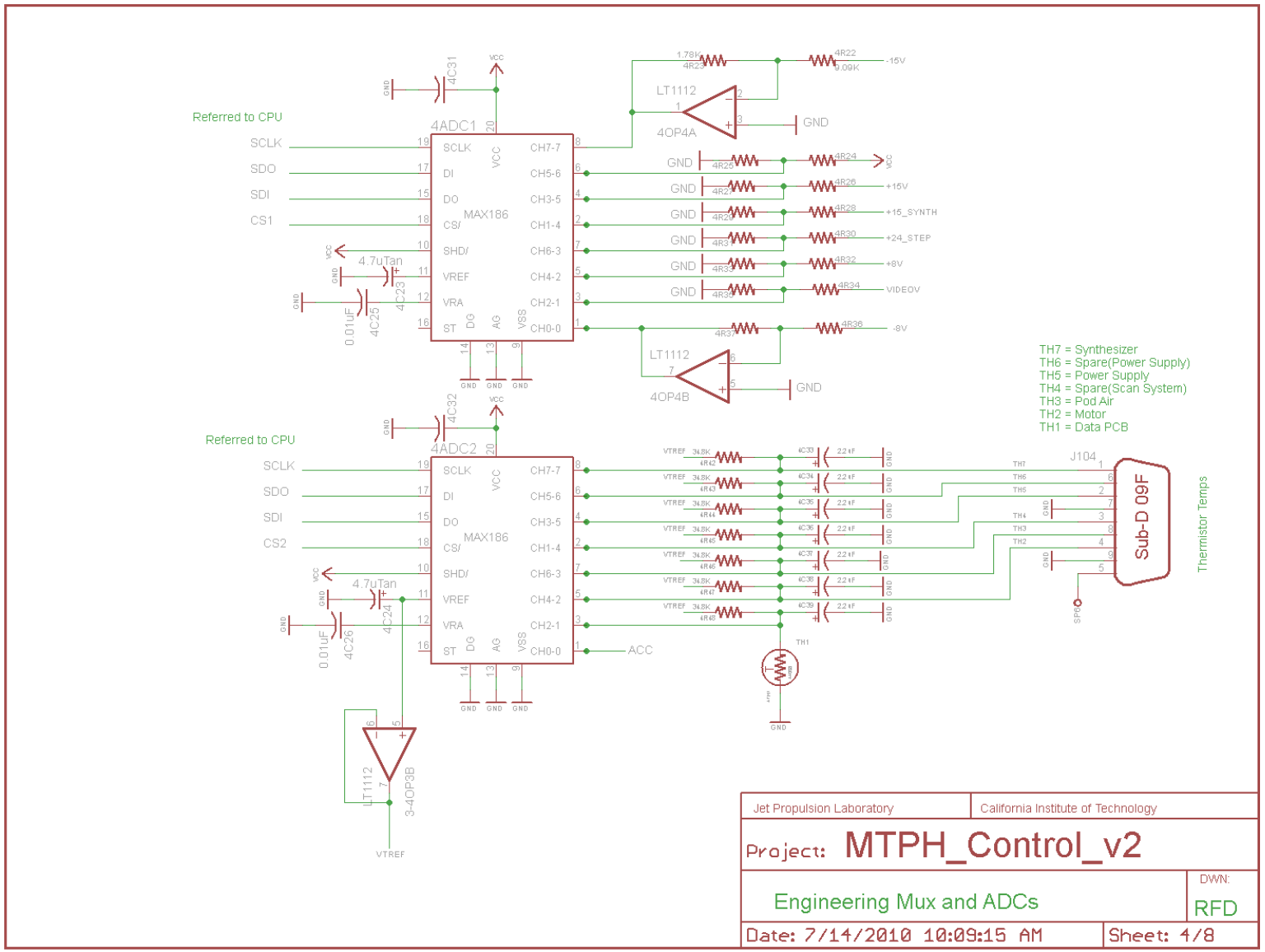
All DC power used on the *Controller Board* comes in on J106 and passes through simple capacitive filters before distribution. Two voltage regulators drop the +/- 15 V to +/- 8 V for use in the RTD measurement circuit on **Sheet 5**. R/C decoupling circuits prevent cross-coupling through the power supply between several op-amps on the board.

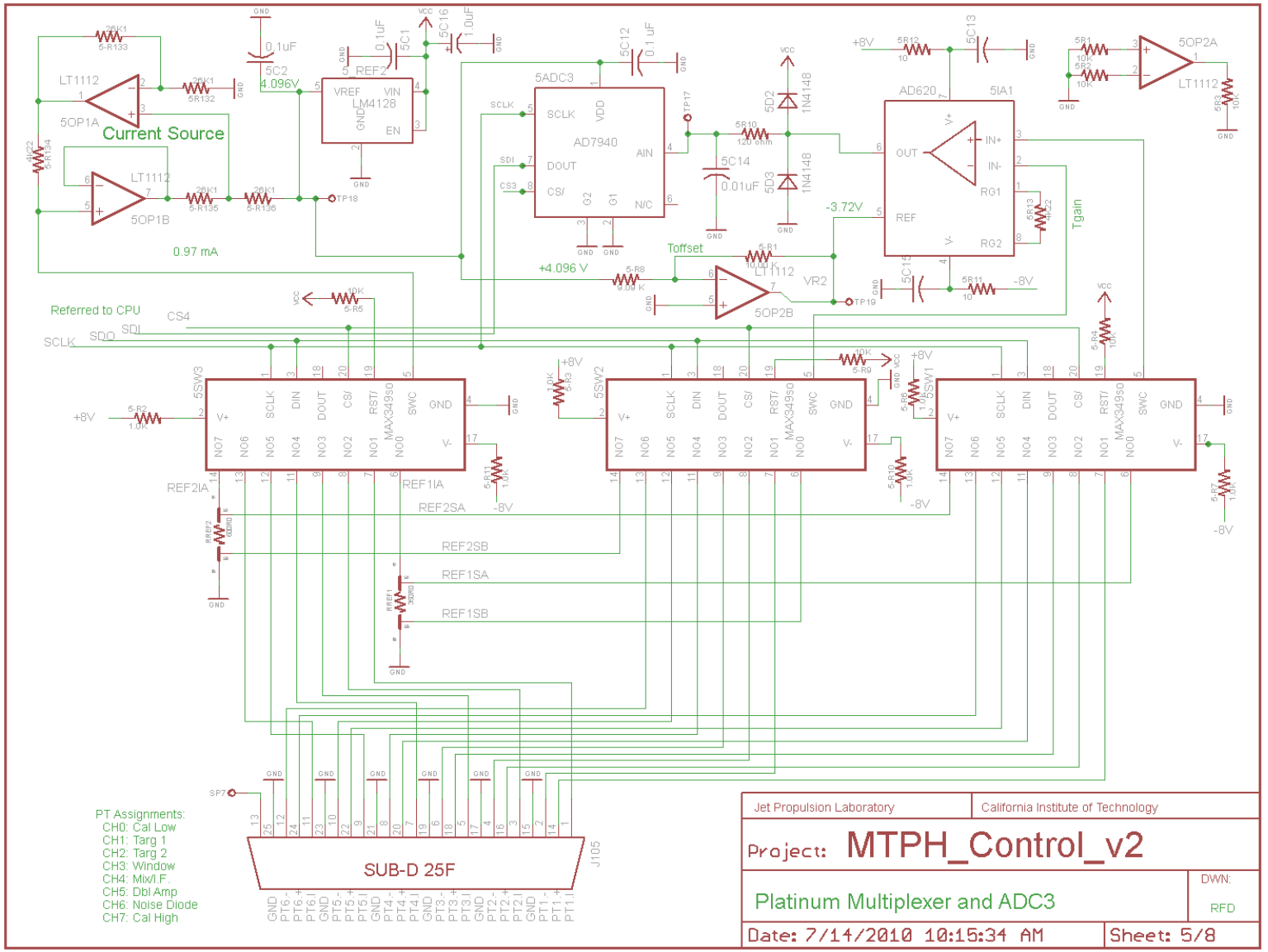


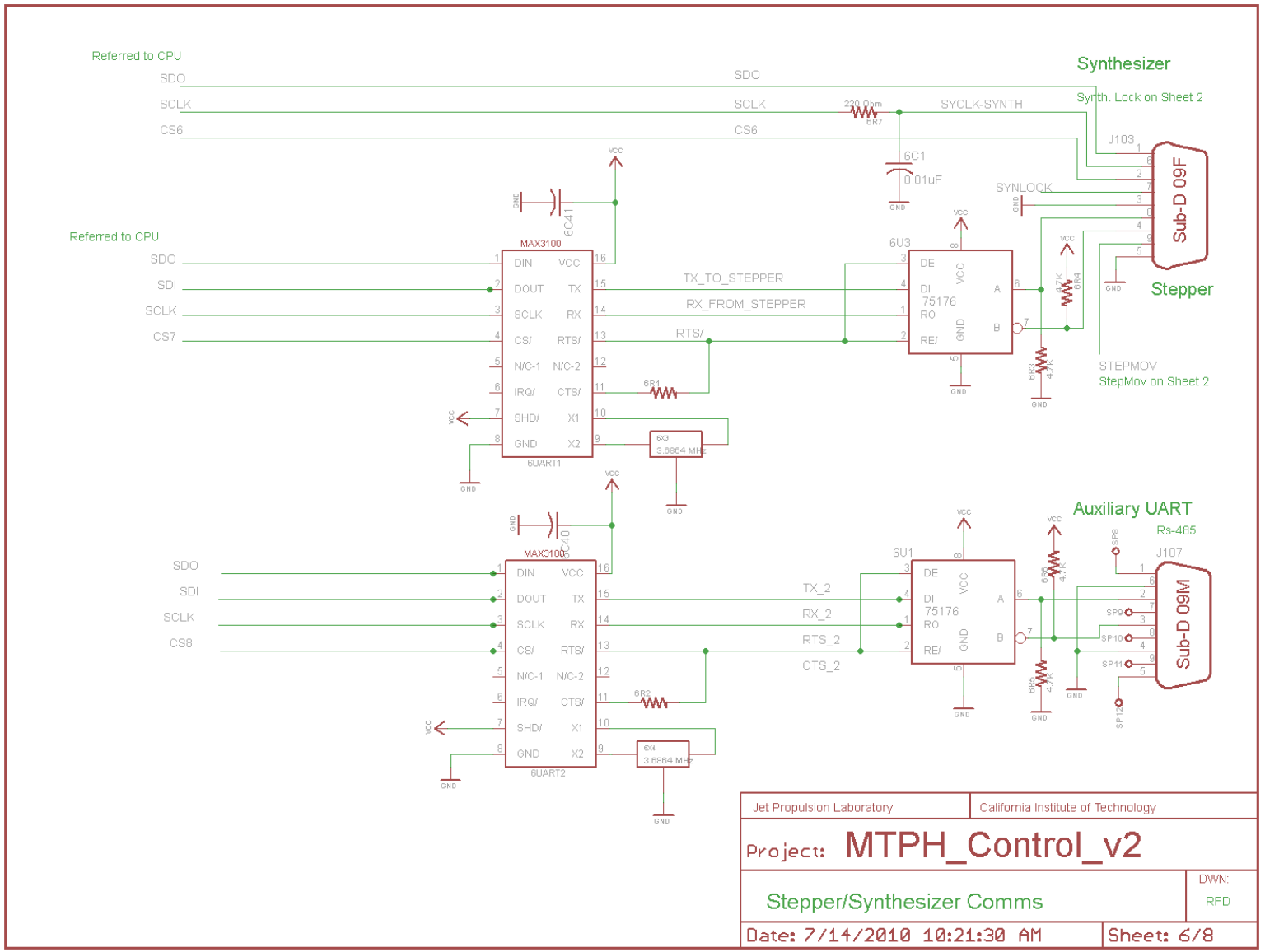

PIC 16F876
Serial I/O
Commands
Peripheral I/O
Sheet 2
RS-422
SPI Bus
Counter
(PIC 16F88)
VFC
Sheet 3
Engineering ADC's
Temps/Voltage
Sheet 4
Communications
Stepper/Synthesizer
Sheet 6
DC Distribution
Filters
Sheet 8
Platinum RTD
Multiplexer/ADC
Sheet 5
Accellerometer
Sheet 7
RS-485
Synthesizer
Module
Stepper
Card
RS-422
to Control Computer
Radiometer Plate
Scan System
Target
Jet Propulsion Laboratory
California Institute of Technology
Project: MTPH_Control_v2
Controller Block Diagram
DWN:
RFD
Date: 7/14/2010 9:39:21 AM
Sheet: 1/8

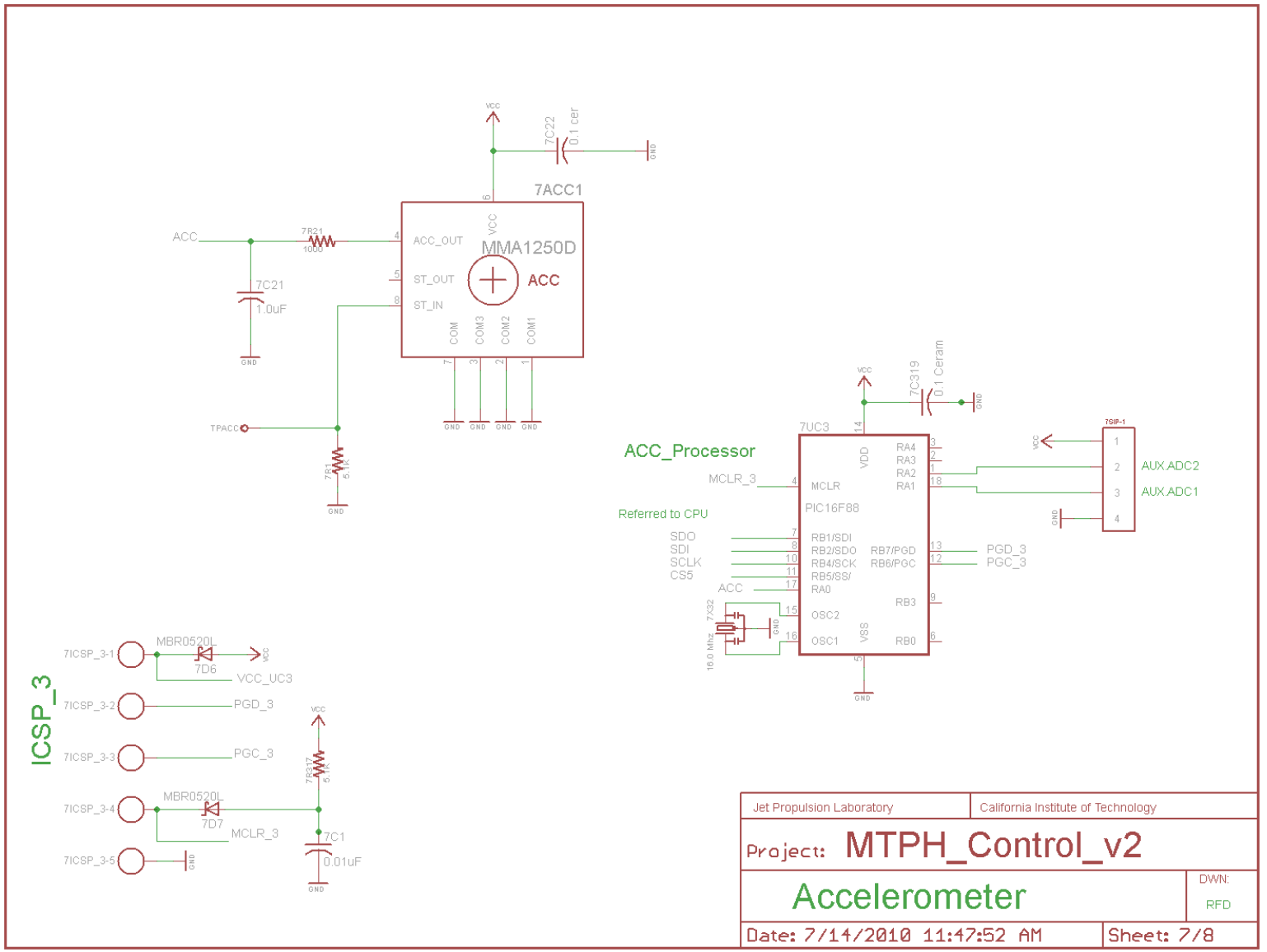

From PC
To PC
J101
Sub-D 09M
TD(A)-
TD(B)+
RD(A)-
RD(B)+
U1
U2
75176
UC1
PIC16F876A
MCLR_1
20.0 Mhz
ECS-ZTT-20.00MX
SPI Bus Signals
SDO
SDI
SCLK
CS7 UART(Stepper)
CS6 Synth.
CS5 Acc Processor
CS4 PtMux
CS3 ADC3 (Pt)
CS2 ADC2
CS1 ADC1
CS0 INTEG
CS9 Spare CS
CS8 Aux. UART
See Note
Value depends on Synth. Delay
ICSP_1
MBR0520L
MCLR_PU_1
Test Connector
Radiometer Video
To sheet 3
Intr
STEPMOV
SYNLOCK
Sheet 3
Sheet 8
Sheet 6
J102
Note: RA4(CS8) is open drain
Jet Propulsion Laboratory
California Institute of Technology
Project: MTPH_Control_v2
PIC Communication & I/O
DWN: RFD
Date: 7/14/2010 9:39:21 AM
Sheet: 2/8

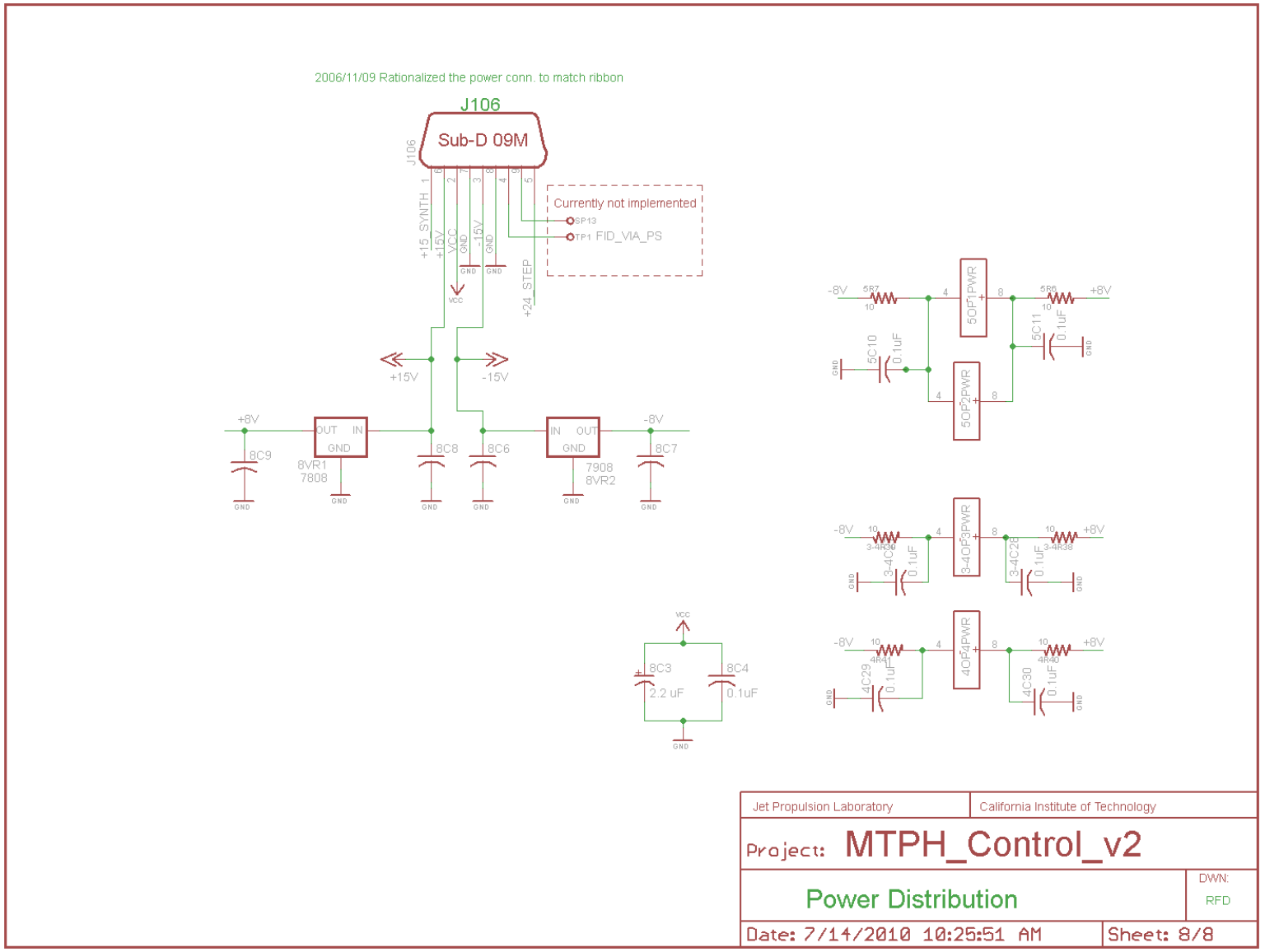

Counter
VCC_UC2
3C19
0.1 Ceram
UC2
MCLR2
Referred to CPU
SDO
SDI
SCLK
CS0
ITG_BSY
16F88
PGD_2
RB6_2
VFC
3VFC1
AD654
3C17
390 pF Poly
1 nF Poly
3C1
TP16
3R1
1.2K
VIDEOV
3R15
10K
VF Scale
VIDEOV
to Eng. Mux
MBR0520L
3D4
AD620SO
3IA2
1 nF Poly
3C2
RADVID+
RADVID-
3C3
1 nF Poly
Sheet 2
Gain
R G
49.9K 1.99
12.4K 4.984
4.22K 11.70
10.2K 5.8
3-4OP3A
LT1112
3R20
TBD
3R19
22.1K
TP15
Roffset
3R20 only installed
if offset used
3C20
1.0 uF
3VREF1
LM4040-4.1
3R18
750
ICSP_2
3ICSP2-1
3ICSP2-2
3ICSP2-3
3ICSP2-4
3ICSP2-5
MBR0520L
3D5
VCC_UC2
PGD_2
RB6_2
3R17
6.1K
MBR0520L
3D6
MCLR2
3C4
0.01 uF
Jet Propulsion Laboratory
California Institute of Technology
Project: MTPH_Control_v2
VFC and Integration Counter
DWN:
RFD
Date: 7/14/2010 10:09:15 AM
Sheet: 3/8


Referred to CPU
SCLK
SDO
SDI
CS1
4ADC1
MAX186
LT1112
4OP4A
4OP4B
GND
-15V
+15V
+15_SYNTH
+24_STEP
+8V
VIDEOV
-8V
4.7uTan
0.01uF
Referred to CPU
CS2
4ADC2
ACC
TH7 = Synthesizer
TH6 = Spare(Power Supply)
TH5 = Power Supply
TH4 = Spare(Scan System)
TH3 = Pod Air
TH2 = Motor
TH1 = Data PCB
J104
Sub-D 09F
Thermistor Temps
3-4OP3B
VTREF
Jet Propulsion Laboratory
California Institute of Technology
Project: MTPH_Control_v2
Engineering Mux and ADCs
DWN: RFD
Date: 7/14/2010 10:09:15 AM
Sheet: 4/8


Current Source
0.97 mA
+4.096 V
Toffset
Tgain
Referred to CPU
SUB-D 25F
PT Assignments:
CH0: Cal Low
CH1: Targ 1
CH2: Targ 2
CH3: Window
CH4: Mix/I.F.
CH5: Dbl Amp
CH6: Noise Diode
CH7: Cal High
Jet Propulsion Laboratory
California Institute of Technology
Project: MTPH_Control_v2
Platinum Multiplexer and ADC3
DWN: RFD
Date: 7/14/2010 10:15:34 AM
Sheet: 5/8


Referred to CPU
SDO
SCLK
CS6
Synthesizer
Synth. Lock on Sheet 2
SYCLK-SYNTH
220 Ohm
6R7
6C1
0.01uF
J103
Sub-D 09F
SYNLOCK
Stepper
Referred to CPU
SDO
SDI
SCLK
CS7
MAX3100
6C41
6UART1
TX_TO_STEPPER
RX_FROM_STEPPER
RTS/
6R1
6X3
3.6864 MHz
6U3
75176
STEPMOV
StepMov on Sheet 2
Auxiliary UART
Rs-485
SDO
SDI
SCLK
CS8
MAX3100
6C40
6UART2
TX_2
RX_2
RTS_2
CTS_2
6R2
6X4
3.6864 MHz
6U1
75176
J107
Sub-D 09M
Jet Propulsion Laboratory
California Institute of Technology
Project: MTPH_Control_v2
Stepper/Synthesizer Comms
DWN: RFD
Date: 7/14/2010 10:21:30 AM
Sheet: 6/8


7ACC1
MMA1250D
ACC
ACC_OUT
ST_OUT
ST_IN
VCC
7C22
0.1 cer
7R21
100
7C21
1.0uF
TPACC
7R1
5.1K
ACC_Processor
7UC3
PIC16F88
MCLR_3
Referred to CPU
SDO
SDI
SCLK
CS5
ACC
PGD_3
PGC_3
7C319
0.1 Ceram
7SIP-1
AUX.ADC2
AUX.ADC1
ICSP_3
MBR0520L
7D6
VCC_UC3
7D7
7R37
5.1K
7C1
0.01uF
Jet Propulsion Laboratory
California Institute of Technology
Project: MTPH_Control_v2
Accelerometer
DWN: RFD
Date: 7/14/2010 11:47:52 AM
Sheet: 7/8


2006/11/09 Rationalized the power conn. to match ribbon
J106
Sub-D 09M
Currently not implemented
SP13
TP1 FID_VIA_PS
+15 SYNTH
+15V
VCC
GND
-15V
+24 STEP
+8V
-8V
8C9
8VR1
7808
8C8
8C6
7908
8VR2
8C7
5OP1PWR
5OP2PWR
5R7
5R6
5C10
5C11
0.1uF
3-4OP3PWR
3-4OP4PWR
8C3
2.2 uF
8C4
0.1uF
Jet Propulsion Laboratory
California Institute of Technology
Project: MTPH_Control_v2
Power Distribution
DWN: RFD
Date: 7/14/2010 10:25:51 AM
Sheet: 8/8

14 MTP-H Wiring

The first figure below shows the cable drawing for all signals between the *MTP Power and Status Panel* in the cabin and the connector on the back of the DMT canister. The cabin portion of the cable will be long enough (45 feet) so that any rack location in the cabin can be reached via underfloor and crossover conduits from the Wing Connector #4.

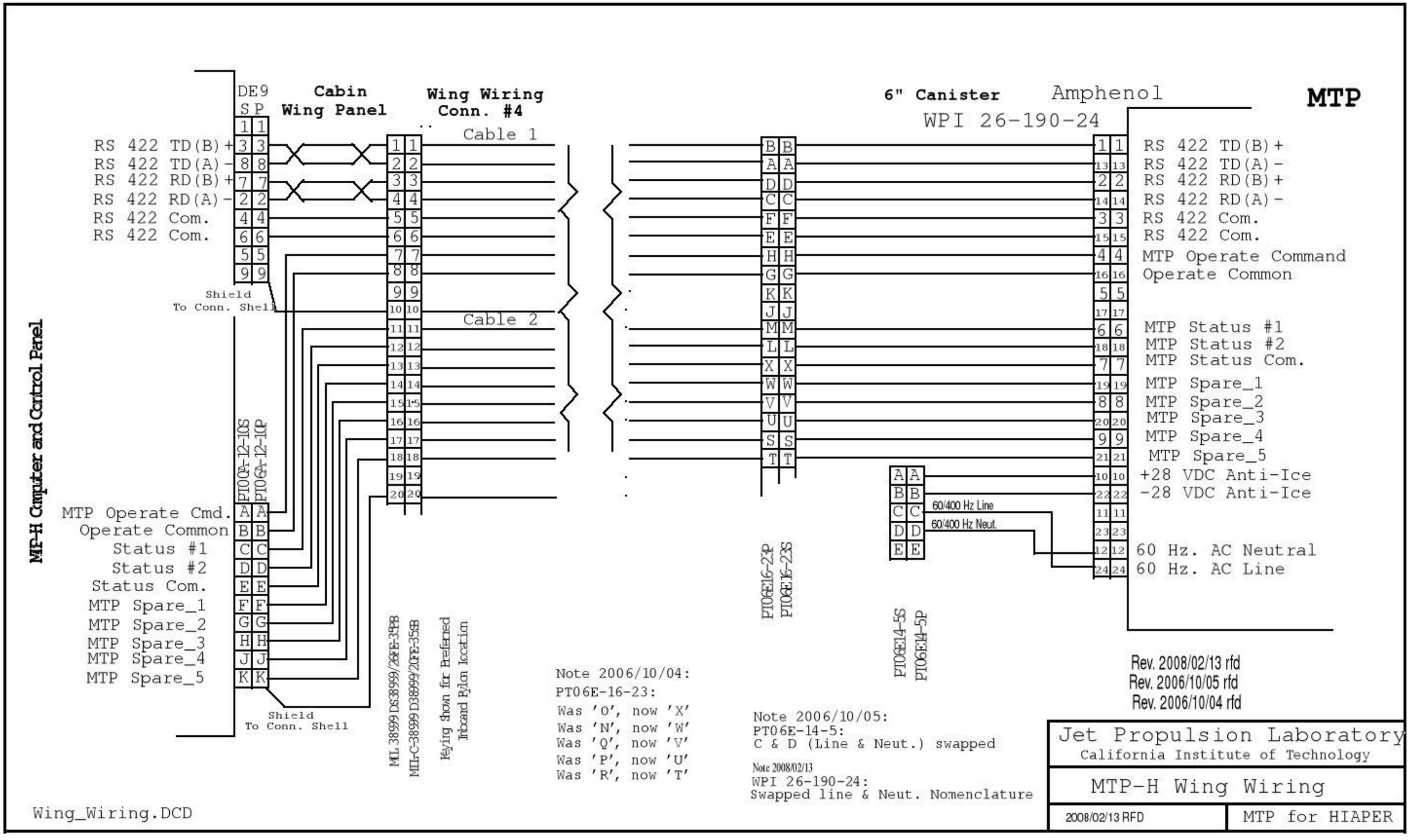
The next drawing shows a schematic of the major wiring blocks: 100 Block for the *Controller Board*, 200 Block for the *Power Supply and Synthesizer Plate*, 300 Block for the *Radiometer Plate*, and 400 Block for the *Scan Area*.

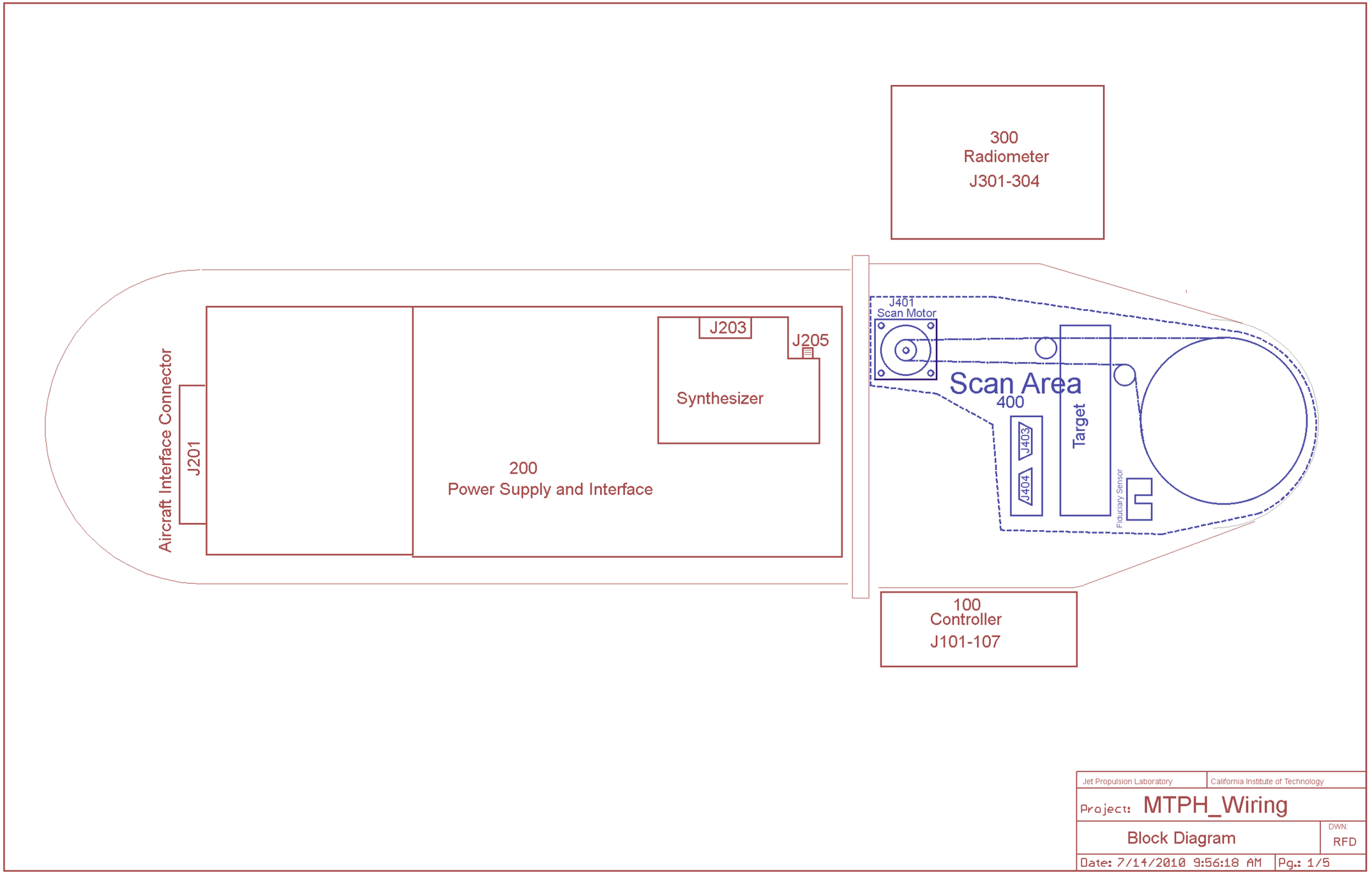
The third drawing shows the 200 Block wiring for the *Power Supply and Synthesizer Plate*. There are three classes of wiring in the MTP canister: 115 VAC 60 Hz power, low-voltage DC power, and control/data signals.

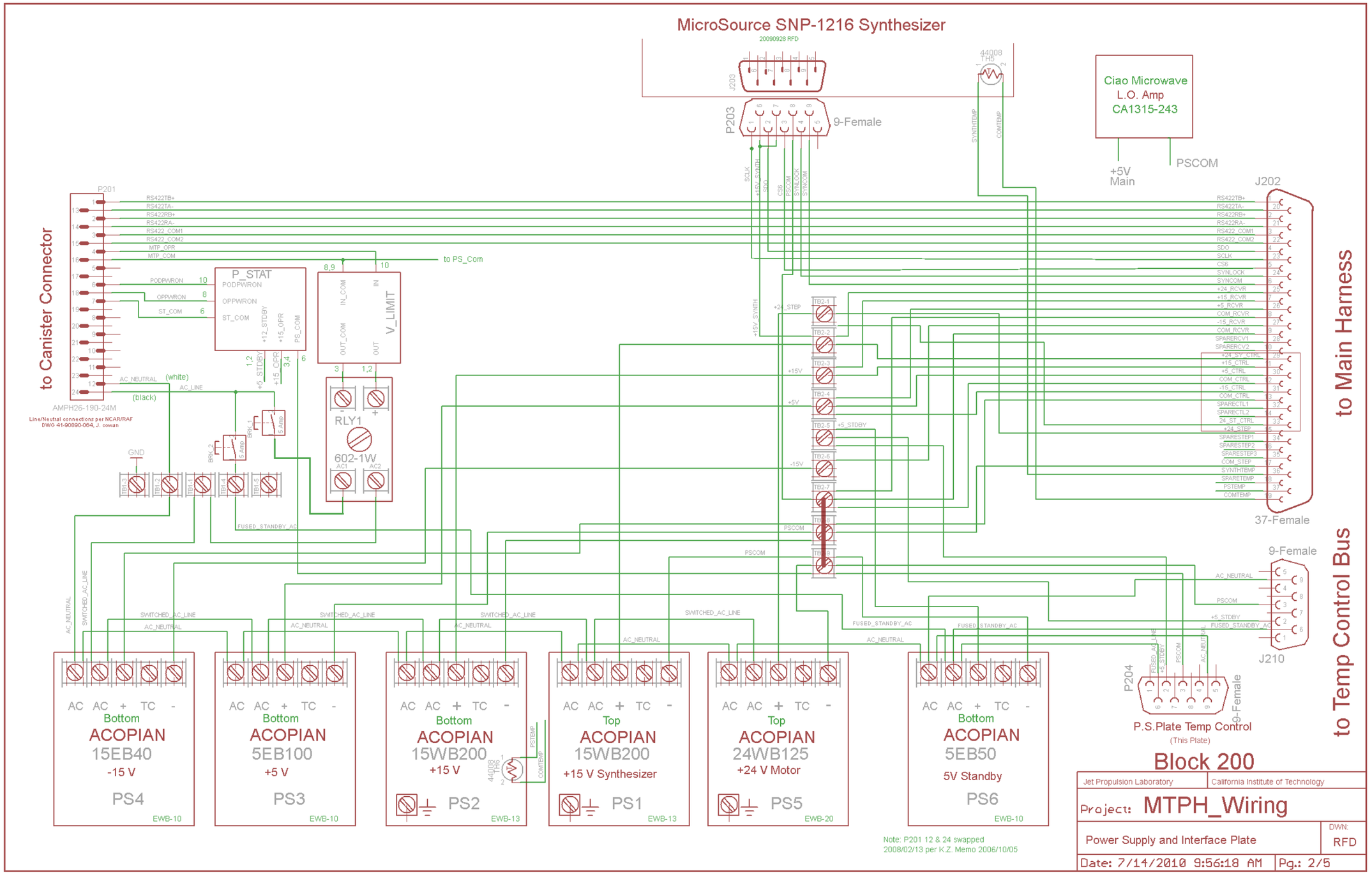
115 VAC power from the wing store passes directly from the connector through a 5 A breaker to a terminal strip (TB1). An Acopian power supply (PS6) operates from the protected AC line at TB1 to generate 12 VDC to power the temperature control circuits. The temperature control heater power is from the same protected 115 VAC line. A Teledyne 602-1W relay (RLY1), switches the AC power from TB1 to the remaining 5 Acopian power supplies on a signal from the cabin MTP *Power and Status Panel*. When this relay is **On**, the instrument is in **Operate** mode, and can receive and execute commands from the MCC.

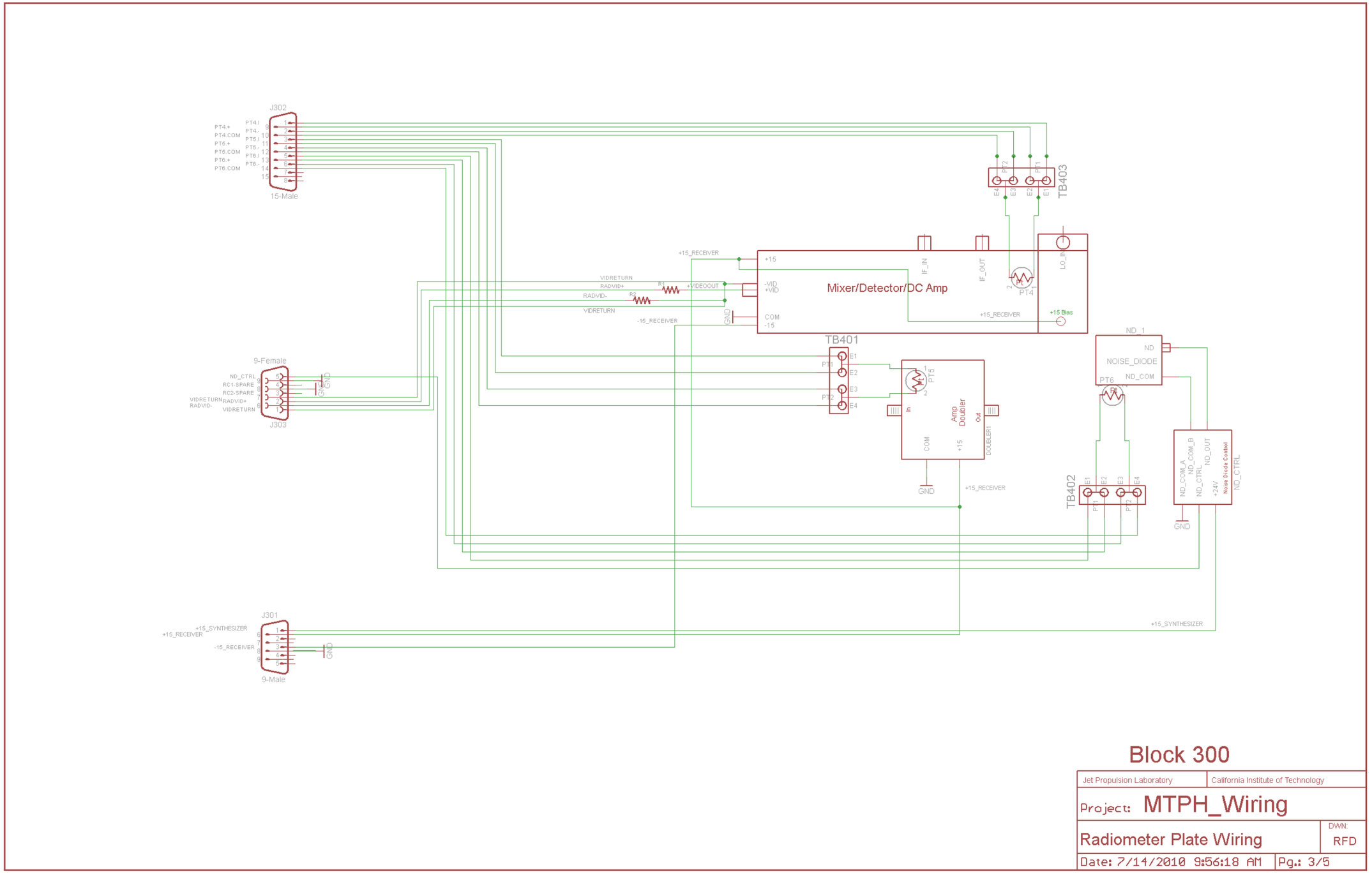
Low-voltage DC power is distributed to the various subsystems from another terminal strip (TB2). The synthesizer, being on the same plate as the power supplies, is wired directly to TB2 through connectors P203/J203. The *Temperature Control* subsystems are powered from TB1 and TB2 via J210 (Radiometer, Target, Controller enclosure) and P204 (*Power Supply/Synthesizer Plate*). All other subsystems receive DC power from TB2 via J202/P202 and the Main Wiring Harness (see last figure in this section).
The controller PCB is connected to all other subsystems by the Main Wiring Harness (Sheet 5 of MTP-H Wiring below). Through the Main Harness, commands are sent to the stepper controller, the radiometer and the synthesizer, and data (temperatures, voltages, radiometer output) is read back from all parts of the instrument.

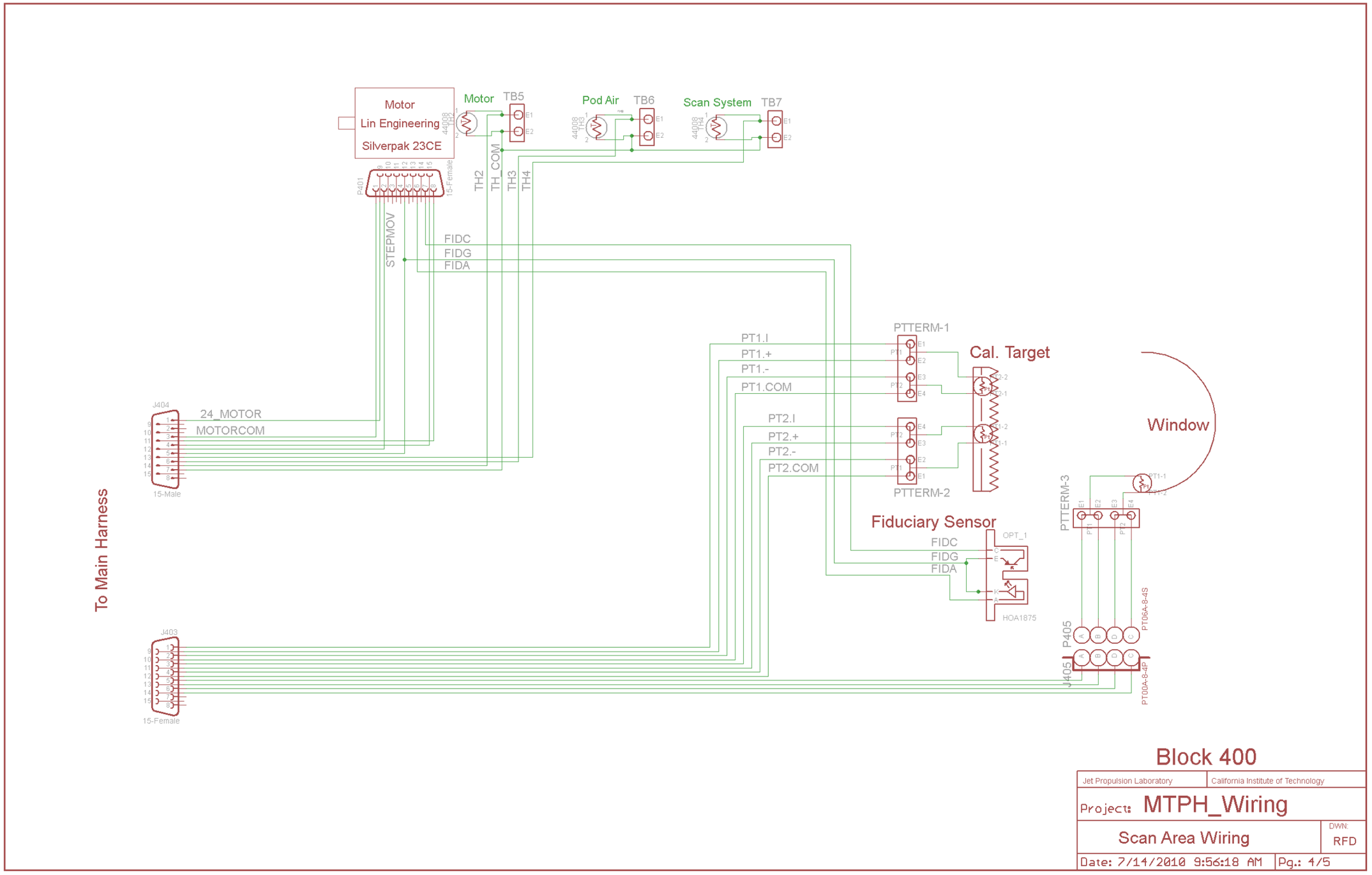
Drawings for the 300 Block (Radiometer plate) and 400 Block (Scan Area) can also be found below.

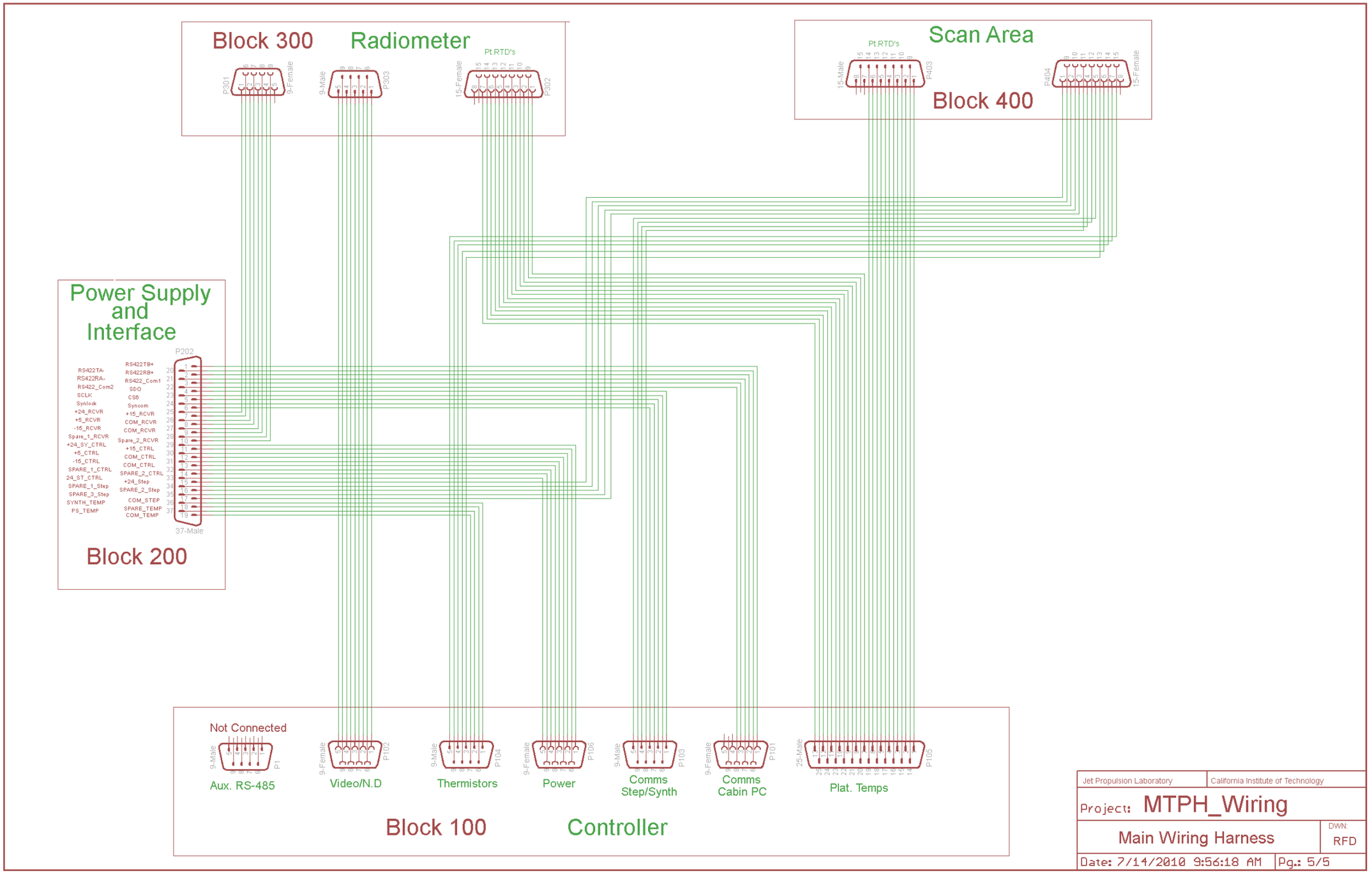

MTP-H Computer and Control Panel
DE9 S P
Cabin Wing Panel
Wing Wiring Conn. #4
Cable 1
Cable 2
6" Canister
Amphenol
WPI 26-190-24
MTP
RS 422 TD(B)+
RS 422 TD(A)-
RS 422 RD(B)+
RS 422 RD(A)-
RS 422 Com.
RS 422 Com.
Shield To Conn. Shell
PT00A-12-10S
PT06A-12-10P
MTP Operate Cmd.
Operate Common
Status #1
Status #2
Status Com.
MTP Spare_1
MTP Spare_2
MTP Spare_3
MTP Spare_4
MTP Spare_5
Shield To Conn. Shell
MIL-38999 D38999/26FE-39PB
MIL-C-38999 D38999/20FE-35SB
Keying Shown for Preferred Inboard Pylon Location
PT06E16-23P
PT06E16-23S
PT06E14-5S
PT06E14-5P
60/400 Hz Line
60/400 Hz Neut.
RS 422 TD(B)+
RS 422 TD(A)-
RS 422 RD(B)+
RS 422 RD(A)-
RS 422 Com.
RS 422 Com.
MTP Operate Command
Operate Common
MTP Status #1
MTP Status #2
MTP Status Com.
MTP Spare_1
MTP Spare_2
MTP Spare_3
MTP Spare_4
MTP Spare_5
+28 VDC Anti-Ice
-28 VDC Anti-Ice
60 Hz. AC Neutral
60 Hz. AC Line
Note 2006/10/04:
PT06E-16-23:
Was 'O', now 'X'
Was 'N', now 'W'
Was 'Q', now 'V'
Was 'P', now 'U'
Was 'R', now 'T'
Note 2006/10/05:
PT06E-14-5:
C & D (Line & Neut.) swapped
Note 2008/02/13
WPI 26-190-24:
Swapped line & Neut. Nomenclature
Rev. 2008/02/13 rfd
Rev. 2006/10/05 rfd
Rev. 2006/10/04 rfd
Jet Propulsion Laboratory
California Institute of Technology
MTP-H Wing Wiring
2008/02/13 RFD
MTP for HIAPER
Wing_Wiring.DCD


300
Radiometer
J301-304
J401
Scan Motor
Scan Area
400
J403
J404
Target
Fiducial Sensor
J203
J205
Synthesizer
Aircraft Interface Connector
J201
200
Power Supply and Interface
100
Controller
J101-107
Jet Propulsion Laboratory
California Institute of Technology
Project: MTPH_Wiring
Block Diagram
DWN:
RFD
Date: 7/14/2010 9:56:18 AM
Pg.: 1/5


MicroSource SNP-1216 Synthesizer
20090928 RFD
J203
P203
9-Female
44008 TH5
SYNTHTEMP
COMTEMP
Ciao Microwave
L.O. Amp
CA1315-243
+5V Main
PSCOM
J202
37-Female
to Main Harness
to Temp Control Bus
P201
to Canister Connector
AMPH26-190-24M
Line/Neutral connections per NCAR/RAF DWG 41-90890-064, J. cowan
AC_NEUTRAL
(white)
AC_LINE
(black)
GND
P_STAT
PODPWRON
OPPWRON
ST_COM
+12_STDBY
+15_OPR
PS_COM
to PS_Com
V_LIMIT
IN_COM
OUT_COM
IN
OUT
RLY1
602-1W
BRK_1
BRK_2
5 Amp
TB1-1
TB1-2
TB1-3
TB1-4
TB1-5
FUSED_STANDBY_AC
SWITCHED_AC_LINE
PSCOM
+15V_SYNTH
+2_STEP
+15V
+5V
+5_STDBY
-15V
TB2-1
TB2-2
TB2-3
TB2-4
TB2-5
TB2-6
TB2-7
TB2-8
TB2-9
P204
9-Female
J210
P.S.Plate Temp Control
(This Plate)
AC AC + TC -
Bottom
ACOPIAN
15EB40
-15 V
PS4
EWB-10
Bottom
ACOPIAN
5EB100
+5 V
PS3
EWB-10
Bottom
ACOPIAN
15WB200
+15 V
PS2
EWB-13
Top
ACOPIAN
15WB200
+15 V Synthesizer
PS1
EWB-13
Top
ACOPIAN
24WB125
+24 V Motor
PS5
EWB-20
Bottom
ACOPIAN
5EB50
5V Standby
PS6
EWB-10
Note: P201 12 & 24 swapped
2008/02/13 per K.Z. Memo 2006/10/05
Block 200
Jet Propulsion Laboratory
California Institute of Technology
Project: MTPH_Wiring
Power Supply and Interface Plate
DWN: RFD
Date: 7/14/2010 9:56:18 AM
Pg.: 2/5


J302
15-Male
PT4.+
PT4.COM
PT5.+
PT5.COM
PT6.+
PT6.COM
PT4.I
PT4.-
PT5.I
PT5.-
PT6.I
PT6.-
TB403
PT2
PT1
E4
E3
E2
E1
Mixer/Detector/DC Amp
+15
-VID
+VID
COM
-15
IF_IN
IF_OUT
LO_IN
PT4
+15_RECEIVER
+15 Bias
VIDRETURN
RADVID+
RADVID-
R1
R2
+VIDEOOUT
-15_RECEIVER
GND
TB401
PT1
PT2
E1
E2
E3
E4
PT5
Amp Doubler
In
Out
COM
+15
DOUBLER1
ND_1
ND
NOISE_DIODE
ND_COM
PT6
ND_COM_A
ND_COM_B
ND_CTRL
ND_OUT
+24V
Noise Diode Control
ND_CTRL
TB402
9-Female
J303
ND_CTRL
RC1-SPARE
RC2-SPARE
RADVID+
VIDRETURN
RADVID-
J301
9-Male
+15_SYNTHESIZER
+15_RECEIVER
-15_RECEIVER
Block 300
Jet Propulsion Laboratory
California Institute of Technology
Project: MTPH_Wiring
Radiometer Plate Wiring
DWN: RFD
Date: 7/14/2010 9:56:18 AM
Pg.: 3/5


Motor
Lin Engineering
Silverpak 23CE
Motor
TB5
Pod Air
TB6
Scan System
TB7
P401
15-Female
TH2
TH_COM
TH3
TH4
STEPMOV
FIDC
FIDG
FIDA
PTTERM-1
PT1.I
PT1.+
PT1.-
PT1.COM
PT2.I
PT2.+
PT2.-
PT2.COM
PTTERM-2
Cal. Target
Window
PTTERM-3
Fiduciary Sensor
OPT_1
HOA1875
P405
J405
PT06A-8-4S
PT00A-8-4P
J404
24_MOTOR
MOTORCOM
15-Male
J403
15-Female
To Main Harness
Block 400
Jet Propulsion Laboratory
California Institute of Technology
Project: MTPH_Wiring
Scan Area Wiring
DWN: RFD
Date: 7/14/2010 9:56:18 AM
Pg.: 4/5


Block 300
Radiometer
Pt.RTD's
P301
9-Female
9-Male
P303
15-Female
P302
Scan Area
Pt.RTD's
15-Male
P403
P404
15-Female
Block 400
Power Supply and Interface
P202
RS422TA-
RS422RA-
RS422_Com2
SCLK
Synlock
+24_RCVR
+5_RCVR
-15_RCVR
Spare_1_RCVR
+24_SY_CTRL
+5_CTRL
-15_CTRL
SPARE_1_CTRL
24_ST_CTRL
SPARE_1_Step
SPARE_3_Step
SYNTH_TEMP
PS_TEMP
RS422TB+
RS422RB+
RS422_Com1
SDO
CS6
Syncom
+15_RCVR
COM_RCVR
COM_RCVR
Spare_2_RCVR
+15_CTRL
COM_CTRL
COM_CTRL
SPARE_2_CTRL
+24_Step
SPARE_2_Step
COM_STEP
SPARE_TEMP
COM_TEMP
37-Male
Block 200
Not Connected
9-Male
P1
Aux. RS-485
9-Female
P102
Video/N.D
9-Male
P104
Thermistors
9-Female
P106
Power
9-Male
P103
Comms
Step/Synth
9-Female
P101
Comms
Cabin PC
25-Male
P105
Plat. Temps
Block 100
Controller
Jet Propulsion Laboratory
California Institute of Technology
Project: MTPH_Wiring
Main Wiring Harness
DWN: RFD
Date: 7/14/2010 9:56:18 AM
Pg.: 5/5

15 Temperature Controller PIC Software Listing

Beginning on the next page is the C code used to program the PIC on the *Temperature Controller Boards*.

C:\Documents and Settings\rdenning\Desktop\HIAPER_TEMP\Small_TC_C.c

```
//  California Institute of Technology
//  Jet Propulsion Laboratory
//  Richard F. Denning
//  MTP for HIAPER
//  Small_TC.c -- Interim version for breadboard
//  From TCPI_C.C
//
//  Temperature controller in PICC Lite for 12F675 with P, I terms
//  This version used for WAC-II temp control for AVE/Houston '05
//  Outputs P,I, PWM parameters as TTL serial on unused pin for
testing
//
//  03.03.24 - Translated from TCPID.asm - RFD
//  03.04.09 - Working version archived
//
//  Hardware Notes:
//  PIC12F675 uses internal 4 MHz oscillator
//  no External _MCLR connection required
//
#include <pic.h>     // code in pic.h gets the pic type from the
configure dialog
                     // and includes, in this case, 'pic12F6x.h'
//
//  Configuration Fuses
    __CONFIG(0x11C4);    //   Configuration Fuses:
                         //   - INTIO Oscillator
                         //   - 70 msecs Power Up Timer On
                         //   - Watchdog Timer Off
                         //   - Code Protection Off
                         //   - MCLR internal , I/O
                         //   - 70 msecs Power Up Timer On
                         //   - Watchdog Timer Off
                         //   - Code Protection Off
                         //   - BODEN Enabled
//  ---------------------------------------------------------------
//  ----------------------Global Variables-------------------------
volatile unsigned int RTC = 0;  //  Real Time Clock Counter
volatile unsigned int ADC0; // space for ADC readings
volatile unsigned int ADC1;
volatile unsigned int ADC3;
volatile unsigned char adtmp;
volatile unsigned char adstart;
//
signed int P;
signed int Pc;
signed int I;
signed int D;
signed int Ton = 0; // these are the PWM vars
signed int Toff = 0x200;
```

C:\Documents and Settings\rdenning\Desktop\HIAPER_TEMP\Small_TC_C.c

```
//
// these are used in the Serial out routines
unsigned char bitimr ;
unsigned int charout;
unsigned char charhi;
unsigned char charlo;
signed int sbcnt;
volatile char bval;
signed char tmp;

//
volatile char zz;
volatile int xx;
volatile int yy;
// const char version[17] ="TCPIC 041129rfd\0";
//
//  ------------------------------------------------------------
//  ----------------------Convert Byte to ascii hex-------------
void cvb2asc(char cvbin)
{
    charlo = cvbin & 0x0f;
    charhi = ((cvbin & 0xf0)/16);
//
    if (charhi > 9)
        charhi =charhi + 55;
    else
        charhi= charhi + 48;

    if (charlo > 9)
        charlo =charlo + 55;
    else
        charlo= charlo + 48;


}   // end cvb2asc
//  ------------------------------------------------------------
//  ----------------------Bit-Bang Serial Output----------------
void bittime(char bloops)
{

    bitimr = bloops;
    {
        #asm
            decfsz  _bitimr, f  ; must be a global
            goto $-1
    #endasm
    }
} // end bittime
//
```

C:\Documents and Settings\rdenning\Desktop\HIAPER_TEMP\Small_TC_C.c

```
// Emit
void emit(unsigned char emasc)  // sends emasc out on GPIO2 as RS-232
{
    charout = emasc ;
    GPIO2 = 1; // generate start bit
    bittime(22);
    bval=1;

        for (sbcnt = 0; sbcnt <= 7; sbcnt++ )
        {
        {
        if (charout & bval)
            GPIO2 = 0;  // note inversion for RS-232...
        else
            GPIO2 = 1;
        }
        bval=bval*2;
        bittime(22);

        }   // end for

    GPIO2 = 0; // generate 2 stop bits
    bittime(22);
    bittime(22);
}  //  End Emit
//
//  ---------------------Bit-Bang Serial Output----------------------
//  -----------------------------------------------------------------
//  ----------------------------version out--------------------------
void vout(void)

{
    char version[] = "TCPIC 050413rfd\0";
    char i;
    while (version[i] != '\0')
        {
        emit(version[i]);
        i++;
        }
    emit(0x0d);
    emit(0x0a);
}
//  -------------------------End of version
out--------------------------
//  -----------------------------------------------------------------

//  -------------------------------INITIO-------------------------
void initio (void)
{
```

3

C:\Documents and Settings\rdenning\Desktop\HIAPER_TEMP\Small_TC_C.c

```
    CMCON =     0b00000111; // Comparators off
    ADCON0 =    0b10000001; // ADC right justify, ADC enabled
    ANSEL =     0b00001011; // ADC Clk= Fosc/2, CH0, CH1, CH3 = ADC
    TRISIO =    0b00011011; // GPIO bits 2, 5 = output
    GPIO =      0b00000000; // Startup value for GPIO
}
//
//  ------------------------------INITIO-------------------------
//  -------------------------------------------------------------
//  ------------------------------ADC read-----------------------
int unsigned ADC(unsigned char adch)
{
    adtmp = (adch << 2) ;
    adstart = (adtmp | 0b10000001); // | = OR
    ADCON0 = adstart; // this selects the channel, but doesn't start
adc
    // put some settling time here for internal multplexer
    bittime(3);
    adstart = adstart | 2; // this starts it
    ADCON0 = adstart;

    while(ADCON0 & 2); // wait for /done bit

    return ((ADRESH << 8)+ ADRESL);
}

//  ------------------------------ADC read----------------------
//  ------------------------------------------------------------
//  --------------------------Send out 16 bit int---------------
void send16( unsigned int s16, unsigned char s16t)
{
        cvb2asc(s16 >> 8);
        emit(charhi);
        emit(charlo);
        cvb2asc(s16 & 0xff);
        emit(charhi);
        emit(charlo);
        emit(s16t);

}
//  --------------------Send out 16 bit int as ascii------------
//  ------------------------------------------------------------
//  ----------------------Long delays---------------------------
void ldelay (unsigned int ldt)
{
    yy = 0;
    while (yy <= ldt)
    {
        bittime(0xCC);
```

C:\Documents and Settings\rdenning\Desktop\HIAPER_TEMP\Small_TC_C.c

```
        bittime(0x66);
        yy++;
    }
}
//  ----------------------Long delays-------------------------
// ----------------------------------------------------------
//  ----------------------------ADC-------------------------------
void readADC(void)
{
        ADC0 = ADC(0);
        ADC1 = ADC(1);
        ADC3 = ADC(3);
}
//  ----------------------------ADC-------------------------------
// ----------------------------------------------------------
//  ----------------------------Report-------------------------------
void report(void)
{
        send16(ADC0,0x20);
        send16(ADC1,0x20);
        send16(ADC3,0x20);
        send16(P,0x20);
        send16(I,0x20);
        send16(D,0x20);
        send16(Ton,0x0d);    // cr to end the line
}

//  ----------------------------Report-------------------------
// ----------------------------------------------------------
//  ----------------------------PWM Calc-------------------------

    void pwmcalc(void)
    {
        signed int ptemp;

// Calc the current P value
        P = (ADC0 - 0x200);
        P = P * 16; // left shift 4, retain sign bit
        P = P + 0x100;
// integrate the error, check limits, then add in the Integration
    if ((ADC0 > 0x1F0) & (ADC0 < 0x210))     // don't do anything with
I
    {                                        // if far from null
        if (ADC0 > 0x200)    // increment or decrement I as needed
            I = I + 1;
        if (ADC0 < 0x200)
            I = I - 1;
//          I = I + (ADC0 - 0x200); // This should increase the
response
```

C:\Documents and Settings\rdenning\Desktop\HIAPER_TEMP\Small_TC_C.c

```
        if (I > 0x0FF)       // limit I to +/- 0xFF
            I = 0x0FF;
        if (I < -0x0FF)
            I = -0x0FF;
    }
    else
        I = 0;  //  sets I to zero if far from home
                // avoids wrap-up of I

//


//
        ptemp = P + Pc + I;

        if (ptemp < 0)
            ptemp = 0;
        if (ptemp > 0x200)
            ptemp = 0x200;
//      Set the PWM times
        Ton = ptemp;
        Toff = 0x200 - Ton;
// for diagnosis, will put out GPIO state as 'D' term which isn't
used now
        D = GPIO;
    }
//  ----------------------------PWM Calc-------------------------
// -------------------------------------------------------------
//  ----------------------------Cycle----------------------------
    void cycle(void)
    {
    GPIO5 = 1;
    ldelay(Ton);
    GPIO5 = 0;
    if (Ton == 0x200)    // ensure max time for full scale
        GPIO5 = 1;
    ldelay(Toff);
    }
//  ----------------------------Cycle----------------------------
// -------------------------------------------------------------
//  ----------------------------MAIN-----------------------------
//  Mainline
void main(void)              //  Mainline
{
//  hardware initialization
    initio();
    vout(); // sends out the version string on sign-on
//
```

C:\Documents and Settings\rdenning\Desktop\HIAPER_TEMP\Small_TC_C.c

```
    while (1)
    {
        readADC();
        pwmcalc();
        report();
        cycle();
    }
}  //  End of Main
//  ------------------------------MAIN-----------------------------
//  ---------------------------------------------------------------
```

7

16 Integration Timer and Counter PIC Software Listing

Beginning on the next page is the assembly language code used to program the PIC on the Controller Board that operates the integration timer and counter.

X:\HIAPER_firmware\F88_integ_IRQ.asm

```
; California Institute of Technology
; Jet Propulsion Laboratory
; Richard F. Denning
; Interim version for Breadboard
;
; Integration timer and counter for HIAPER MTP
;
; F88_Integ_IRQ.asm v. 20061023 rfd
; SPI i/o version of integrator/timer from Phoenix MTP
; "I" followed by byte (20mS increments) starts integrator and
; sets 'busy' hi, then low when done.
; "R 0 0 0 0" sends out "R" + 3 bytes of last counter value
;
; from 628Count.asm
; 7/24/02 added command "I" to implement integration timer/counter
;
    list P=16F88, R=DEC
    INCLUDE "p16f88.inc"
;
    CBLOCK 0x020
;
; integration timer/counter registers
;
dlylo
dlymd
dlyhi
tlo
tmid
thi
;
char_lo
char_hi
;
counterlo   ;   lo byte, counter store
countermid
counterhi   ;   3rd byte
;
inchar
ochar

xx          ; general purpose temporary
yy
zz

w_save          ; for interrupt service routine
status_save

    ENDC
;************************************************
```

1

X:\HIAPER_firmware\F88_integ_IRQ.asm

```
 PAGE

    __CONFIG    _CONFIG1, _CP_OFF & _CCP1_RB0 & _DEBUG_OFF &
_WRT_ENABLE_OFF & _CPD_OFF & _LVP_OFF & _BODEN_ON & _MCLR_OFF &
_PWRTE_ON & _WDT_OFF & _XT_OSC
    __CONFIG    _CONFIG2, _IESO_OFF & _FCMEN_OFF

    ORG 000H
    goto    main

    ORG 004H
    goto    int_serv
;
; Main line
main
;
;-------------------------------------------------------------
;   disable ADC's on Porta
    banksel ADCON0  ;bank 0
    clrf    ADCON0

    banksel ANSEL
    movlw   0x00
    movwf   ANSEL

;   turn off comparators
    banksel CMCON ; bank 1
    movlw   0x07
    movwf   CMCON

;-------------------------------------------------------
; SPI Setup
    movlw   0x40
    movwf   SSPSTAT     ; still bank 1

    banksel SSPCON      ; bank 0
    movlw   0x24        ; CKP=0
    movwf   SSPCON
;
; Set PORTA, B i/o state
    banksel TRISA
    movlw   0x10        ; Busy line = PORTA, 0 ... 1-3 outs for
diagnostic
                        ; 6, 7 don't exist
    movwf   TRISA       ; bank 1

    movlw   0xFB        ; all IN except RB2/SDO
    movwf   TRISB       ; bank 1
;
```

2

X:\HIAPER_firmware\P88_integ_IRQ.asm

```
;-------------------------------------
; serial setup (disable uart)
    bsf     STATUS, RP0     ; set to bank 1 for tx status reg
    bcf     TXSTA,  TXEN    ; disable transmit
;
    bcf     STATUS, RP0     ; set to bank 0 for rc status reg
    bcf     RCSTA,  CREN    ; disable reception
    bcf     RCSTA,  SPEN    ; disable Serial Port(ausart) (bank 0)
;
;-------------------------------------
    bcf     STATUS, RP0 ; back to bank 0
;-------------------------------------
; Setup default registers for counter/integrator
    movlw   0
    movwf   dlylo       ;these numbers yield an overall time of ~20ms
per dlyhi
    movlw   26
    movwf   dlymd
    movlw   10      ; 10=200 mS
    movwf   dlyhi
    goto    Chk4Dat

;
;*************************************
; Set up interrupt for timer 1 overflow
    banksel PIE1
    movlw   0x00 | (1 << TMR1IE) ; enable T1 int.
    iorwf   PIE1, f
    banksel INTCON
    bsf     INTCON, GIE
;
;*************************************
;   Wait here for next command
Chk4Dat
    banksel SSPSTAT
    btfss   SSPSTAT, BF
    goto    Chk4Dat
;
    ;if here, cmd chr is waiting in SSPBUF
    call    rcv
;
;-------------------------------------
; Command interpreter
    ; rcv returns here with cmd byte in inchar
    movfw   inchar
    sublw   'I'
    btfsc   STATUS, Z   ; if = 'I' then
    call    integ
;                       else
```

X:\HIAPER_firmware\F88_integ_IRQ.asm

```
    movfw   inchar
    sublw   'R'
    btfsc   STATUS, Z
    call    report
;
cixit
    goto    Chk4Dat
;  wait for next command
;   end of main
;-----------------------------------
;   common routines
on
    banksel PORTA
    bsf PORTA,  0
    return
;
off
    banksel PORTA
    bcf PORTA,  0
    return
;
;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
||||||||||||||
; SPI rcv/send routines
rcv
        ; on entry here, SSPSTAT, BF = chr ready, or wait for new char
    banksel SSPSTAT     ; bank 1
    btfss   SSPSTAT, BF
    goto    rcv

    banksel SSPBUF      ; Sw to bank 0
    movf    SSPBUF, w   ; rcv returns with chr in w as well as inchar
    movwf   inchar

    return ; from rcv
; end of rcv
;-------------------------------------------------------------------------
-------------
;
send2   ; sends ochar out on SPI
    banksel SSPSTAT         ; bank1
sloop2
    btfsc   SSPSTAT, BF     ; check if buffer empty
    goto    sloop2          ; no, loop back
    banksel SSPBUF          ; yes, load chr.
    movf    ochar, w
    movwf   SSPBUF          ; ready for transmission
    banksel SSPSTAT
sloop3
```

4

X:\HIAPER_firmware\P88_integ_IRQ.asm

```
    btfss   SSPSTAT, BF     ; check if buffer empty
    goto    sloop3          ; no, loop back

    banksel SSPBUF
    movf    SSPBUF, w
    movwf   xx              ; get rid of dummy input chr.
    ;
    return  ; from send2
; end send2
;-------------------------------------------------------------------------
-------------
;   Do an integration...
integ
    call    on      ; set bay hi, leave in bank 0
    call    rcv     ; get next byte (ascii= number of 20mS increments)
    movwf   dlyhi   ; rcv returns with number in inchar and w
    call    count
    call    off     ;bay lite off

    return  ; from integ to command interpret
;*************************************************************************

;   Integration timer/counter
;
; ~200mS DELAY SUBROUTINE WITH 4 MHz CLOCK- dlyhi=10, dlymd
;
count
    banksel dlylo       ; Everything in sub is on bank 0
    movf    dlylo, w    ; init. the delay loops
    movwf   tlo
    movf    dlymd, w
    movwf   tmid
    movf    dlyhi, w    ;#instructions*.25(instruction time,
uS)*12*256*256
    movwf   thi
;
    clrf    counterhi       ;Reset the count registers
    clrf    countermid
    clrf    counterlo
;
; make sure tmr1 is off and in correct mode: ext clk, async, tlon=0
    banksel TMR1H   ;TMR1H, L  bank 0, INTCON all banks
    movlw   06
    movwf   T1CON
    movlw   0
    movwf   TMR1H           ;set counter to zero
    movwf   TMR1L

    movlw   0xC1
```

5

X:\HIAPER_firmware\F88_integ_IRQ.asm

```
    movwf   INTCON
    ;bsf         INTCON, PEIE    ; enable interrupt
    banksel PIE1
    movlw   0x01
    movwf   PIE1
    banksel T1CON

;
; Start the counter
    bsf     T1CON, TMR1ON   ; start T1 counter
    banksel TMR1H   ; back to bank 0
;
cloop000
    movf    dlyhi, w        ; init. the delay loops
    movwf   thi
cloop00
    movf    dlymd, w
    movwf   tmid
cloop0
    movf    dlylo, w
    movwf   tlo
;
cloop
    decfsz tlo,F        ; basic timing loops
    goto    cloop
;
    decfsz tmid,F
    goto    cloop0
;
    decfsz  thi,F
    goto    cloop00
;
    ; done counting; tmr1 off , store counter values
    movlw   06
    movwf   T1CON   ; timer off
    bcf     INTCON, PEIE    ; disable periph. int.
    banksel PIE1
    bcf     PIE1, TMR1IE
    banksel TMR1H   ; back to bank 0
    ; counter hi is already current here (intserv)
    movf    TMR1H, w
    movwf   countermid
    movf    TMR1L, w
    movwf   counterlo
    return
; end count
;*********************************************************
;wait for SSPSTAT, BF
wtbf
```

6

X:\HIAPER_firmware\F88_integ_IRQ.asm

```
     ;call on
     banksel SSPSTAT
     btfss   SSPSTAT, BF
     goto    $-1
     ;call off
     return ;from wtbf
;*********************************************************
;
report
     call    wtbf

     banksel counterhi
     movf    counterhi, w
     movwf   SSPBUF
     call    wtbf

     banksel countermid
     movf    countermid, w
     movwf   SSPBUF
     call    wtbf

     banksel counterlo
     movf    counterlo, w
     movwf   SSPBUF
     call    wtbf
     ;
     return ; from report
;
;IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
; Interrupt service routine to handle counter roll-over
int_serv
     banksel status_save
     movwf w_save          ; save w
     swapf STATUS, f       ; save STATUS
     swapf STATUS, w
     movwf status_save

     banksel PORTA
     bsf PORTA, 1
     nop
     bcf PORTA, 1

     banksel PIR1
; deed is done here: PIR1 is in bank 0 also
     btfss   PIR1, TMR1IF    ; is it overflow?
     goto    restore         ; no, so exit

     banksel counterhi       ; yes, add 1
     incf    counterhi, f
```

X:\HIAPER_firmware\F88_integ_IRQ.asm

```
    bcf     PIR1, TMR1IF    ; mark as serviced


restore
    banksel status_save
    swapf status_save, f    ; restore STATUS
    swapf status_save, w
    movwf STATUS

    swapf w_save, f     ; restore w
    swapf w_save, w

    retfie
; End of int_serv
;IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
;////////////////////////////////////////////////////////////////////
    end
```

8

17 Control and IO PIC Software Listing

Beginning below is a listing of the C code used by the PIC on the Controller Board that performs all the control functions including IO with the MTP Cabin Computer.

X:\HIAPER_firmware\MTPH_Control.C

```
//  California Institute of Technology
//  Jet Propulsion Laboratory
//  MTP_H Control.C  **  MTP for HIAPER
//  Interim Breadboard Control Program
//  Compiled with Hi-Tech C in MPLAB IDE
//  Richard P. Denning
//  Version of 2006/11/29 added 'N (1/0) for ND control
//  Version of 2006/11/27 Baud rate increased to 19200, saving ~.9
Sec / cycle
//  Version of 2006/10/30 includes:  (Arguments are Hex)
//  F (0-9600) select channel for synthesizer, starts at 13550 mhz,
by 0.125 MHz steps
//
//  I nn = Integrate for nn * 20 mS
//  R    = read out the counter
//
//  M 1 = read all Mux 1 values
//  M 2 = read "    "  2
//  P = read all Platinum channels
//  S = read status:
//      bit 0 = iteg busy
//      bit 1 = Stepper moving
//      bit 2 = synlock = synthesizer alarm
//      bit 3 = spare
//  U $$$$$$$ = send $tring to Stepper UART and get response
//  V = return version date of this program
//  X aa dd ee ff gg etc  sends bytes ee ff, etc to SPI (CS=SS) and
reports
//      any response = Diagnostic only
// ---------------------------------------------------------------
//  Use TMR0, 1 msec interrupt as base for RTC ticks (from Myke
Predko)
//  Hardware Notes:
//  PIC16F876 running at 20 MHz uses HS oscillator
//  External _MCLR connection required (16F876 always has MCLR/)
//
// ---------------------------------------------------------------
// ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
#include <pic.h>
#include <stdlib.h>
#include <delay.h>  // need both for Ms delays
#include <delay.c>
#include <string.h>
// ---------------------------------------------------------------
//  Configuration Fuses
#if defined (_16F876)
#warning PIC16F876 selected
    __CONFIG(0x03F72);  //  PIC16F876 Configuration Word:
//  __CONFIG(0x03B72);  //  PIC16F876 Configuration Word:
```

X:\HIAPER_firmware\MTPH_Control.C

```
                        //    - HS Oscillator
                        //    - Power Up Timer On
                        //    - Watchdog Timer Off
                        //    - Code Protection Off
#else
#error Unsupported PICmicro MCU selected
#endif

// ------------------------------------------------------------------
// ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
// ------------------------------------------------------------------

//  Global Variables
bank1 volatile unsigned int RTC = 0;     //  Real Time Clock Counter
bank1 static char charry[16];
bank1 volatile bit trisLED1 @ (unsigned) &TRISC*8+1;     //  LED
Physical Bits
bank1 volatile bit      LED1 @ (unsigned) &PORTC*8+1;
bank1 volatile bit trisLED2 @ (unsigned) &TRISC*8+2;     //  LED
Physical Bits
bank1 volatile bit      LED2 @ (unsigned) &PORTC*8+2;


const int LEDon  = 1;        //  Declare values for LED ON/off
const int LEDoff = 0;
char pa, pal, pb, pc; // ram copy of ports value

unsigned char itemp, temp, temp2, temp3[2];
bank1 int iarg[6];
bank1 int inttemp;

char n;
char CharInIndex = 0;
char CmdString[24];
volatile char cmdRdy;
char txchar;
char CharOutIndex = 0;
char cmdi, mi;
unsigned int camask = 0x3FF; // represents the value for NO chip
selects active; digitizer is act. hi

//int *pt;
static bank2 char spin[8], spout[8], hex[8], nbytes;
bank1 static char StringOut[48];
bank2 char StringIn[16], itsave;
//bank2 char Stemp[32];
bank2 int adcint, adcinth, adcintl; //digtemp[4];
//bank2 char mi;
```

X:\HIAPER_firmware\MTPH_Control.C

```
bank3 int mux[16];

const char version[32] = "MTPH_Control 06/11/29 rfd\r\n";
//                          1 23456789012345678901234567890 12
// ------------------------------------------------------------------
// ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
// ------------------------------------------------------------------
// isr send chr
void iputch(unsigned char byte)
{
    while(!TXIF)    /* set when register is empty */
        continue;
    TXREG = byte;

    return;
}
//-----------------------------------------------------------------
// Serial I/O primitives
void putch(unsigned char byte)
{
    while(!TXIF)    /* set when register is empty */
        continue;
    TXREG = byte;

    return;
}

unsigned char getch(void)
{
    while(!RCIF)    /* set when register is not empty */
        continue;

    return RCREG;   /* RXD9 and PERR are gone now */
}
//
-------------------------------------------------------------------
-

void xmit(void)   // transmit a null-terminated string in StringOut
{
    char temp, i;
    i = 0;
    while((temp = StringOut[i++]) != 0)
        putch(temp);
}
//
-----------------------------------------------------------------
//
.................................................................
```

X:\HIAPER_firmware\MTPH_Control.C

```
.
//
//  Interrupt Handler
void interrupt intrAll_int(void)        //  All sources Interrupt
Handler

{

    if (T0IF) {

        T0IF = 0;           //  Reset Interrupt Flag

        RTC++;              //  Increment the Clock

//  Put additional interface code for 1 msec interrupt here

        if ((RTC % 512) == 0)
            {

            // pc= (pc ^ 4); //  XOR...Toggle LED Bit every 1024 msecs
            // PORTC= pc;
            //pc= (pc ^ 4); //  XOR...Toggle LED Bit every 1024 msecs
            PORTC= PORTC ^ 4;

            }
    }  //  endif

// end t0 services
//
.....................................................................
// uart rx services this is the 16F876 UART, not the MAX3100

    if (RCIF)           //  Serial Character Received
    {
        itemp = RCREG;
        switch (itemp)
        {
        case (0x08):            //  BackSpace?
            if (CharInIndex > 0)    //  Yes, and Something to Backspace
                {
                iputch(itemp);
                iputch(' ');
                iputch(itemp);
                CharInIndex--;
                }
            break;
        case (0x0A):            //  line feed, ignore
            break;
        case 0x0d:
```

X:\HIAPER_firmware\MTPH_Control.C

```
            {                       //  "Enter", End String and Print
            // iputch(CharInIndex + 64); // to report len of input
            CharInIndex = 0;
            cmdRdy = 1;
            }
            break;
        default:
            //if ((itemp >= 'a') && (itemp <= 'z'))
            //  itemp -= ('a' - 'A');    // make capa

            StringIn[CharInIndex] = itemp;
            CharInIndex++;
            StringIn[CharInIndex] = '\0';
            iputch(itemp);
            //TXREG = itemp;
        }
        RCIF = 0;               //  Reset Interrupt Request
    }  //  endif rcif
        RCIF = 0;               //  Reset Interrupt Request


}  //  End Interrupt Handler
// ---------------------------------------------------------------
// ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
// ---------------------------------------------------------------
 char ntox(nx)       // converts nibble to asc hex 0-f
    char nx;
{
    char hx [17] = "0123456789ABCDEF";
    return hx[nx];
}
// ---------------------------------------------------------------
// ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
// ---------------------------------------------------------------
 void btox(bx)       // converts byte to 2 asc digs in hex[]
    char bx;
{
    hex[0] = ntox((bx >> 4) & 0x0f);
    hex[1] = ntox(bx & 0x0f);
    hex[2] = '\0';                  // terminate the string
}   // end btox
// ---------------------------------------------------------------
 void lntox(bx)      // converts lo nibble of byte to 1 asc dig in hex[]
    char bx;
{
    hex[0] = ntox(bx & 0x0f);
    hex[1] = '\0';                  // terminate the string
}   // end btox
// // ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

5

X:\HIAPER_firmware\MTPH_Control.C

```
// ---------------------------------------------------------------
 void itox(ix)                  // converts integer to 4 asc digs(0) in hex[]
    int i
x;
{
    char bl, bh;
    bl = ix & 0xff;             // low byte
    bh = ((ix >> 8) & 0xff);    // high byte

hex[0] = ntox((bh >> 4) & 0x0f);
hex[1] = ntox(bh & 0x0f);
hex[2] = ntox((bl >> 4) & 0x0f);
hex[3] = ntox(bl & 0x0f);
hex[4] = '\0';                  // terminate the string

}   // end itox

// ---------------------------------------------------------------
// |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
// ---------------------------------------------------------------
 void chr2bin(char achr)
{
        char i, j;
        static char ts[11];
        ts[0] = '\0';
        j = 128;
            ts[0] = 'B';
        for (i = 1; i < 9; i++)
            {
            if (achr & j)
                ts[i] = '1';
            else
                ts[i] = '0';

            j = j >> 1;
            }
            ts[i+1] = ' ';
        strcpy(charry, ts);
    return;
}



// ---------------------------------------------------------------
// |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
// ---------------------------------------------------------------
void enableLED()                //  Set "eLED" according to
{
```

X:\HIAPER_firmware\MTPH_Control.C

```
        LED1 = LEDoff;          //  Start with LEDs off
        LED2 = LEDoff;

        trisLED1 = 0;           //  Make LED1 Bit Output
        trisLED2 = 0;           //  Make LED2 Bit Output

}  //  End enableLED

// ---------------------------------------------------------------
// |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
// ---------------------------------------------------------------
// UART Initialise
    void initUART(void)
{
//    Enable the USART
    TXSTA = 0x26;
    RCSTA = 0x96;
    SPBRG = 64;             //  19200 bps @ 20 MHz
//  SPBRG = 129;            //  9600 bps @ 20 MHz
//  SPBRG = 51;             //  9600 bps @ 8       MHz

    temp = RCREG;           //  clears RCIF
    RCIE = 1;               //  Enable Receive Interrupt
    TXIE = 0;               //  Disable Transmit Interrupt
    PEIE = 1;               //  Enable PIE Interrupt Sources
    CharInIndex = 3;
    return;
} // end initUART
// ---------------------------------------------------------------
// |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
// ---------------------------------------------------------------
// Send spnb bytes to SPI @ SPADD from spout[]
void SPX(spadd, spnb) // data is passed by globals spin[], spout[]
char spadd, spnb;
{
char i;
int spmask;
    --spnb; // need to fix this in several places

    PORTA = PORTA | 0x30;   // Set all CS's High
    PORTB = 0xFF;
    spmask = csmask ^ ( 1 << spadd ); // exclusive or: active low
    if (spadd <= 7)
    {
        PORTB = spmask & 0xFF;              // low byte, cs 0-7
        PORTA = PORTA | 0x30;
    }

    if (spadd == 8)
```

X:\HIAPER_firmware\MTPH_Control.C

```
        {
        PORTB = 0xFF;
        PORTA = PORTA & 0xEF;
        }

    if (spadd == 9)
        {
        PORTB = 0xFF;
        PORTA = PORTA & 0xDF;
        }

for (i = 1; i <= spnb; i++)
    {
    SSPIF = 0;              // clear flag
    SSPBUF = spput[i];      // set data to be sent
    while(!SSPIF) continue; // wait for buffer to be emptied
    while (!STAT_BF) continue;
    spin[i - 1] = SSPBUF;       // get received data

    // 16F88 seems to need CS to rise after each chr
    if (spadd == 0)
        {
        PORTB = 0xFF;       //toggle CS0 between chrs
        DelayUs(20);
        PORTB = 0xFE;
        }
    } // end for

    // Return to all CS's off
    PORTB = 0xFF;           // low byte, cs 0-7
    PORTA = PORTA | 0x30;   // high byte, cs 8&9

}
// end SPX
//---------------------------------------------------------------------
// Report n chars from spin[]
void report(nb)
char nb;
{
char ri;
nb--;
nb--;

        // strcpy(StringOut, "\r\n"); // return string started in Cmd Interp.
        btox(iarg[0]);
        strcat(StringOut, hex);
        btox(iarg[1]);
        strcat(StringOut, hex);
```

```
        strcat(StringOut, ">");

    for ( ri = 0; ri <= (nb - 1); ri++)
    {
    btox(spin[ri]);
    strcat(StringOut, hex);
    strcat(StringOut, " ");
    }

    strcat(StringOut, "\r\n");
    xmit();

}
// end report
//---------------------------------------------------------------------
// Wait for datardy from integrator
//void waititg()
//{
//  while((PORTC & 1) != 1);
//}
//  End of waititg
//---------------------------------------------------------------------
//
//||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
char parscmd(void)  // Parses CmdString, putting all #s into iarg[]
                    // returns number of args found
{
    char cmdlen, i, j; // loc in cmd$, substr #
        cmdlen = strlen(CmdString);
        for(i=0; i< cmdlen; i++)
        {
            if(CmdString[i] < '0') // first terminate sub-strings
                CmdString[i] = 0;
        }
        j = 0;
        for(i=0; i < cmdlen; i++)
        {
            if(CmdString[i] == 0)    // evauate the sub-strings
                {
                iarg[j] = atoi(CmdString + i + 1);
                j++;
                }
        }
    return j;
} // end parscmd
//---------------------------------------------------------------------
char uartst(char ust)
{
        spout[1] = 0x40;
```

X:\HIAPER_firmware\MTPH_Control.C

```
        spout[2] = 0;
        SPX(7,3);               // input goes to spin[0] and [1]

    return spin[0] & 0x80;
}
//-----------------------------------------------------------------
// toUART
void toUART()   // sends cmdstring(after init chr) to Stepper UART
{
    char tui;

    spout[1] = 0xC4;        // irq on rcv for diagnostic
    spout[2] = 0x0A;        // config. baud =9600 w/1.8342 xtal, 1 stop bit
    SPX(7, 3);              // spadd, spnb
    DelayMs(3);

    spout[1] = 0x84;        // set RTS, no transmission yet
    spout[2] = 0x00;
    SPX(7, 3);              // spadd, spnb
    DelayMs(3);

    spout[1] = 0x80;
    for(tui = 1; tui <= (strlen(CmdString)-1); tui++) // skip leading chr.
        {
            spout[2] = CmdString[tui];
            SPX(7, 3);      //SPX(7, 2); need to fix these
            DelayMs(10);        // wait for rs485 transmission
        }
    spout[2]=13;            //add CR
    SPX(7, 3);
    DelayMs(10);            // wait for rs485 transmission

//  spout[2]=10;            //add LF
//  SPX(7, 3);
//  DelayMs(10);            // wait for rs485 transmission

    spout[1] =  0x86;       // Release RTS
    spout[2] = 0;
    SPX(7, 3);

    spout[1] = 0xC4;        // irq bit on rcv chr.
    spout[2] = 0x0A;        // config. baud =9600 w/1.8342 xtal, 1 stop bit
    SPX(7, 3);              // spadd, spnb
    DelayMs(10);            // actually ~3 mS


} // end of toUART
//-----------------------------------------------------------------
// frUART
```

X:\HIAPER_firmware\MTPH_Control.C

```
void frUART()    // gets UART string fromStepper UART
{
    char uat,eos, frdx, di;

    // ''''''''''''''check UART status''''''''''''''''''''''''''''''''''''''''
    eos = 0;
    //StringOut has been started with "\r\nstep:" in cmd decoder
    frdx = strlen(StringOut);
    while(eos == 0)          // until end of string
    {
        while(uat == 0) // wait for rcv chr
        {
            uat = uartst(uat);
            DelayUs(10);             //
        } // end of while(uat == 0)

    // get chr
        spout[1] = 0;
        spout[2] = 0;
        SPX(7, 3);

//      PORTC = PORTC | 1;
//      DelayMs(1);          // actually ~.3 mS

    for(di = 0; di < 7; di++)
        DelayUs(214);  // estimated best for n = 7 = 0.96 mS
    //  DelayUs(200);            // 7 of these = 0.9 mS 2006/10/04
                                 // 6 bad, 7, 8 good, 9 bad

    //  PORTC = PORTC & 0xFE;


        switch(spin[1])  // Allmotion controller sends 'ETX' (= chr$(3)) at er
        {
            case 3:
                //eos = 1;
                break;
            case 0x0D:
                //eos = 1;
                break;
            case 0x0A:
                eos = 1;
                break;
            default:
                StringOut[frdx] = spin[1];
                frdx++;
                if(frdx > 45) frdx = 45;
                break;
        }   // end of switch
```

X:\HIAPER_firmware\MTPH_Control.C

```
    }   // end of while(eos == 0)
    spout[1] = 0xC4;        // irq bit on rcv chr.
    spout[2] = 0x0A;        // config. baud =9600 w/1.8342 xtal
    SPX(7, 3);              // spadd, spnb
    DelayMs(15);


    // got complete string now
    StringOut[frdx] = 0; // terminate string
}
// end of frUART
// ---------------------------------------------------------------
// ustat
void ustat(void)
{       // get Stepper UART status
    strcpy(StringOut, "\r\nS:");
    spout[1] = 0x40;
    spout[2]= 00;
    SPX(7, 3);
    btox(spin[0]);
    strcat(StringOut, hex);
    btox(spin[1]);
    strcat(StringOut, hex);

        strcat(StringOut, "\r\n");
        xmit();
}
// end of ustat
// ---------------------------------------------------------------
// reverse bit order in an integer
    int reverse(unsigned int x)
    {
    unsigned int h = 0;
    int i = 0;

        for(h = i = 0; i < 16; i++)
            {
                h = (h<<1)+ (x & 1);
                x >>= 1;
            }
    return h;
    }
// end of reverse
// ---------------------------------------------------------------
// ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
// ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
// ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
//  Mainline
```

X:\HIAPER_firmware\MTPH_Control.C

```
void main(void)             //  Template Mainline

{

//  hardware interface initialization
//  ADCON1 = 0x06;          // pg. 29 16F876 manual
    TRISA = 0b00001111;     // 4, 5 are chip selects 8, 9, 0-3 flags inputs, 6
    TRISB = 0;              // PB all outputs
    TRISC = 0xD0;           // 0=N.D. out , 1, 2 spare (LED or ??)
                            // 6,7 UART bits (pg 85)
    PORTA = 0x3F;           // CS's hi
    PORTB = 0xFF;

    TMR0 = 0;               //  Reset the Timer for Start
//  OPTION = 0xD1;          //  Assign Prescaler to TMR0
    OPTION = 0x02;          //  Assign Prescaler to TMR0
                            //  Prescaler is /8
    T0IE = 1;               //  Enable Timer Interrupts
    T0CS = 0;               //  Enable Timer
//  T0IE = 0;               //  Disable Timer Interrupts
//  T0CS = 1;               //  Disable Timer
////
    initUART();
    CCP1CON = 0;
//
    ADCON0 = 0;             // Disable ADC's
    ADCON1 = 0x06;
// Setup Timer2 for use as SPI clk
    T2CON = 0x05;           // on, prescale = 4
    PR2 =  0x01;            // normal SPI rate  ~ 330 KHz/1.5uS
//  PR2 = 0x10;             // 16 clocks = 3.125 uS            = 160 KHz
//  PR2 = 0x40;             // 64 clocks = 50 uS high/low = 10 KHz Synth. only
//  PR2 = 0x80;             // 128 clocks = 20 KHz

// Setup the SSP for SPI
    SSPCON2 = 0;            // these are all IIC bits
    SSPEN = 0;              // turn port off til set-up
    SSPSTAT = 0x40;         // 0xC0;
//  SSPCON = 0b00100010;    // 0010 = ckp=0,sspen=1,master, 1/64 clk
//  SSPCON = 0b00100001;    // 0010 = ckp=0,sspen=1,master, 1/16 clk
    SSPCON = 0b00100011;    // 0011 = "     "       "     , tmr2 cl
k
//

    GIE = 1;                //  Enable Interrupts
//  GIE = 0;                //  Disable Interrupts

//*******************************************************
// |||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

X:\HIAPER_firmware\MTPH_Control.C

```
//*******************************************************
    strcpy(StringOut, version); // send version as logon
    xmit();
    StringIn[0] = '0';
    CharInIndex = 0;

    while (1 == 1)  //  Loop forever
    {
//  Command interpreter
    if(cmdRdy)
        {
        strcpy(CmdString, StringIn);
        StringIn[0] = '\0';             // effectively Erase StringIn

        switch (CmdString[0]) // Should be first chr of cmd
            {
            case 'A':   // development diagnostic
                strcpy(StringOut, "\r\nA:");    // start the return string
                n = parscmd();          // parses input$ into as many #'s as i
                pb =(char) iarg[0];     // first number after "A"
                btox(pb);               // conv to hex$
                strcat(StringOut, hex);
                strcat(StringOut, " ");
                PORTB = pb;             // Stick all 8 bits into PB

                n =(char) iarg[1];      // second number after "A"
                btox(n);                // conv to hex$
                strcat(StringOut, hex);
                strcat(StringOut, " ");
                n = n & 0x03;           //  mask off unused bits
                pa = PORTA;             // get current PORTA value
                pa = pa & 0xFC;         // mask off our bits
                pa = pa | n;            // or it in
                PORTA = pa;

                strcat(StringOut, "\r\n");
                xmit();
                break;
            case 'F':
                strcpy(StringOut, "\r\nF");
                n = parscmd();      // parses input$ into as many #'s as it fi
                // Freq. number in iarg[0] now

                inttemp = iarg[0] << 2;         // pad two lsb 0's to make 14
                inttemp = reverse(inttemp);     // Synth. needs lsb first
                spout[1] = (inttemp >> 8);
                spout[2] = (inttemp & 0xFF);
                n = 3;
```

X:\HIAPER_firmware\MTPH_Control.C

```
            PR2 = 0x44;             // 68 clocks = 18 KHz/55uS for synth.

            nbytes = n;
            SPX(6, n);      // spadd = 6

            PR2 = 0x01;     // back to normal SPI rate  ~ 660 KHz/
1.5uS

            itox(iarg[0]);
            strcat(StringOut, hex);
            strcat(StringOut, "\r\n");
            xmit();
            break;

        case 'I':   //  "I 10" sends integration cmd to 16F88 integ.
            strcpy(StringOut, "\r\nI");
            n = parsecmd();      // parses input$ into as many #'s as it fi
            n = 3;
            spout[1] = 'I';      // out spi to 16f88
            spout[2] =(char) iarg[0];
            itsave = (char) iarg[0]; // save to put in  'R' response
            nbytes = n;
            SPX(0, n);      // spadd = 0

            btox(iarg[0]);
            strcat(StringOut, hex);
            strcat(StringOut, "\r\n");
            xmit();
            //report(nbytes);
            break;

        case 'M':   // read eng. mux iarg[0]
            strcpy(StringOut, "\r\nM");
            n = parsecmd();      // parses input$ into as many #'s as it fi
            btox(iarg[0]);
            strcat(StringOut, hex);
            strcat(StringOut, ":");

            for(mi = 0; mi<8; mi++)
            {
            //select channel
            spout[1] = 0b10001111;
            spout[1] = spout[1] | (mi << 4); // max186 cntrl word includes
            // spx goes from spout[1] to...
            spout[2] = 0;
            spout[3] = 0;
            nbytes = 4;                     // currently needs nbytes+1
            SPX(iarg[0], nbytes);
            // now right justify to 12 bits
```

X:\HIAPER_firmware\MTPH_Control.C

```
    adcinth = spin[1] * 256;
    adcintl = spin[2] ;
    adcint = (adcinth | adcintl)/8;
    spin[2] = (char) adcint & 0xff;
    adcint= adcint/256;
    spin[1] = (char) adcint & 0x0f;

    lntox(spin[1]);      // just the low nibble is significant
    strcat(StringOut, hex);
    btox(spin[2]);
    strcat(StringOut, hex);
    strcat(StringOut, " ");
    }
    strcat(StringOut, "\r\n");
    xmit();
    break;

case 'N':   //  "N (1/0)" N.D. On/Off
    strcpy(StringOut, "\r\nN");
    n = paracmd();       // parses input$ into as many #'s as it fi
    if(iarg[0]==1)
        PORTC=PORTC | 1;
    else
        PORTC=PORTC & 254;
    btox(iarg[0]);
    strcat(StringOut, hex);
    strcat(StringOut, "\r\n");
    xmit();

    break;


case 'P':   // read Ptmux iarg[0]
    strcpy(StringOut, "\r\nPt:"); // Return full pt mux
    for(mi = 0; mi<8; mi++)
    {
    //select channel
    spout[1] = (1 << mi);    // max349 uses one-of-eight addressing
                             // spx goes from spout[1] to...
    nbytes = 2; // currently needs nbytes+1
    SPX(4, nbytes); // CS 4 = pt mux
    // read adc
    DelayUs(10);
    spout[1] = 144;
    spout[2] = 0;
    spout[3] = 0;
    SPX(3, 4); // conversion goes into spin[1,2]

    // now right justify to 12 bits
```

16

X:\HIAPER_firmware\MTPH_Control.C

```
    adcinth = spin[1] * 256;
    adcintl = spin[2] ;
    adcint = (adcinth | adcintl)/8;
    spin[2] = (char) adcint & 0xff;
    adcint= adcint/256;
    spin[1] = (char) adcint & 0x0f;

    lntox(spin[1]);      // just the low nibble is significant
    strcat(StringOut, hex);
    btox(spin[2]);
    strcat(StringOut, hex);
    strcat(StringOut, " ");
    }
    strcat(StringOut, "\r\n");
    xmit();
    break;

case 'R':   // return Integration count, "Rii:" + 3 bytes msb firs
    strcpy(StringOut, "\r\nR"); // Return Count
    n = parsemd();  // get SPI addr into iarg[0]
    n = 6;          // this is the number to return incl. "R"
    btox(itsave);   // get the integration time last used
    strcat(StringOut, hex);
    strcat(StringOut, ":");

    spout[1] = 'R';
    for (cmdi=2; cmdi < n; cmdi++) // for loop builds up spout[] s
        {
        spout[cmdi] = 0; // these are dummy bytes to push data out
        }
    nbytes = n;
    SPX(0, n);
    for(cmdi = 2; cmdi < (nbytes-1); cmdi++)
        {
        btox(spin[cmdi]);
        strcat(StringOut, hex);
        }
    strcat(StringOut, "\r\n");
    xmit();

    break;

case 'S':               // return status; rc0 = datardy rc2 = bias
    strcpy(StringOut, "\r\nST:"); // made "ST" to dif. from Echo s
    btox(PORTA & 0b00000111); // was portc til 06/10/16 rfd
    strcat(StringOut, hex);
    strcat(StringOut, "\r\n");
    xmit();
    break;
```

X:\HIAPER_firmware\MTPH_Control.C

```
            case 'U':                 // String to Stepper UART
                strcpy(StringOut, "\r\nU:");
                strcat(StringOut, CmdString);
                strcat(StringOut, "\r\n");
                xmit(); // send echo etc.
                strcpy(StringOut, "Step:");
                toUART(); // send cmdstring to Stepper UART
                frUART(); // Get Stepper response strcat'd to StringO
ut
                strcat(StringOut, "\r\n");
                xmit();
                break;

            case 'V':       // version date
                strcpy(StringOut, "\r\nVersion:");
                strcat(StringOut, version);
                xmit();
                break;

            case 'X':   //  "X 01 dd ddd d" sends dec. numbers byte by byte to
                n = parscmd();              // parses input$ into as many #'s
                for (cmdi=1; cmdi < n; cmdi++) // for loop builds up spout[] s
                    {
                    spout[cmdi] =(char) iarg[cmdi];
                    btox(spout[cmdi]);              // conv to hex$
                    }
                nbytes = n;
                SPX(iarg[0], n);    // first number after "X" = spadd


                report(nbytes);

                break;

            default:
                strcpy(StringOut, "\r\nUnknown cmd\r\n");
                xmit();
                break;

        }

            cmdRdy = 0;
            //while(strlen(StringOut) > 4); // wait here for buffer to be sent
            strcat(StringOut, CmdString); // this prints out the last command
            }

    } //  endwhile
//*******************************************************
```

X:\HIAPER_firmware\MTPH_Control.C

```
}  //  End of Mainline
// ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

19

END OF DOCUMENT