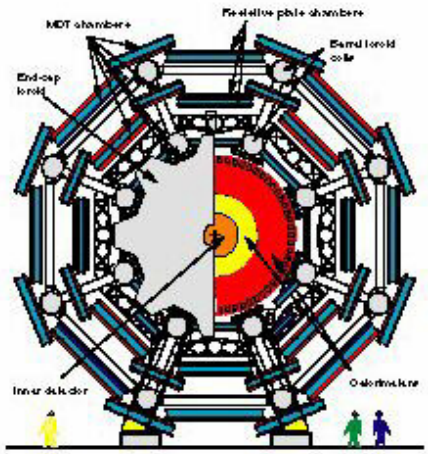


ATLAS

Muon Spectrometer

MDT–ASD Production Chip Tester

Firmware, Database, & Programming Manual

Manual Ver 1.00
Firmware Ver 1.00
Software(C++) Ver 1.01

27-Mar-03

E.Hazen, L.Kirsch, J.Oliver, M.Nudell, C.Posch

Except for this sentence, the page number at the bottom, and of course the header,
this page has been left intentionally blank

Table of Contents

0. General

All tests of the MDT-ASD will, in general, be performed by analog circuitry residing on the Chip Tester. This circuitry is operated under direct control of a Controller FPGA also residing on the Tester. In this way, lower level or "primitive" operations are controlled locally and do not require detailed traffic from the PC. The PC requests lower level commands by writing high level commands into a fifo called the "Inbox". The Controller reads these commands, performs the associated tests, and then deposits the results into another fifo call the "Outbox". Thus, the PC only need write high level commands into the Inbox, and then read resulting data from the Outbox. The Outbox data format is constructed in such a way that the command from which it came is appended to the data and is easily identifiable by the PC.

Commands, in general, are 16 bits in length. The upper byte of the command is reserved for a "Command Code" while the lower byte transmits other data such as channel number, DAC data, etc. For detailed codes, refer to the "Commands" spreadsheet at Commands .

0.0. Channel Address Format

The MDT-ASD has eight analog channels and so should require a 3-bit code for addressing. Each channel, however, has two input amplifiers and an LVDS output with two pads per channel. Thus for some tests, such as preamp and LVDS pad voltage measurements, a 4-bit code is required, while for other tests, such as Wilkinson parameters and noise, only a 3-bit code is used. The details in each case are specified in the Commands spreadsheet. But in any case, it's not that confusing, just get over it.

1. System functions

1.0. Version number

The Version Number refers to revisions of the Chip-Tester firmware. It is a three digit decimal number encoded as three hex digits (BCD : digits A-F will not be used).

Versions will be assigned as follow;

- 0.XX : Pre-release versions used in firmware development. It will not be incremented with every minor day to day change.
- 1.XX : Released firmware to be used in initial production testing.
- 2.XX : Reserved for major changes to Chip Tester functionality such as the implementation of new tests not included in earlier releases.

The three digits will be returned to the Outbox in a single 12 bit word and will be stored in the database as part of the record for each chip tested.

Test	Command	Hex Code	Outbox
1.2	Ver	10	(1) 12 bit word

2. String functions

All sub-registers of the ASD's 53 bit serial string are available for downloading and uploading. To download to the chip under test, a collection of sub-registers called the Default String Register (DFSR) must be loaded. These are static registers and will not be overwritten except via request by the PC. On downloading the chip, the DFSR is first transferred to a 53 bit shift register (Shadow) and then loaded into the Device Under Test, (DUT). On uploading, the 53 bit string is first retrieved from the DUT and its sub-registers may then be read out for comparison. *The DFSR does not change during uploading.*

2.0. Write DFSR Sub-Registers

Write Sub-strings to the Default String Register (DFSR) in the Controller.

Test	Command	Hex Code	Outbox
2.0	WDFSR	41 to 4B	-

2.1. Download DFSR

Download the DFSR via the Shadow register to the Device Under Test.

Test	Command	Hex Code	Outbox
2.1	DDFSR	50	-

Contents of the DFSR remain in the Shadow until overwritten by a DDFSR or UPSR command.

2.2. Upload Shadow Register

Upload the DUT string into the 53 bit Shadow string.

Test	Command	Hex Code	Outbox
2.2	UPSR	51	-

2.3. Read Shadow Sub-Register

Addressed Shadow register sub-strings are read out. To be a meaningful chip test, the Shadow register needs first to be Uploaded

Test	Command	Hex Code	Outbox
2.3	RSSR	61 to 6B	1 Byte

2.4. Write Pulse Width Register (PWR) and MODE

The **PWR** is a register in the controller which determines the pulse width of calibration (cal_inject) pulses. It is an 8 bits and results in a cal_inject pulse width of

$$(N+1)\cdot 16ns$$

where N is the register contents. Pulse width is programmable from 16ns to 4096 ns. The PWR need only be programmed once on power-up.

The MODE-Register determines whether the calibration circuits are set to Internal or External Modes. In External MODE, either EVEN or ODD channels may be selected as follows.

- (0x) Internal Mode
- (10) External EVEN channels
- (11) External ODD channels

Test	Command	Hex Code	Outbox
2.4	WPWR	52	-
2.4b	WMODE	55	-

2.5. Write DAC channels

The 12 bit DAC has two channels.

- Channel A ⇔ External Cal_inj pulse
- Channel B ⇔ Bias circuit adjust

While the DAC is accessed mainly through the higher level DC tests, it can be initialized through the following commands. For reasons of simplicity, it is treated as an 8-bit object when loading directly; the lower 4-bits will be downloaded as zeros.

Test	Command	Hex Code	Outbox
2.5	WDAC ChA ChB	53 to 54 53 54	-

3. DC Tests

These tests include DC measurements of amplifier input voltages, LVDS output voltages, and bias generator voltages. Measurements are all made with on board ADCs and multiplexers.

3.0. Bias circuit test

Using external DAC and current sense/limiting resistor, current into Vb2 node is swept from approximately –500 µA to +500 µA into Vb2 node in 256 steps. In addition to the four bias voltages Vb[1:4], the voltage at the current sense resistor is also monitored. These measurements are used to plot standard bias voltage characteristics as well as to extract nfet and pfet transistor parameters (K, Vth).

During this sweep, five voltages for each of 256 DAC settings are measured for a total of 1280 data points. These 12-bit ADC values are transmitted as 3 nibbles each for a total of 3840 fifo memory cells. See Commands.xls for detailed description of data structure.

Test	Command	Hex Code	Outbox
3.0	BTEST	80	3840 nibbles

We expect each chip to have a slightly different nominal voltage at its Vb2 pin. Therefore, the value of the DAC setting corresponding to zero additional bias current will be different for each chip. This value is extracted from the bias test data and is used in all subsequent tests of the chip. Thus, the bias test is the first test done in the chip test sequence.

3.1. Preamp Input Voltage

Command measures preamp input voltages for all channels. 12-bit result data are returned as three consecutive nibbles for each Signal and Dummy channel.

Test	Command	Hex Code	Outbox Data
3.1	PIV	90	48 nibbles

3.2. LVDS levels

Command measures LVDS output voltages for all channels. 12-bit result data are returned as channel number along with three consecutive nibbles for each of the eight “OUTa” and “OUTb” levels. The results of this test will, of course, depend on the state of the LVDS outputs. Normally, these will sit LO but are forced either HI or LO prior to the test.

Test	Command	Hex Code	Outbox
3.2	LVDS	A0	48 nibbles

3.3. Total ASD current

Returns total current drawn by the DUT. 12-bit result is transmitted as three consecutive nibbles.

Test	**Command**	**Hex Code**	**Outbox**
3.3	ITOT	A4	3 nibbles

4. Dynamic Tests

4.0. Noise test

Test	**Command**	**Hex Code**	**Outbox**
4.0	NOISE	B0	3 nibbles

The noise test measures the rate of thermal noise hits as a function of threshold setting. To perform the Noise Test, the programmer must;

- download a **threshold setting**
- set the Chip Mode to **TOT**
- set up desired output channels to Active Mode
- set Disc 1 Hysteresis to **minimum value**

The noise tests are done by counting a fixed number of pulses, N_{max}, and returning the elapsed time required to count them. This method provides a constant error of $\frac{dR}{R} = \frac{1}{\sqrt{N_{max}}}$ for each data point which is useful for Gaussian curve fitting. In principle, noise hits from all eight channels can be counted simultaneously. Total test time in this scenario will be dominated by very quiet channels which will cause the timer to overflow at some fixed value. This value is chosen to keep the total test time under one second.

Elapsed time will be reported in units of clock periods (16 ns) in floating point format as follows

$$T = N \cdot 2^{E}$$

where N and E are the mantissa and exponent respectively. Detailed parameter values are given below

Parameter	Value
Maximum pulse count : N_{max}	32
Mantissa	8-bits
Exponent	4-bits
Maximum exponent : E_{max}	14
Timer dynamic range	21 bits
Timer overflow time	68 ms
Estimated test time	1.2 sec
Threshold range	- 44 mv to + 44 mv, 2mv steps

In practice, the tester socket, even though of a very low inductance design, does not allow for stable operation of all eight channels simultaneously at very low threshold settings. We have found that eight enabled channels will oscillate in unison at low thresholds if all are enabled in the tester socket. Stable operation is seen with only four of eight channels enabled, and for an adequate safety margin, we have chosen to enable only two channels at a time. The test must therefore be performed four times.

For each noise test performed with the B0 test code, two data words are returned per channel, or four words total as follows;

Test	Command	Hex Code	Outbox Data
4.0	NOISE	B0	Ch(A) mantissa
		B1	Ch(A) exponent
		B0	Ch(A+4) mantissa
		B1	Ch(A+4) exponent

In the above table, “A” refers to the channel which has been downloaded along with the B0 command. The firmware automatically returns both Channel A data and Channel A+4 data. Thus the command is issued four time with A=0,1,..,3

4.4. Wilkinson functions – general

All Wilkinson tests involve measurement of pulse width of the output LVDS signals. A timestamp TDC is implemented within the ChipTester FPGA which has a resolution of ¼ of a clock cycle or 4 ns. Thus, for all Wilkinson tests, the returned codes LSB corresponds to this interval.

4.5. Wilkinson gate

Test	Command	Hex Code	Outbox
4.5	WGATE	C5	3 nibbles

In Wilkinson Mode, varying the Wilkinson Gate DAC varies the LVDS output pulse width. In Time-over-Threshold Mode, pulse width is narrow and constant. This test (ADC mode only) is used to measure the LVDS pulse width as a function of Wilkinson gate width. For a fixed set of parameters (injection signal, VTH1, VTH2, rundown current), leading and trailing edge of output pulse are recorded on all of 8 channels for all sixteen Wilkinson gate DAC settings.

4.6. Wilkinson rundown

Test	Command	Hex Code	Outbox
4.6	WRD	C6	3 nibbles

This will be done in software using **TimeStamp** (5.5) command

In Wilkinson (ADC) Mode, standard values of cal_inject and threshold are used and the leading and trailing edges are time stamped for all ASD channels. Pulse width is measured as a function of rundown current

4.7. Disc2 threshold

Test	Command	Hex Code	Outbox
4.7	D2THR	C7	3 nibbles

This will be done in software using **TimeStamp** (5.5) command

In Wilkinson Mode, vary the Disc2 threshold setting and observe the effect on output pulse width. For a fixed set of parameters (injection signal, VTH1, rundown current, Integration Gate), leading and trailing edge of output pulse are recorded on all channels for each of the sixteen Vth2 DAC settings.

4.8. Wilkinson transfer characteristic

Test	Command	Hex Code	Outbox
4.8	WILK	C8	3 nibbles

This is implemented in software using **TimeStamp** (5.5) command

In Wilkinson Mode, vary the input amplitude and observe the effect on output pulse width. For a fixed set of parameters (Vth1, Vth2, Wilkinson Integration Gate, Wilkinson Rundown Current), leading and trailing edge of output pulse are recorded on all of 8 channels. Programmer downloads a new Cal_inject DAC value and repeats test as desired to reconstruct transfer characteristic.

4.9. Deadtime

In Wilkinson Mode, the Deadtime is measured by varying the separation of a pulse pair and observing the emergence of the second pulse. A binary search routine is used to locate Deadtime for each of the eight possible values of the Deadtime DAC. The Wilkinson Gate Width and Cal_inject pulse width must be carefully selected during this test. Both must be fairly narrow, with the Cal_inject width being somewhat larger than the Gate but within the typical rundown time. This insures that the "wrong sign" trailing edge of the Cal_inject pulse does not affect the Wilkinson integration. This value is taken to be three clocks or 48ns and is set in firmware. It does not need to be downloaded to the Pulse Width Register prior to this test.

For each deadtime test, the channel number is specified along with the DTIME (**C9**) command. All string parameters must be set up by the software prior to the test. The 8-bit deadtime value, measured in units of 16 ns clocks, is returned for the requested channel.

Test	Command	Hex Code	Outbox
4.9	DTIM	C9	Ch0 DT

4.10. Nominal settings

The space of settings for the ASD is quite large and so each test has been done centered about a nominal point in this space. Nominal settings are listed below.

Parameter	Meaning	Nominal (Mini-DAQ) setting	Nominal hex value
Disc1_thr	Main discriminator threshold	- 44 mv	69
Hysteresis	Main discriminator hysteresis	2.5 mv	2
Wgate	Integration gate width	20 ns	6
Disc2_thr	Wilkinson disc threshold	30 mv	1
Wrund	Integrator rundown current	2 ua	5

5. Diagnostic routines

Diagnostics are used by expert users only and are intended as a tool set to help diagnose hardware problems in the Chip Tester itself.

5.4. Cal_test

This sends a continuous stream of Cal_inj pulses to the MDT-ASD. The separation of these pulses is 4096 ns or 256 clock periods. The pulse width, pulse height, and Mode (internal or external) must be set up prior to the test. Since the pulses are sent continuously for an indefinite period, the controller becomes "stuck" in this state. The controller must then be restored by toggling the *chip carrier power switch*, which functions also as a system **RESET** switch.

Test	Command	Hex Code	Outbox Data
-	Cal_test	D0	-

5.5. Time_stamp

This is a primitive diagnostic test which sends a single strobe to the Cal_inject circuits, operates the Time Stamp sub-circuit of the ASD Monitor, and returns both leading edge and trailing edge data to the Outbox. All relevant registers must be set up in advance. Note that this command is channel specific. Channel data is passed as the lower byte of the command as detailed in the Commands.xls . document.

Time stamp data are 10 bits; 2 bits of 2ns DLL interpolation plus 8 bits of course counter running at 62.5 MHz (16 ns period). The Time-stamp command will return two bytes of data for each of leading and trailing edge time as follows.

Test	Command	Hex Code	Outbox Data
-	Time_stamp	D1	Tle[7:0]
-		D2	Tle[9:8]
-		D3	Tte[7:0]
-		D4	Tte[9:8]

Note that it is up to the programmer to set up the Cal_inject and Time_stamp command to point to the same channel, or no useful data will be returned.

6. Quality Codes

Each chip will be assigned a composite quality rating consisting of three digits in the form

X-Y-Z

corresponding to three classes of tests as follows

- X : String test (Tests 2.x)
- Y : DC tests (Tests 3.x)
- Z : Dynamic tests (Tests 4.x)

In general, each field will have a value of 1 – 4 with 1 being the best and 4 being fatal. We expect to be able to define tolerances so that sufficient 1-1-1 chips will be available for the entire detector.

Values of each of these tests will be assigned using criteria described below.

6.6. String test codes (X)

It is possible that a chip will fail the full 53 bit string test but will be still quite functional. An example might be a failure of the LSB of the Deadtime register. For this reason, failure of the string test is not necessarily fatal. The following codes will be used.

Code	Meaning
1	No error bits
2	Single non-fatal error bit
3	Multiple non-fatal error bits
4	Any fatal error bit

A set of 11 registers will be maintained in the Data Base (DB) corresponding to the 11 sub-string registers in the MDT-ASD chip. Each bit of these registers will be assigned as follows;

Code	Meaning
0	Non-fatal error bits
1	Fatal error bits

Our experience is that bit errors will be extremely rare. For this reason, we will consider **any bit error to be fatal**. Thus, codes 2 and 3 will not be relevant. The only possibilities will be 1 (No error bits) or 4 (One or more error bits).

Thus, the masking scheme for string bits described above is to be regarded as overkill. It is already implemented however, and it need not be removed so long as any bit failure is now considered fatal. This information is entered into the database.

6.7. DC test codes (Y)

For each of the dc parameters measured, an expected value and three sets of tolerances will be maintained. Each of these sets will bracket the expected value. A chip which falls into the innermost brackets will get a Code 1 and so forth for each of the larger tolerance brackets.

Code	Description
1	All parameters within lowest tolerance
2	At least one parameter exceeds lower tolerance
3	At least one parameter exceeds mid tolerance
4	**Fatal.** At least one parameter exceeds upper tolerance

6.8. Dynamic test codes (Z)

The Dynamic coding scheme is identical to the DC scheme above.

6.9. Tray binning

Tested chips will be collected into multiple trays as follows.

Tray no.	Codes	
1	1.1.1	Highest quality
2	112 in any order	Single low level out-of-spec
3	122 or 222 in any order	Multiple low level out-of-spec
4	xx3 in any order (no 4s)	At least one high level problem
5	xx4 in any order	Trash

Preliminary testing indicates that ~70% of tested chips fall into Tray 1 and another 15% into Tray 2. The Tray 2 chips typically have larger chip-wide offset spread than those of Tray 1, but otherwise are quite usable.

7. Database requirements

Test results will be read from the Outbox in raw form. Raw data is expected to be of order 10k bytes or more, per chip. The Chip Tester on-line software will reduce these data to information which will be maintained in the Database as follows. In each case, the database will associate the data with the "Test Number" which generated it. Additionally, each Chip Tester will be bar-coded with a unique serial number which will also be recorded into the data base. In this way, slight variations in Chip Tester performance can be tracked.

7.1. String Tests

On chip string register (11 sub-registers) will be written to and read back sufficiently to determine whether a register problem exists. Only one such string exists for each chip. Any register problem is considered fatal.

Database entry (Test 2)

7.1.1. A set of error codes corresponding to each of the bits in each of the string registers. This point is moot, however. We consider any single bit failure to be fatal, and result in a string quality 4.

7.2. Version Number

This is a three digit number corresponding to ChipTester firmware revision. The format of the Version Number is as follows.

Database entry

7.2.1. Ver=X.XX

7.3. Bias test (DC)

A variable bias current is injected into the bias circuit node Vb2. The current monitored and returned into the Oubox as is the voltages at the Vb1, …., Vb4 nodes. The on-line software will fit these results and extract parameters which characterize transistors on the chip. These parameters are the so called "transconductance parameters" K_p, and K_n as well as two threshold voltages, V_{0p} & V_{0n} for the pfets and nfets respectively. A range of values will be considered acceptable. More important for acceptance is the ratio K_n/K_n.

Two of the voltages in the scan are expected to be equal at some value of current. This current, I_x, must be larger than the nominal current for proper operation and its value will be part of the acceptance criteria.

One set of values described above characterizes transistors in the entire chip.

Database entry (Test 3.0)

7.3.1. K_p , K_n , V_{0p} , V_{0n} : Single precision floating point values.

7.3.2. I_x : Single precision floating point value

7.4. Preamp inputs (DC)

Input voltages for both signal and dummy preamps will be monitored for each channel. From this, the *common mode* and *differential* voltages will be computed by on-line software and used as acceptance criteria. Differential voltage specification will be considerably stricter than the common mode spec.

Database entry (Test 3.1)

7.4.1. Vcm_in_i, ? Vin_i i=0,7 : 2 single precision floating point values for each of 8 channels (Total 16 floating point values).

7.5. LVDS outputs (DC)

Each channel's LVDS output driver will be forced into a Hi state and then a Lo state. In both cases, 8 common mode and 8 differential voltages will be measured. The common mode voltages should be independent of state (Hi, Lo) as should the absolute value of the differential voltage. These values will be used as acceptance criteria.

Database entry (Test 3.2)

7.5.1. Vcm_out_i, ? $Vout_i$ i=0,7 for each of Hi & Lo state: 4 single precision floating point values for each of 8 channels (Total 32 floating point values).

7.6. Power supply current (DC)

The power supply current for the whole chip will be monitored. Its value must be consistent with correct chip operation and will be used as an acceptance criterion.

Database entry (Test 3.3)

7.6.1. I_{total}: single precision floating point value.

7.7. Noise test (Dynamic)

Rates of thermal noise hits are recorded for a range of threshold voltage values. This is done over negative (normal operating mode) and positive thresholds. Noise hit rate vs. threshold voltage is Gaussian whose mean is the channel's *offset voltage*. A Gaussian fit is performed by on-line code and the *means*, *sigmas*, and *amplitudes (Rate)* are extracted for each channel. The *Rate* will be reported at *nominal threshold*. These values are used as acceptance criteria.

Database entry (Test 4.0)

7.7.1. $Voff_i$, s_i, $ln(Rate0_i)$ i=0,.. 7 : Total of 24 single precision floating point values.

7.8. Wilkinson tests.

For each of the parameters affecting the LVDS output pulse width in ADC mode, that parameter is swept over its range of values and the pulse width recorded. The data are fitted to quadratics whose parameters are placed in the database. In each case, the x coordinate is taken to be the hex code corresponding to the register associated with the test. Thus x takes on values;

x = 0,1, …..

7.8.1. Wilkinson gate (Dynamic)

Relationship of Wilkinson gate width to output pulse width is fit to a quadratic ($a \cdot x^2 + b \cdot x + c$) where x is the hex code corresponding to the Wilkinson Gate Register.

Database entry (Test 4.5)

7.8.2. a_i , b_i ,c_i i=0,.. 7 : Total of 24 single precision floating point values.

7.8.3. Wilkinson rundown (Dynamic)

Relationship of Wilkinson rundown current to output pulse width is fit to a quadratic ($a \cdot x^2 + b \cdot x + c$) function for nominal gate width a nominal input pulse height. This is done for each channel.

Database entry (Test 4.6)

7.8.4. a_i , b_i ,c_i i=0,.. 7 : Total of 24 single precision floating point values.

7.8.5. Wilkinson transfer function (Dynamic)

Result of a single Wilkinson scan is a somewhat non-linear, time vs amplitude characteristic. A parabolic fit ($a \cdot x^2 + b \cdot x + c$) will be done for each channel at nominal value of Wilkinson gate width and Rundown current.

Database entry (Test 4.8)

7.8.6. a_i , b_i ,c_i i=0,.. 7 : Total of 24 single precision floating point values.

7.9. Deadtime (Dynamic)

For each channel, the measured vs programmed deadtime is fit to a linear ($a \cdot x + b$)

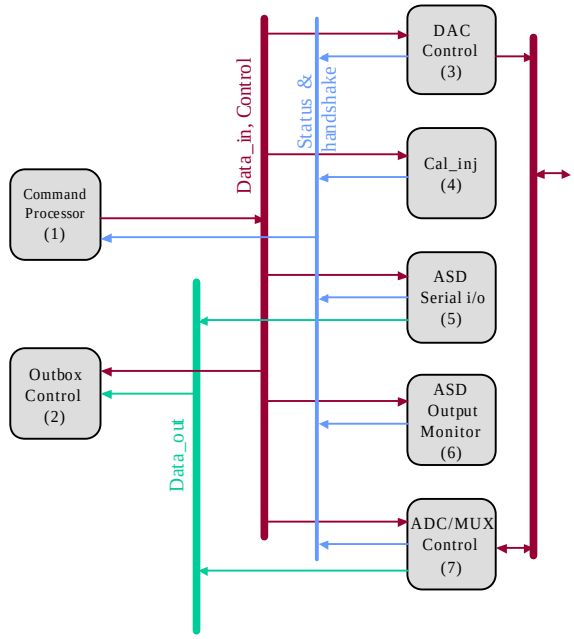
Database entry (Test 4.9)

7.9.1. a_i , b_i i=0,.. 7 : Total of 24 single precision floating point values.

8. Firmware

This description is kind of obsolete and only represents a very preliminary structure made at the outset of firmware development. I may update this at a later time…. or I may not.

8.1. General



8.1.1. /BUSY

Negative true signal from sub-sections back to the Command Processor. Indicates that a Command is in progress and that no other command should be processed. When un-driven, /BUSY is pulled up to a Hi level. Sub-sections require an active Select line to drive /BUSY.

8.1.2. /DTACK

Negative true signal from sub-sections back to CP. Indicates that output data have been placed on the DATA_OUT bus and can be stuffed into the Outbox fifo if necessary. When un-driven, /DTACK is pulled up to a Hi level. Sub-sections require an active Select line to drive /DTACK.

8.1.3. SS[2:8] (Sub_section Select)

These are select lines from the Command Processor to the sub-sections. When a sub-section is selected, it immediately drives the /BUSY and /DTACK lines to an appropriate state. These lines are released when SSx is removed.

8.1.4. XQT

This is a strobe line which initiates some action, when required, in a sub-section selected by the SS[2:8] lines.

8.1.5. Data_in (16)

Data path from Command processor to sub_sections.

8.1.6. Data_out (16)

Data path from sub_sections to wherever needed.

8.1.7. TEST[12:1]

This collection of lines is used to over-ride default values programmed into the PWR and DFSR. These lines are used only in high level tests for which specific values of these registers are required. The DUT and Controller registers are all returned to their original values on completion of all such tests.

8.1.8. Fifo control (8)

8.1.9. PCI handshake (3+clk=4)

Performs handshake operation with the PCI interface. A handshake is issues as long as the WRITE operation is allowed to take place. The only reason for this not to occur is if the Inbox fifo is full. In all other cases, the handshake is promptly completed.

8.2. Command Processor (CP)

The Command Processor is used to recognize and process incoming commands. The CP interprets commands and in turn, sends controls and data to the other sub-sections as needed. Any data coming in from the Inbox is embedded in the appropriate command: there are no data words per se in the Inbox, only commands.

Commands are loaded into the Inbox by the controlling pc. After a command is executed by the Control Processor, a new one is retrieved as long as the Inbox is “not empty”. Command results are place in the Outbox fifo.

On decoding a command from the Inbox, the Command Processor asserts the appropriate Sub-section Select lines SS[2:8]. The XQT line is controlled by the command processor and is strobed to initiate actions from the sub-sections when necessary. On receipt of the SS[2:8] lines, the selected Sub-section asserts /BUSY. Sub-sections signal the presence of new valid data on the output bus by asserting the /DTACK line which is then recognized by the Command Processor.

The details of this interaction are contained within the Command Processor State Machine (**CPSM**)

8.2.1. InReq State Machine

This is contained within the Command Processor and performs a simple handshake operation with the National Instruments PCI interface card. The card requests data transfer by asserting the ACK line. The InReq machine checks the Inbox Fifo, and if it is not full, strobes in the new command and responds to the interface card with a REQ.

8.3. Outbox

The Outbox Fifo stores data which results from particular commands (tests). All data which results from such tests is stored in this fashion; there is no communication with the PCI interface directly. The Outbox state machine handles the handshake with the National Instruments PCI interface card. The request for a Read operation is granted unless the Outbox is empty. In this case, the interface card times out.

It also packs the Outbox with data from the Command Processor (test results). The Command Processor presents the data and asserts a **GO** line. The Outbox state machine checks the Outbox fifo and, if not full, loads the data and responds to the Command Processor with **OK**. The CP releases **GO**, the Outbox SM releases **OK**, and everyone is happy. This function removes the need for detailed interaction with the Outbox fifo from the Command Processor.

8.4. DAC control

DAC control is used in both DC tests such as bias sweeps (test# 1.2.1) and in Dynamic tests such as Cal_inject (test# 1.3.2). Command processor puts data onto the DATA_IN bus and a DAC address onto the ADDR bus. It then hits LD_DAC. DAC sub-section asserts **/BUSY** for the duration of the operation. Since there are only two DACs, only the ADDR0 bit is relevant; all the rest are ignored. The **DAC** sub-section buffers the 12 bit data and sends it in Lo-Byte, Hi-Nibble format as required by the physical device.

The AD5343 spec calls for minimum 20ns control pulse widths. Thus the DAC control state machine will run at one half speed of the 62.5 MHz clock to guarantee 32ns minimum pulse widths.

8.5. Cal_inject

Used in both Internal and External calibrations. This section has two register locations as defined below.

ADDR	Name	No bits	Function	Definition
0	CALMOD	2	Mode	0x : internal 10 : external even 11 : external odd
1	PWR	8	Pulse width	1 to 256 clock cycles

Once registers are set up, each XQT will result in a Cal_inj strobe signal whose width is given by the contents of the Pulse Width Register (PWR). Pulse width is N+1 clock periods where N is the number programmed into PWR.

Contents of the PWR can be over-ridden by assertion of the TEST line into the Cal_inject sub-circuit. This is done, for example, in deadtime measurements where a non-standard pulse width setting is used. The setting is determined by an 8-bit code in firmware and is not programmable. On completion of such tests, the default value of the PWR is restored.

8.6. Subsection i/o

Data_in[7:0]

Used to load Pulse Width and Mode sub-registers

LD

Loads addressed sub-register

STR

Strobe output for internal (on-chip) cal_inject.

INJ_EVEN

Strobe output for external cal_inject (even channels)

INJ_ODD

Strobe output for external cal_inject (odd channels)

8.7. Serial i/o control

Used in primitive serial string tests and in all other tests requiring configuration of the 53 bit ASD serial string. Controller maintains default string in a 53 bit **Default String Register (DFSR)**. This register is composed of 11 smaller sub_registers appropriate to the MDT-ASD chip serial string. A second identical register, the **Dynamic String Register (DYSR)**, is used to modify the ASD string during tests. On completion of such tests, the default string is reloaded.

8.7.1. Subsection i/o

DATA_IN[7:0]

8 bit data bus to input **sub_register** contents to controller's **string registers (DFSR, DYSR)**

DATA_OUT[7:0]

8 bit tri-state bus which carries **sub_register** contents which have been uploaded from the chip.

TEST[11:1]

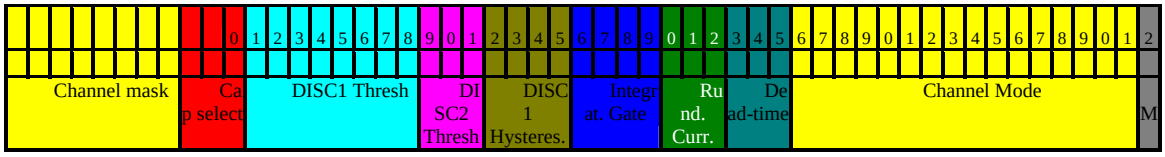
To be used during tests, this bus selects one or more of the sub_registers to be overridden by data placed in the **dynamic string register** and downloaded to the chip.

SR_ADDR[3:0]

Selects a **sub_register** of the **string registers**.

Reg No	Register	No. bits	7	6	5	4	3	2	1	0
1	Cal_inject mask	8	ch7	ch6	ch5	ch4	ch3	ch2	ch1	ch0
2	Cal_inj caps	3						msb		lsb
3	Disc1 threshold	8	msb							lsb
4	Disc2 threshold (Wilkinson)	3						msb		lsb
5	Disc1 hysteresis	4					msb			lsb
6	Wilkinson integration gate	4					msb			lsb
7	Wilkinson rundown current	3						msb		lsb
8	Deadtime	3						msb		lsb
9	Channel mode lower	8	msb							lsb
10	Channel mode upper	8	msb							lsb
11	Chip mode	1								lsb

Note to programmer: All DAC bit assignments will be inverted in firmware. *Programmer is to assign these bits in the normal manner*. This pertains to **sub_registers** 2, 3, 4, 5, 6, 7, 8



LD_DFSR

Loads **sub_register** of **Default String Register** pointed to by REG_ADDR with data on the DATA_IN bus.

RD_SS

Places the contents of the Sub-String of the Serial String (**SS**) pointed to by the **SR_ADDR** bus onto the tri-state DATA_OUT bus.

LD_DYSR

Loads **sub_register** of **Dynamic String Register** pointed to by **SR_ADDR** bus with data on the DATA_IN bus.

LD_CHIP

Downloads **serial string register** to ASD chip. String is taken from default string register (**DFSR**) unless over-ridden by the TEST lines.

RD_CHIP

Uploads serial string. Sub_string pointed to by SR_ADDR[3:0] is output onto the DATA_OUT bus.

MDT-ASD serial i/o lines

- SCLK
- SIN
- SDOWN
- SHIFT
- SLOAD
- SOUT

8.8. ASD output monitor

This performs time-stamping and counting operations used in dynamic tests. Operation is initiated by asserting select line SS8 and then hitting XQT. Operation is performed on the channel pointed to by the lower three bits on the ADDR bus. Resulting data are placed on the DATA_OUT bus and are accompanied by DTACK which is equivalent to “Data Valid”. If the operation requested returns leading and trailing edge timing, these are output sequentially along with two DTACKs. If no hit was detected during the interval, the returned data word or words will contain the value **zero**.

TDC leading and trailing edge data are 10 bit, with 2.0ns least count. Total time range is 0, .. , 2048ns. Output word is 12 bits as follows;

DATA11 = 0 leading edge, 1 trailing edge
DATA10 = 0
DATA[9:2] = course counter (8ns bins)
DATA[1:0]= interpolator (2ns bins)

The TDC is a sub-section contained within the ASD Output Monitor (ASD_MON). The TDC output consists of Leading Edge Time (**TLE[9:0]**), Trailing Edge Time (**TTE[9:0]**) and a READY flag. The READY flag is asserted on trailing edge data valid, or on 2048 ns timeout, whichever comes first.

8.9. ADC & Mux control

Used in DC tests of input and LVDS output levels. Also used in bias circuit sweeps (test #1.2.1). Two 12-bit ADCs (MAX1291) provide 8 input channels each. Four external dual 4:1 muxes are used to increase the number of available ADC channels. The six-bit ADC address space is configured as follows.

ADDR5 Selects ADC#2 / #1
ADDR4 Selects ASD chip IN/OUT (Mux address, ASD#2 only)
ADDR[3:1] Selects one of 8 ADC input channels
ADDR0 Selects MUX channel used on ADC#2 only

From programmer's point of view, the address map is as follows.

ADC#1

ADDR[5:4]	channel	ADDR[3:0]	Voltage
0x	0	000x	Vb_1
0x	1	001x	Vb_2
0x	2	010x	Vb_3
0x	3	011x	Vb4
0x	4	100x	VPDS
0x	5	101x	not used
0x	6	110x	not used
0x	7	111x	V_0

Note: x = don't care

ADC#2

ADDR5	ADDR4	ADDR[3:0]	Channel
1	0	0	ina0
1	0	1	inb0
1	0	2	ina1
1	0	3	inb1
1	0	4	ina2
1	0	5	inb2
1	0	6	ina3
1	0	7	inb3
1	0	8	ina4
1	0	9	inb4
1	0	A	ina5
1	0	B	inb5
1	0	C	ina6
1	0	D	inb6
1	0	E	ina7
1	0	F	inb7
1	1	0	outa0
1	1	1	outb0
1	1	2	outa1
1	1	3	outb1
1	1	4	outa2
1	1	5	outb2
1	1	6	outa3
1	1	7	outb3
1	1	8	outa4
1	1	9	outb4
1	1	A	outa5
1	1	B	outb5
1	1	C	outa6
1	1	D	outb6
1	1	E	outa7
1	1	F	outb7

NOTE to programmer: For programming simplicity, the firmware rearranges the original hardware address map, which was not in numerical order, to match the above normally ordered table. *In other words, don't even think about it.*

An ADC operation is initiated by placing the appropriate address on the ADDR bus, asserting the sub-section select line **SS8**, and hitting **XQT**. The DAC subsection asserts /BUSY during the digitization and on completion, places data on the DATA_OUT bus and asserts /DTACK to indicate "data ready".

8.10. Spare i/o

32 spare i/o lines used for logic analyzer ports and other nefarious purposes.

9. Firmware Versions Log

9.11. Ver0.89

31 Jan '03

Timestamp test is functional with resolution 4ns, although its accuracy has not been confirmed by histogramming fine time distributions. Deadtime had some problems which have now been fixed. The Deadtime binary search routine now works well.

Noise test was very messy until we discovered that the logic analyzer probes, which were spying on the asd signals, were feeding back to the chip. We removed them and things improved dramatically. We found that the test was marginally stable with six channels enabled, and completely unstable with eight channels enabled. We decided that four channels was too close for comfort and backed down to two. This works nicely, but of course dominates the test time, which is now several seconds.

9.12. Ver0.90

03 Feb '03

A general purpose delay element was added. The basic time interval is 1.024 us (64 x 16 ns) and is programmable to eight bits or 256us. The new command code is E0.

9.13. Ver0.91

25 Feb '03

The deadtime test still had some bugs in it. For one thing, the CPSM was not sending the ADDR bits to the DeadTime cell. Thus, the DeadTime cell was always interpreting the address as zero. This has now been fixed.

9.14. Ver0.92

27 Feb '03

There is occasional quirky behavior in the time stamp TDC which may well be due to excessive skew in the Din (input) signal. A MAXSKEW = 1ns attribute was added and achieved during Place and Route. No other functional change was made.

9.15. Ver0.93

1 Mar '03

The programmable delay function was not working correctly. The sense of the PDLY's BUSY line was up-side-down. This has been corrected and the delay now works as expected.

9.16. Ver1.00

4 Mar '03

This is just a new name so that Ver1.00 is the first production version. As far as we know, everything works.

10. Software Versions Log(C++)

10.17. Ver1.00

10 Mar '03

Initial release version of production code.

10.18. Ver1.01

14 Mar '03

Parameters for Deadtime, Wilkinson Transfer, Wilkinson Gate, and Wilkinson Rundown have been transformed and all now take the digital input code 0,1, .. ,Nmax as the independent variable.