ATS4000 Advanced Test System

GAP1616 Test Head Manual

Version C

Lynium

April 2, 2010
Tucson, Arizona

(This page intentionally left blank)

Document: GAP1616 Test Head
Version C
April 2, 2010
Lynium Document Number: TH4K15

Author:
Jerry Horn
Senior Design Engineer
Lynium LLC
2030 N Forbes Blvd, Suite 102
Tucson, AZ 85745
Phone: (520) 579-0047
Fax: (520) 579-8712
E-mail: jhorn@lynium.com

Company:
Lynium LLC
2030 N Forbes Blvd, Suite 102
Tucson, AZ 85745
Phone: (520) 579-0047
Fax: (520) 579-8712
Web site: www.lynium.com

© 2010 Lynium LLC

(This page intentionally left blank)

Abstract

The General Access Platform (GAP) 1616 test head is used in conjunction with the Advanced Resource Module (ARM) 6444 daughtercard (bottom), a device specific daughtercard (top), and the Advanced Test System (ATS) 4000 for testing mixed-signal products such as analog-to-digital converters (ADCs). This document describes the resources provided by the combination of the GAP1616, ARM6444, and ATS4000 and how these resources can be controlled and configured for testing ADCs.

(This page intentionally left blank)

Table of Contents

(This page intentionally left blank)

1. Introduction

The General Access Platform (GAP) 1616 test head is used in conjunction with the Advanced Resource Module (ARM) 6444 daughtercard, a device board, and the Advanced Test System (ATS) 4000 for testing mixed-signal products such as analog-to-digital converters (ADCs). Figure 1 provides a simple diagram of how the various pieces fit together.

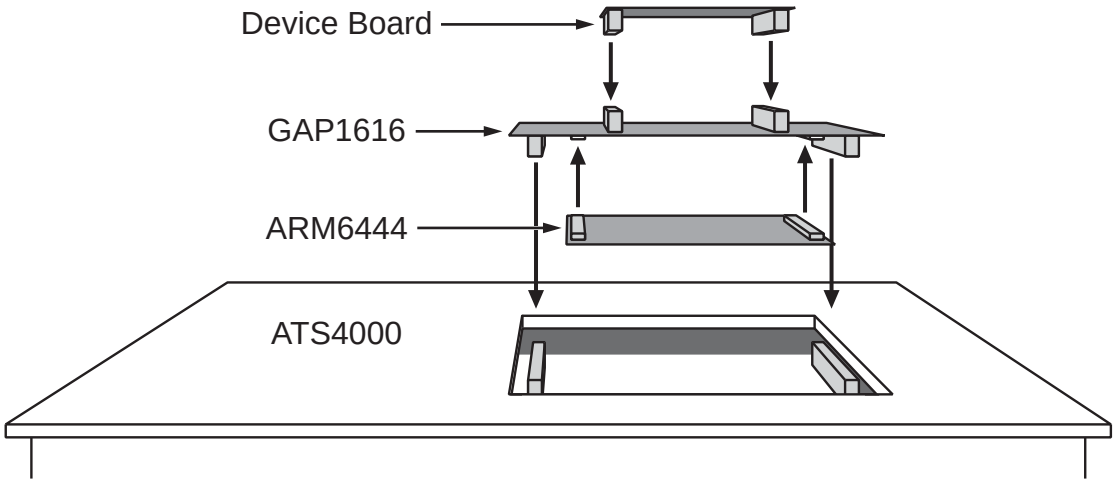

Figure 1. Interconnection of the Device Board, GAP1616, ARM6444, and ATS4000.

The ARM6444 daughtercard is mounted on the bottom of the GAP1616 test head and the combination is plugged into the ATS4000. Device boards, which are generally developed by the end-user, are plugged onto the top of the GAP1616. This document describes the platform that results from the combination of the ATS4000, GAP1616, and ARM6444. From a user's perspective, the critical functionality is that provided by the GAP1616 test head via the board's two 96-pin connectors. The remainder of this document will refer to the combination of the ATS4000, GAP1616, and ARM6444 as simply the GAP1616.

The GAP1616 provides various features which are designed to make testing industrial ADCs fast and simple. These features, along with the external GPIB equipment, provide a powerful test system for debugging, characterizing, and testing a variety of mixed-signal devices. For ADCs, tests include integral non-linearity (INL), differential non-linearity (DNL), offset error, gain error, signal-to-noise ratio (SNR), total harmonic distortion (THD), and spurious-free dynamic range (SFDR). (These tests are not discussed or described in this document. Contact Lynium or visit their web site at http://www.lynium.com for additional information.)

Along with the external equipment recommended in Section 2, the GAP1616 provides the ability to test an ADC with a resolution up to 16-bits and a conversion rate up to 50MHz. It should be possible to test higher speed converters or higher resolution converters, but the performance may not be optimum, particularly for those cases combining high resolution with a high conversion rate.

For cases where the GAP1616 is not an adequate solution, the end-user has the option of developing their own test head for use with the ARM6444. Another option is to have Lynium develop a custom test head (which may or may not use the ARM6444). If the custom solution uses the ARM6444, its functionality may be modified by Lynium as required to meet the needs of the ADC. In some cases, it may also make sense to have Lynium "stream-line" the software to provide a much simpler or more DUT-specific user interface. This can be useful when extensive in-house characterization is planned. A custom setup is generally more robust (particularly in regards to temperature cycling) and the software is useable by more individuals (such as lower level technicians).

The purpose of the GAP1616 is to allow the end-user to develop their own device boards as well as to get up and running quickly. However, its important to keep in mind that other options are available should they be needed. The document will focus solely on developing device boards for the GAP1616. Contact Lynium for more information regarding other options.

2. Equipment Required

The GAP1616 Test Head requires the following equipment:

- ATS4000 Test System
- ARM6444 Advanced Resource Module
- Function Generator with GPIB Interface (Agilent 33250A recommended)
- Digital Multi-meter with GPIB Interface (Agilent 3458A recommended)

The following equipment is also required for AC testing:

- Sine Generator with GPIB Interface (Agilent 33120A or Brüel & Kjær 1051)

The Agilent 33120A should be adequate for converters with a resolution of 10-bits and lower. At higher resolutions, the Brüel & Kjær 1051 is strongly recommended. For ADCs with conversion rates higher than a few megahertz, other generators may be required such as the Gigatronics 6080 or Agilent 8644B. Contact Lynium for specific recommendations.

See the ATS4000 Test System Manual for a complete list of generators and meters which are currently supported.

3. Calibration, Maintenance, and Troubleshooting

3.1. Calibration

The GAP1616 test head does not require calibration. Assuming that the performance of the device is not being limited by layout or other hardware issues, all results obtained using the GAP1616 test head are solely dependent on the external GPIB equipment, which should be properly calibrated at all times.

3.2. Maintenance

The GAP1616 test head requires very little maintenance. Over many insertions and removals, the device board connectors, the ARM6444 connectors, and/or the ATS4000 connectors may wear out. If this occurs, the recommended solution is to replace the entire GAP1616 test head (it is virtually impossible to replace a connector without compromising the reliability and performance of the board).

3.3. Troubleshooting

The RTB1616 daughtercard can be used to “loop-back” a number of GAP1616 digital signals and can be used to verify some analog operation as well. See the RTB1616 section in this manual for more information. Keep in mind that proper operation of the RTB1616 does not guarantee that the GAP1616 test setup is working correctly. However, it should be used as the first step in troubleshooting any test setup problems.

It is recommended that the user establish the average performance over many tests of several “golden units” of a fairly high performance ADC (such as a 12-bit or 16-bit device) as aids in verifying the proper operation of the GAP1616 test head, ATS4000 test system, ARM6444 daughtercard, and external GPIB equipment. At least one such unit should be tested when the GAP1616 test head is first installed, power is cycled, a significant period of time has elapsed since the last test was performed, or when there is any question about the performance of the test head and/or test system.

If the results are not as expected, remove the GAP1616 daughtercard, turn power off to the ATS4000 test system, remove the GAP1616 test head, and inspect it for damage as well as the ARM6444 daughtercard. If none is found, inspect the cabling between the external GPIB equipment and the ATS4000 test system. Confirm that each cable is in good condition and is properly connected at each end.

Note that the IEEE-488 Instruments section in the ATS4000 software’s main Setup panel shows where each piece of external equipment should be connected. A common problem is improper connection of the Master Clock (older test heads require a connection to the Clock input, the GAP1616 requires this connection at EXT 8).

If no problems are found or if they have been corrected, re-install the ARM6444 daughtercard and GAP1616 test head combination, turn power on to the ATS4000 test

system, and run a different golden unit, perhaps even one of a different device type and board. If this still has not corrected the problem, contact Lynium for more assistance.

4. ATS4000 Software DUT User Interface

Figures 1 and 2 show the ATS4000 software user interface that is generic to the GAP1616. The user enters the necessary information and configures the appropriate sections in order to develop a configuration for a specific ADC. For more information concerning other aspects of the ATS4000 software, see the ATS4000 Test System Manual. For more information concerning the ARM6444, consult the ARM6444 User's Manual.

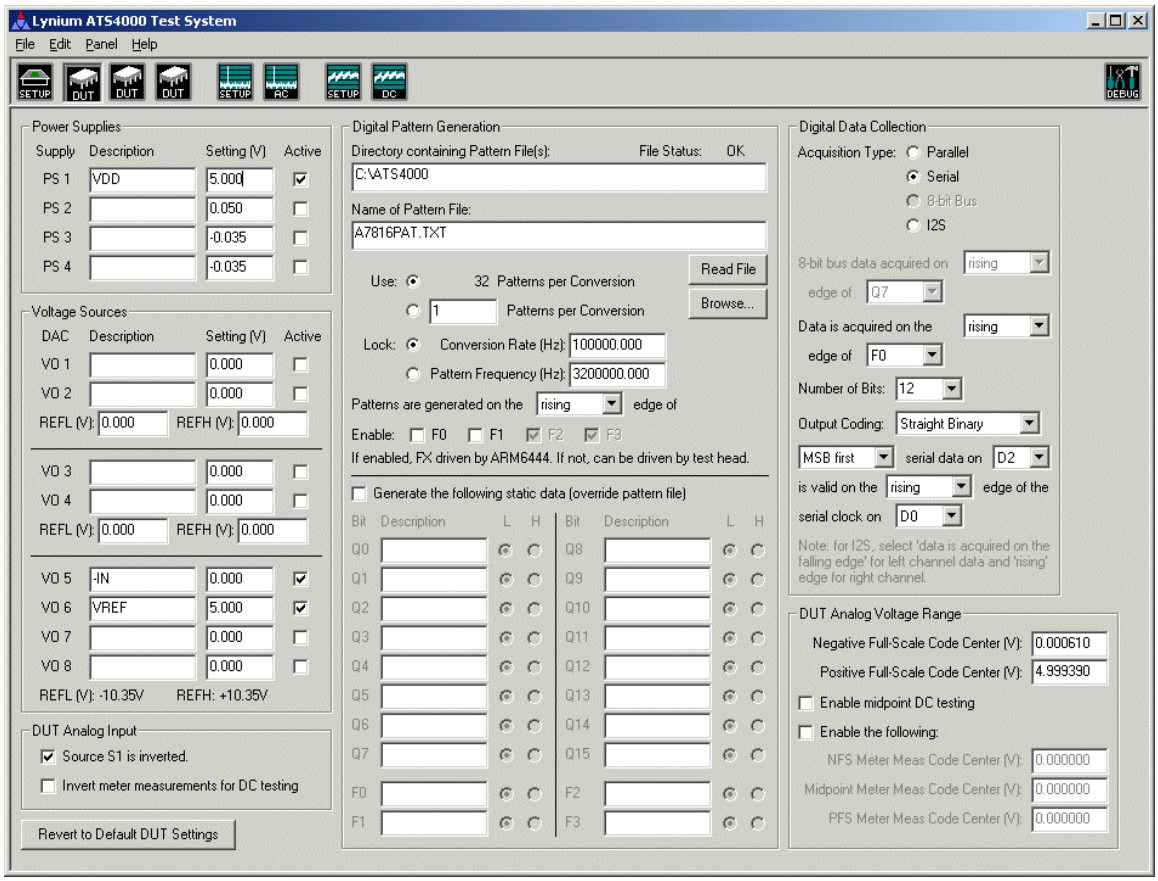

Figure 1. User Interface Panel 1 for the GAP1616.

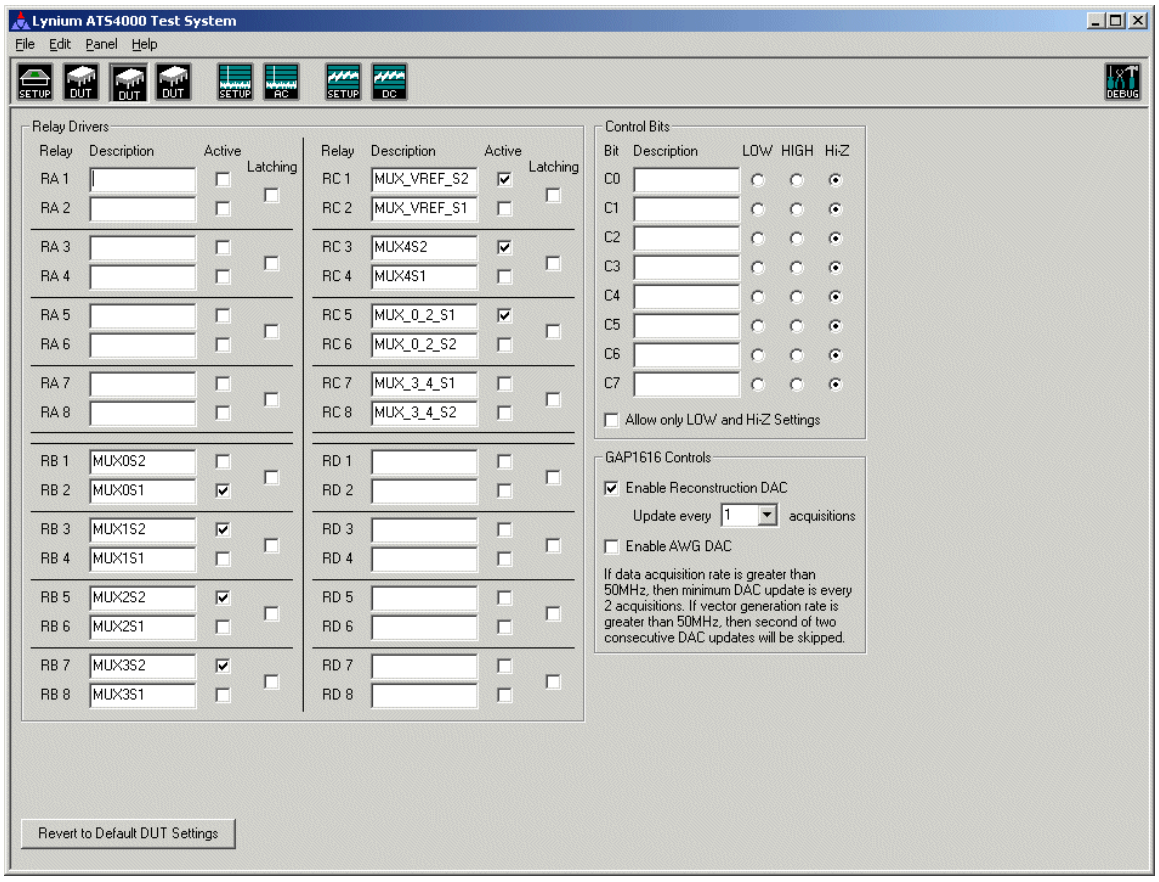

Figure 2. User Interface Panel 2 for the GAP1616.

Keep in mind that the DUT user interface is designed to control all of the resources offered by the GAP1616, ARM6444, and ATS4000. In most cases, only a small subset of those resources will be needed. Once everything is properly configured, the settings can be stored to a user specified file.

Figure 3 shows the user interface for the voltages sources PS 1 through PS 4 while Figure 4 shows the user interface for the voltage sources VO 1 through VO 8.

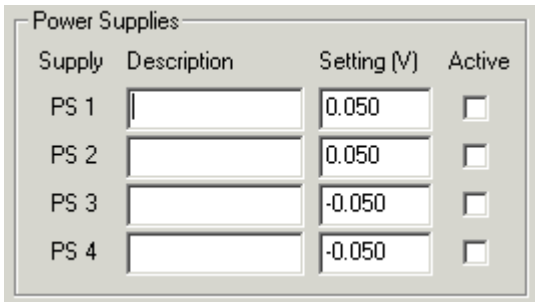

Figure 3. User Interface for the Voltage Sources PS 1 through PS 4.

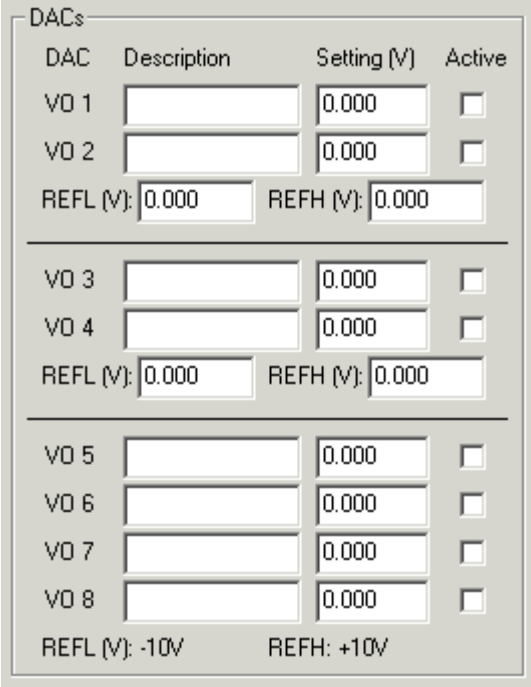

Figure 4. User Interface for the Voltage Sources VO 1 through VO 8.

Configuration of the voltage sources is done by entering the proper value (in volts) into the Setting entry box and "checking" the Active checkbox (clicking on the checkbox so that a check mark appears). Note that the value entered does not result in an immediate change of voltage at the test head. In fact, when the DUT interface is active, all voltages at the DUT socket, including digital signals, are at approximately 0 V. During testing, the active voltages will be configured to within 1 mV of the desired value.

Note that PS 1 through PS 4 have a "limited" disable capability and may float slightly when no testing is being performed or, during testing, if Active is not checked. On the other hand, VO 1 through VO 8 cannot be disabled. When no testing is being performed, these sources are set to 0 V ± 100 mV. If the sources are not Active when testing, the voltage will also be 0 V ± 100 mV. If it is important that a voltage source be within ±1 mV during testing, be sure to enter 0.000 V in the Setting field and check the Active checkbox. Simply removing the check from the Active checkbox will cause the source to be within of ±100 mV, it will not turn the source "off."

The description field is for reference and documentation purposes. It describes the function of each resource to the user. Some descriptions, such as those for voltage sources, may also appear when test results are displayed or data is saved to a file.

The turn-on sequence for the voltage sources is PS 1 through PS 4, then VO 1 through VO 8. The turn-off sequence is for VO 1 through VO 8 to be disabled simultaneously then PS 4 through PS 1 will be turned off in sequence.

The maximum source or sink capabilities for the PS voltage sources is 100 mA. The maximum source or sink current for the VO voltage sources is 1 mA. In general, it is

recommended that the VO voltage sources be lowpass filtered and buffered. However, for some ADCs, the inclusion of a 0.1 µF across the load may be adequate for keeping the output impedance of the voltage source low enough for good performance. The voltage source should be stable with this much capacitance and even higher (the maximum value is still to be determined). For very dynamic loads or for loads that draw more than 1 mA of current, a buffer is required. If the voltage source needs to be as low noise as possible, a lowpass filter prior to the buffer will help reduce noise.

Figure 5 shows the user interface for the relay drivers RA 1 through RD 8.

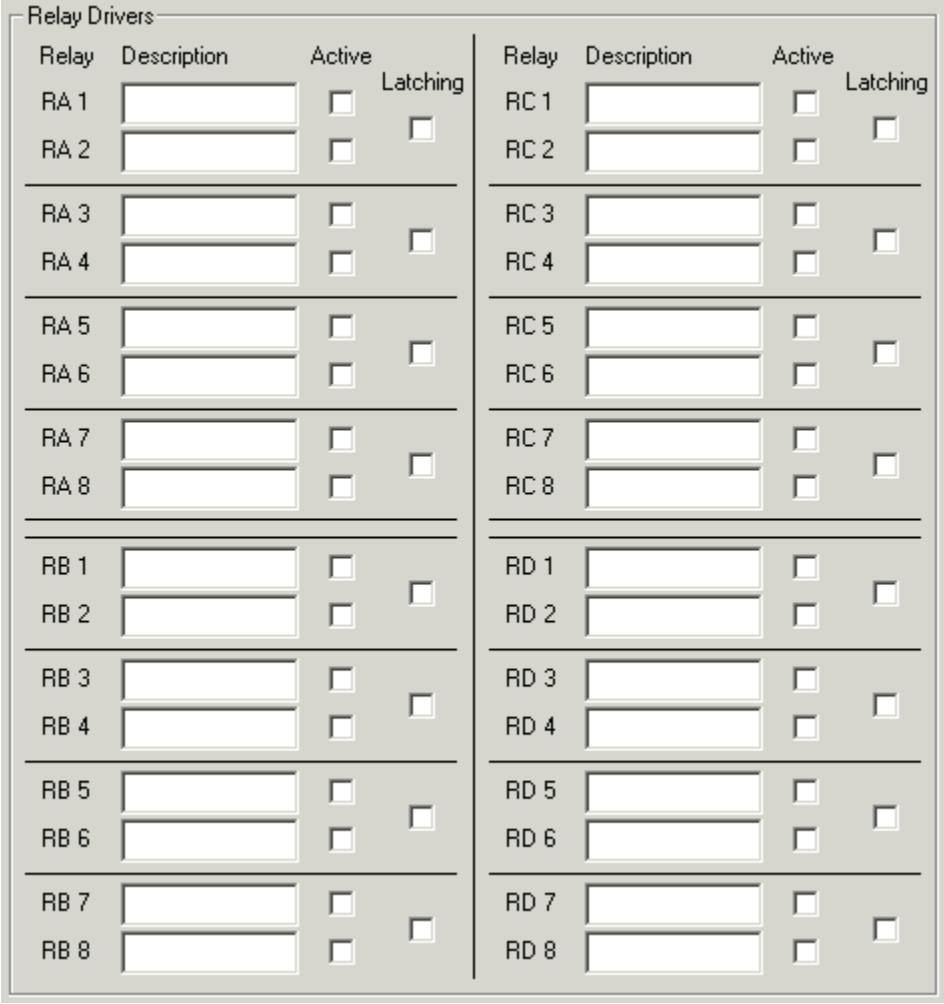

Figure 5. User Interface for the Relay Drivers RA 1 through RD 8.

The relay drivers RA 1 through RA 8 are the only drivers which are physically implemented on the GAP1616 test head. The other relay drivers may be implemented on the daughtercard by including one or more Allegro Microsystems UCN5895A or Micrel MIC5891 devices and connecting them to the RCLOCK and RLATCH pins. The first device should be connected to RDATA with the next device connected to the Serial Data Out of the first device (the serial data is daisy chained). The first relay driver in the chain

corresponds to RB 1 through RB 8, the next to RC 1 through RC 8, and the last to RD 1 through RD 8.

When the Active checkbox is checked, the associated relay driver will be HIGH during the test. If the Latching checkbox is also checked, the relay driver will be pulsed HIGH for 4ms at the start of the test. Also, if the latching checkbox is checked, only one of the relay drivers in that pair may be active.

The UCN5895A has a minimum output voltage of 3.8 V when supplying 120 mA of current. This is the integrated circuit on the GAP1616 that drives RA 1 through RA 8. The maximum power dissipation at 70°C is 1 W. The MIC5891 has a minimum output voltage of 3.1 V when supplying 200 mA of current. The maximum power dissipation at 70°C is 1.25 W. This means that these devices may drive only two or three relays simultaneously at the currents and voltages given. More relays can be driven if they require less current (see the respective datasheets for more information).

For latching relays, each relay is pulsed individually so there is no maximum number of relays. The sequence for enabling relays is RA through RD where all relays on RA will become active at the same time, then RB, etc. The sequence for latching relays is RA 1 through RD 8. The sequence for disabling relays is also RA through RD. The latching relays are not “disabled,” they simply remain in their latched state.

Figure 6 shows the user interface for the eight control bits. These bits are digital signals which are static (held at the value selected) during the test. Note that they are NOT level translated and that a HIGH results in a 5 V output level (no load). When tri-stated, the control bits are pulled HIGH with a 12k Ω resistor tied to 5V. The variation is this resistor’s value is ±5k Ω.

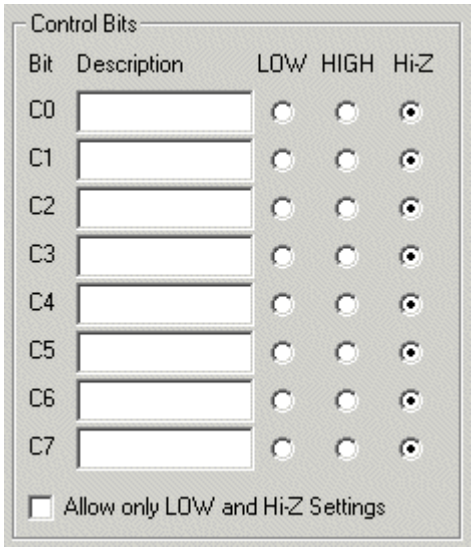

Figure 6. User Interface for the Relay Drivers RA 1 through RD 8.

4.1 Digital Pattern Generation

The file containing the data for the ARM6444's Digital Pattern Generator is specified in the Digital Pattern Generation section of the DUT User Interface panel (see Figure 7). The first text entry box in this section is used to specify the directory containing the pattern file (or files). The second entry box is used to enter the name of the file. The user can also "browse" to the directory and file by clicking on the "Browse" button.

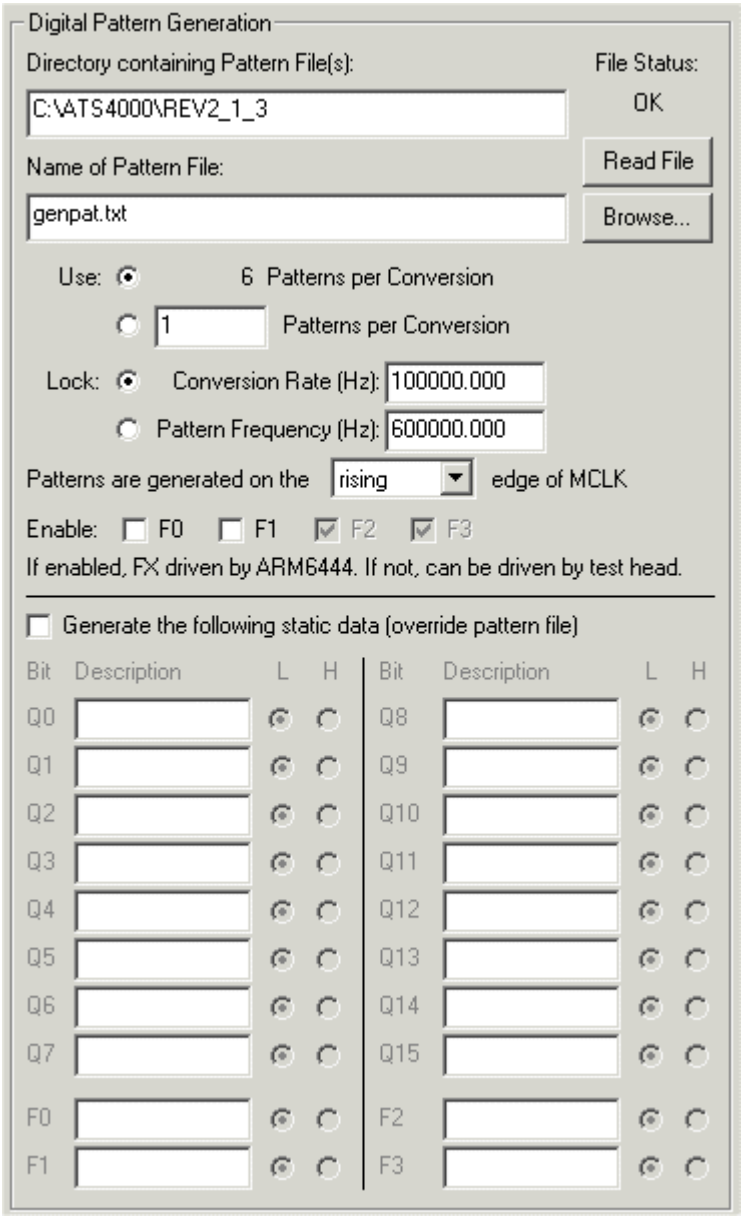

Figure 7. User Interface for the GAP1616 Digital Pattern Generator.

The pattern file is automatically read when the user selects a part number (under Specify Part) in the ATS4000 Setup panel, when there is a change to the directory, a change to the file name, or when "browsing" and OK is selected. However, if a change is made to the pattern file, the program is unaware of the change until the program reads the file. For

this reason, the user can force the program to read the file by clicking on the Read File button.

Also, if there is an error in the pattern file, the word “ERROR” will be shown under File Status. If the file is read during an automatic read, the program will not display any error messages, simply the fact that there is an error in the file. By clicking on the Read File button, the program will re-read the file and the error (or errors) will be displayed. The message or messages should describe the error and may suggest possible solutions.

The pattern file format is very basic. Table I shows a typical pattern file.

```
CONFIG 1.01
;
COLS   REPEAT Q15    Q14    Q13    Q12    Q11    Q10    F3
;
;                                  S
;                        N         A    I
;            C    N      P         D    N
;            L    C      O    G    O    G    D
;            K    S      R    O    N    L    V
;      REP  Q15  Q14  Q13  Q12  Q11  Q10   F3
BEGIN   1    0    0    0    0    0    0    0   ;
        1    0    1    0    0    0    0    0   ;NCS HIGH
; Loop Begin
LOOP    1    0    1    1    0    1    1    0   ;NPOR HIGH, CLK LOW #1
        1    1    1    1    0    1    1    0   ;CLK HIGH
        1    0    0    1    0    1    1    0   ;CLK LOW #2
        1    1    0    1    0    1    1    0   ;
        1    0    0    1    1    1    1    0   ;CLK LOW #3
        1    1    0    1    1    1    1    0   ;
        1    0    0    1    1    1    1    0   ;CLK LOW #4
        1    1    0    1    1    1    1    0   ;
        1    0    0    1    1    1    1    0   ;CLK LOW #5
        1    1    0    1    1    1    1    0   ;
        1    0    0    1    1    1    1    0   ;CLK LOW #6
        1    1    0    1    1    1    1    0   ;
        1    0    0    1    1    1    1    0   ;CLK LOW #7
        1    1    0    1    1    1    1    0   ;
        1    0    0    1    1    1    1    0   ;CLK LOW #8
        1    1    0    1    1    1    1    0   ;
        1    0    0    1    1    1    1    0   ;CLK LOW #9
        1    1    0    1    1    1    1    0   ;
        1    0    0    1    1    1    1    0   ;CLK LOW #10
        1    1    0    1    1    1    1    0   ;
        1    0    0    1    1    1    1    0   ;CLK LOW #11
        1    1    0    1    1    1    1    0   ;
        1    0    0    1    1    1    1    0   ;CLK LOW #12
        1    1    0    1    1    1    1    0   ;
        1    0    0    1    1    1    1    0   ;CLK LOW #13
        1    1    0    1    1    1    1    0   ;
        1    0    0    1    1    1    1    0   ;CLK LOW #14
        1    1    0    1    1    1    1    0   ;
        1    0    0    1    1    1    1    0   ;CLK LOW #15
        1    1    0    1    1    1    1    0   ;
        1    0    0    1    1    1    1    0   ;CLK LOW #16
END     1    1    0    1    1    1    1    1   ;acquire on F3 rising
```

Table I. Typical Pattern File.

A few notes about the pattern file. The file is an ASCII “text” file, generally with a .TXT extension. All lines or remainder of lines beginning with the following characters are considered comments: semicolon (;), hyphen ('), exclamation mark (!), asterisk (*), forward slash (/), or left bracket ([). Comments are ignored to the end of the line. The end-of-line character can be linefeed (ASCII code 10) or carriage return (ASCII code 13) or both. Keywords are not case sensitive.

Columns are tab separated, comma separated, or separated by spaces. All of these can be used within the same file. However, consecutive spaces are grouped as one space while consecutive tabs or commas are not. For example, X<tab>Y (where <tab> represents a single tab character), results in X being in column 1 and Y in column 2. The same is true for X<space><space><space>Y (where <space> represents a single space). On the other hand, X<tab><tab>Y results in X being in column 1 and Y in column 3. Mixing a tab or comma with spaces as a single column separator may produce unexpected results. In general, stick with one type of separator: tab, comma, or space(s).

Keywords always appear in column one. The COLUMNS (or COLS) keyword may be followed by additional keywords.

The file must always start with the CONFIG keyword, which describes the pattern file format and also tells the program which FPGA configuration to use for pattern generation (there are currently three configurations).

The COLUMNS (or COLS) keyword describes the order of the bits in the pattern. The default order is Q15 – Q0 and F5 – F0. This keyword is optional. If it used, it must appear before the BEGIN keyword. Column two must be blank or contain the keyword REPEAT. Columns 3 through 24 contain the keywords Q0 (or Q00) through Q15, F5, and F3 through F0 (F4 is currently reserved). A particular QX keyword establishes the proper column for that particular bit. It is acceptable to repeat a QX keyword, but this is not recommended. Only the right-most occurrence is valid.

An example COLUMNS entry is shown below:

```
COLUMNS    REPEAT    Q4      Q1      Q15     Q14     F0
BEGIN         1       0       0       1       1       0
              0       0       0       0       1       1
```

Table II. Example of a Pattern File Using the COLUMNS Keyword.

The flexibility in naming columns allows the user to group related bits together. It also allows a particular column (representing a digital signal) to be easily moved to a different bit. Thus, several different versions of the same signal may appear in the file, with unused versions being assigned to unused bits. The user can easily select the desired version by simply renaming the column(s).

The BEGIN keyword marks the start of the pattern data. All pattern data prior to BEGIN is ignored. Pattern data between BEGIN and LOOP are generated only once at the beginning of the test. The outputs of the pattern generator, Q15 through Q0 and F3

through F0 are all held LOW while the DUT is being configured for testing (power supplies brought up, input signals, etc.). This is also the case when the DUT is being disabled, at the end of the test. Note that when testing is complete, the pattern generator may go from any pattern to an all LOW condition and the final pattern may not be generated for the specified period of time.

The LOOP keyword indicates the re-entry point for pattern generation after the last pattern has been produced. There are no additional clock delays between the last pattern and the pattern that occurs on the same line as LOOP.

The END keyword marks the end of the pattern data. All file information appearing on the next line (or lines) after END is ignored. During testing, the pattern generator will loop continuously and inclusively over the patterns that appear between LOOP and END.

The smallest “legal” pattern file is shown below:

```
CONFIG     1.01
BEGIN        1     0
LOOP         1     1
END          1     0
```

Table III. Smallest “Legal” Pattern File.

This file will produce a square wave on Q15 whose frequency is ½ of the Pattern Frequency.

The repeat column is used to repeat a particular pattern the specified number of times. The repeat count can even be zero and the pattern will be ignored. However, there must be one valid pattern between BEGIN and LOOP and two valid patterns between LOOP and END.

For version 2.0.1 of the software or later, the number of patterns in the pattern file whose REPEAT count is greater than 0 cannot exceed 65,536. In addition, the maximum REPEAT count for a single pattern is 1,023.

This idea of a “number of patterns per conversion” is critical for setting up a proper pattern file and for controlling pattern generation. First, some definitions need to be established. There is a difference between the patterns specified in the pattern file and the “patterns per conversion.” Within the limits which will be discussed later, the software is not concerned with the number of patterns in the pattern file. For AC testing and for optimum DC testing, it does need to know the total “pattern count” which produces each new ADC result. The software assumes that this is the sum of the numbers entered into the REPEAT column between LOOP and END (including the patterns which appear with both the LOOP and END keywords), but this may not be the case.

For example, this pattern file:

```
CONFIG      1.01
COLUMNS    REPEAT    Q14     Q13
BEGIN        1        0       0
LOOP         3        0       1
             1        0       0
             3        1       1
END          1        1       0
```

Table IV. A Pattern File with a Default Eight Patterns Per Conversion.

contains five patterns, four of which appear between LOOP and END. The number of patterns per conversion will be calculated by the software as eight because two of the patterns appearing between LOOP and END are repeated three times each.

However, it's possible that the ADC may actually produce two conversions during the pattern loop. For example, one conversion might be done with Q14 LOW and the other with Q14 HIGH. The goal here might be to perform one conversion (initiated by Q13 going HIGH) on one channel and another conversion on a different channel, with Q14 being used to select the channel. For some reason, the user may wish to capture both conversion results (perhaps to compare the two channels). In this case, the user can override the software's automatic calculation by entering a value of 4 for Patterns per Conversion and selecting the adjacent radio button.

It's also possible (and more likely) that only every other conversion is needed (from one channel or the other), so the Patterns per Conversion is really eight (as the software automatically calculates it). Jumping ahead a little, the user is responsible for generating the proper signal in order to acquire the correct conversion of the two that are being produced (see section 5.2, Digital Data Collector, for more information).

The final item in the Pattern Generation section is the selection of Conversion Rate and/or Pattern Frequency. These two items are related by the following two equations (which are different expressions of the same equation):

Conversion Rate = Pattern Frequency / Patterns per Conversion
Pattern Frequency = Conversion Rate * Patterns per Conversion

When changing the number of Patterns per Conversion (and the adjacent radio button is selected), the software has the option of changing either Conversion Rate or Pattern Frequency, per the two equations. If the user "Locks" the conversion rate by selecting the radio button adjacent to the Conversion Rate entry box, then the Pattern Frequency is re-calculated when Patterns per Conversion is changed. Likewise, Conversion Rate is re-calculated if the pattern frequency is "locked."

This option is extremely useful should the Patterns per Conversion number change as different pattern files are selected for various test purposes. The user can choose to keep the pattern frequency constant or the conversion rate constant. In most cases, it is

desirable to keep the conversion rate constant, but there are cases where the pattern frequency (which might be associated with clock rate) should be held constant.

Finally, it is possible to accommodate different tests with a single pattern file by embedding two or more pattern files into one. This is done by commenting out the unused BEGIN, LOOP, and END lines and keeping only the desired ones. It is not necessary to comment out the unused patterns that appear without a keyword. An example follows:

```
CONFIG     1.01
COLUMNS    REPEAT    Q14     Q13      F0    ; F0 rising acquires the data
;
; Use this to test channel 0
;BEGIN       1       0       0       0
;LOOP        3       0       1       0
             1       0       0       1    ; Latch channel 0 data
             3       1       1       0
;END         1       1       0       0
;
; Use this to test channel 1
BEGIN        1       0       0       0
LOOP         3       0       1       0
             1       0       0       0
             3       1       1       0
END          1       1       0       1    ; Latch channel 1 data
```

Table V. Multiple Pattern Files in a Single Pattern File.

When this file is read by the ATS4000 software, only the patterns starting at the second BEGIN and ending with the second END will be used. The first set of patterns can be used by removing the ';' characters proceeding each of the three keywords. Note that it is not necessary to "comment out" the second set of keywords because the software will not read beyond the end of the line containing the first occurrence of END.

The F3 through F0 bits are used to control the F3 through F0 signals, respectively. The GAP1616 daughtercard interface routes the F0 and F1 signals to or from the daughtercard. The F2 and F3 signals are always driven by the pattern generator and received by the data collection module. Specifying F0 and F1 data in the pattern file does not result in the pattern generator automatically driving these signals. This option is selectable in the Pattern Generator section (see Figure 7). Note that the F3 through F0 signals are the only signals which can be used to acquire data (see Figure 10).

The pattern file bits F4 and F5 are special signals which do not result in any user observable signal. Currently, F4 is reserved for future use and this keyword should not be used. F5 is used to update the DAC associated with the DAC 1 (AWG) output on the GAP1616 test head. For example, the pattern file:


```
COLS  REPEAT Q15  Q14  Q13  Q12  Q11  Q10  Q9  Q8  Q7  Q6  Q5  Q4  Q3  Q2  Q1  Q0  F5
BEGIN 1      1    0    1    0    1    0
      1000   1    0    0    0    1    0
LOOP  4      1    0    0    0    0    0
      4      1    0    0    0    1    1
      3      1    0    0    0    1    0
      1      1    1    1    1    1    1   1   1   1   1   1   1   1   1   1   1   1 ;+2.5V
      3      1    0    0    0    1    1
      1      0    0    0    0    0    0   0   0   0   0   0   0   0   0   0   0   1 ;-2.5V
      4      1    0    0    0    1    0
      4      1    0    0    0    1    1
      4      1    0    0    0    1    0
      4      1    0    0    0    1    1
      4      1    0    0    0    1    0
END   4      1    0    0    0    1    1
```

Table VI. Pattern File Making Use of DAC 1 Output.

will result in a step function of the DAC 1 (AWG) output from positive full-scale (code $FFFF_H$ or +2.5 V) to minus full-scale (code 0000_H or –2.5 V). This high going pulse will last for a length of time equal to four patterns.

The use of F5 is somewhat experimental under the current software release and there are a number of items to be aware of in regards to using DAC 1. First, the DAC must be enabled, which is done through the GAP1616 Controls section of the DUT Panel 2 (see Figure 8).

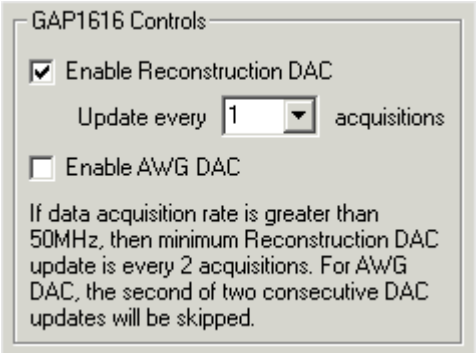

Figure 8. User Interface for the GAP1616 Controls.

Check the “Enable AWG DAC” checkbox in order to make DAC 1 active.

Another important item is that the DAC 1 output varies from –2.5 V to +2.5V and these correspond to the hexadecimal values 0000_H and $FFFF_H$, respectively. The software will generally keep the DAC 1 output at 0 V (code 8000H) when the DAC is not in use or when a test is not being run. If a different voltage range is desired, the DAC output will have to be level shifted and amplified by user supplied circuitry on the daughtercard.

Note that the DAC 1 output has no direct connection to the GAP1616 daughtercard. In order to get the signal to the daughtercard, a cable must be used to connect J1 to a connector on the daughtercard. This was done on purpose because, at this time and for the foreseeable future, the DAC 1 signal source will not perform a direct function in any specific ATS4000 ADC test. However, Lynium will be experimenting with various

possibilities and suggestions on how this DAC might be supported in the software are certainly encouraged.

The simplest way to experiment with the DAC 1 output is to connect it to an ADC input and then perform an AC test. While the results will not be valid for the AC results (unless the DAC output is a sine wave coherent to the ADC's conversion rate), the raw ADC results can be saved to a file and calculations on the data can be performed in Excel.

Finally, it should be noted that the data bits going to DAC 1 can be "out of order" if the pattern file has re-defined the pattern columns. This can be very confusing. For example, the pattern file:

```
COLS  REPEAT Q0  Q14  Q13  Q12  Q11  Q10  Q9  Q8  Q7  Q6  Q5  Q4  Q3  Q2  Q1  Q15 F5
BEGIN 1      1   0    1    0    1    0
      1000   1   0    0    0    1    0
LOOP  4      1   0    0    0    0    0
      4      1   0    0    0    1    1
      3      1   0    0    0    1    0
      1      1   0    0    0    0    0    0   0   0   0   0   0   0   0   0   0   1 ; ?V
      3      1   0    0    0    1    1
      1      0   0    0    0    0    0    0   0   0   0   0   0   0   0   0   0   1 ; ?V
      4      1   0    0    0    1    0
      4      1   0    0    0    1    1
      4      1   0    0    0    1    0
      4      1   0    0    0    1    1
      4      1   0    0    0    1    0
END   4      1   0    0    0    1    1
```

Table VII. Pattern File with "Out-of-order" DAC 1 Data.

might, at first, appear to step DAC 1 between minus full-scale (code 0000_H) and mid-scale (code 8000_H). However, the first bit is actually Q0, not Q15 (Q15 appears at the end). This actually steps the DAC between code 0000_H and 0001_H.

A value in the pattern file that is intended for DAC 1 may also include a change in F3, F2, F1, or F0, just as with a normal pattern value. This functionality is only present in version 2.1.2 of the ATS4000 software or later. Earlier versions of the software will not update F3, F2, F1, or F0 when the value is intended for DAC 1.

Note that two consecutive DAC 1 values are not allowed by the GAP1616 hardware, but the ATS4000 software will not indicate an error when reading such a pattern file. There are two ways around this problem. One solution is to make sure that at least one valid pattern occurs between every two DAC 1 updates. Another is to use a repeat value of 2 for any DAC 1 values that are followed immediately by another DAC 1 value.

4.2 Digital Data Collector

From the point-of-view of the GAP1616, two key functions are provided: pattern generation and data collection. These two items are completely separate and are "connected" only through proper use of F0, F1, F2, or F3.

The pattern generator runs at a frequency specified by the user through the Conversion Rate or Pattern Frequency entry boxes (see Figure 7). The signals contained within the patterns will cause an analog-to-digital conversion to be performed by the DUT. At some point during the repeating pattern loop (it does not matter where), data will be valid. As discussed in the previous section, data can also be valid multiple times during the loop or it may take multiple passes through the loop to produce a single data value.

The data collection circuitry requires a rising or falling edge which marks the point where data is valid. This must occur on one of the four signals mentioned above: F0, F1, F2, or F3. In most cases, it will be best if the DUT can generate the proper signal, so F0 or F1 would be used. However, it is also possible for the signal to originate directly from the pattern generator. Figure 9 shows the user interface for the data collection portion of the GAP1616 test head.

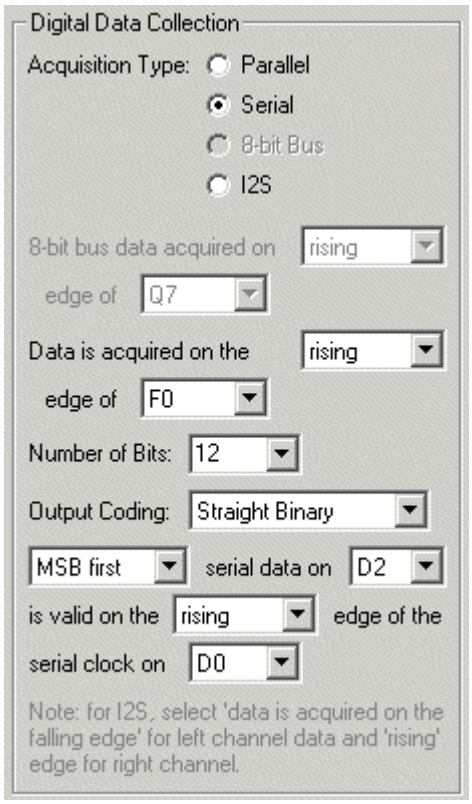

Figure 9. Digital Data Collector User Interface.

Data can be acquired on the rising or falling edge of F0, F1, F2, or F3. F0 and F1 can be sourced by the daughtercard or the pattern generator (user selectable, see Figure 7) while

F2 and F3 are sourced by the pattern generator only. Figure 10 provides a block diagram of the pattern generation, data collection, and DUT interfaces.

Note that if F2 or F3 is used, then data will be acquired even when no DUT is installed (of course, the data will be invalid, but the user may not notice this fact right away). This will also be true if F0 or F1 is selected and the signal is generated by the pattern generator. If the DUT supplies F0 or F1 or if the daughtercard circuitry generates F0 or F1 based on a signal from the DUT, then data will only be acquired when the DUT is installed and operating properly. If the DUT is not operating, the ATS4000 software will issue an error message informing the operator of this fact. This might be important when characterizing a large number of devices as the user may not be paying close attention to the results. So, in general, using F0 or F1 and basing this signal on a DUT output is the preferred method of acquiring data.

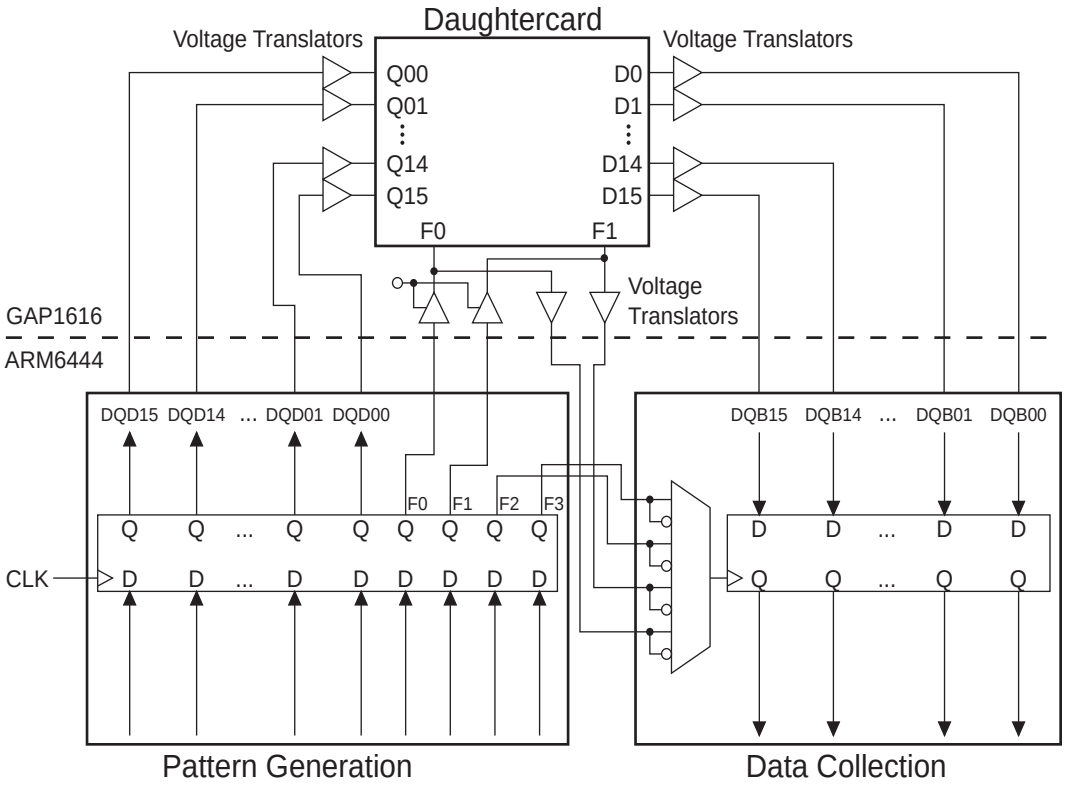

Figure 10. GAP1616 Pattern Generation, Data Collection, and DUT Interfaces.

The data setup and hold times relative to the various edges of F0 through F3 have not been determined. A setup time of 10ns and hold time of 10ns should be more than adequate. If more precise timing information is needed, contact Lynium.

For ADCs whose resolution is less than 16-bits, be sure to connect the MSB of the ADC to the D15 pin. Lower order bits should proceed from there (next lower bit to D14, etc.). Unused bits at the low end (D0, D1, etc.) can be left unconnected—there are 100kΩ pull-down resistors connected to these pins on the GAP1616 test head.

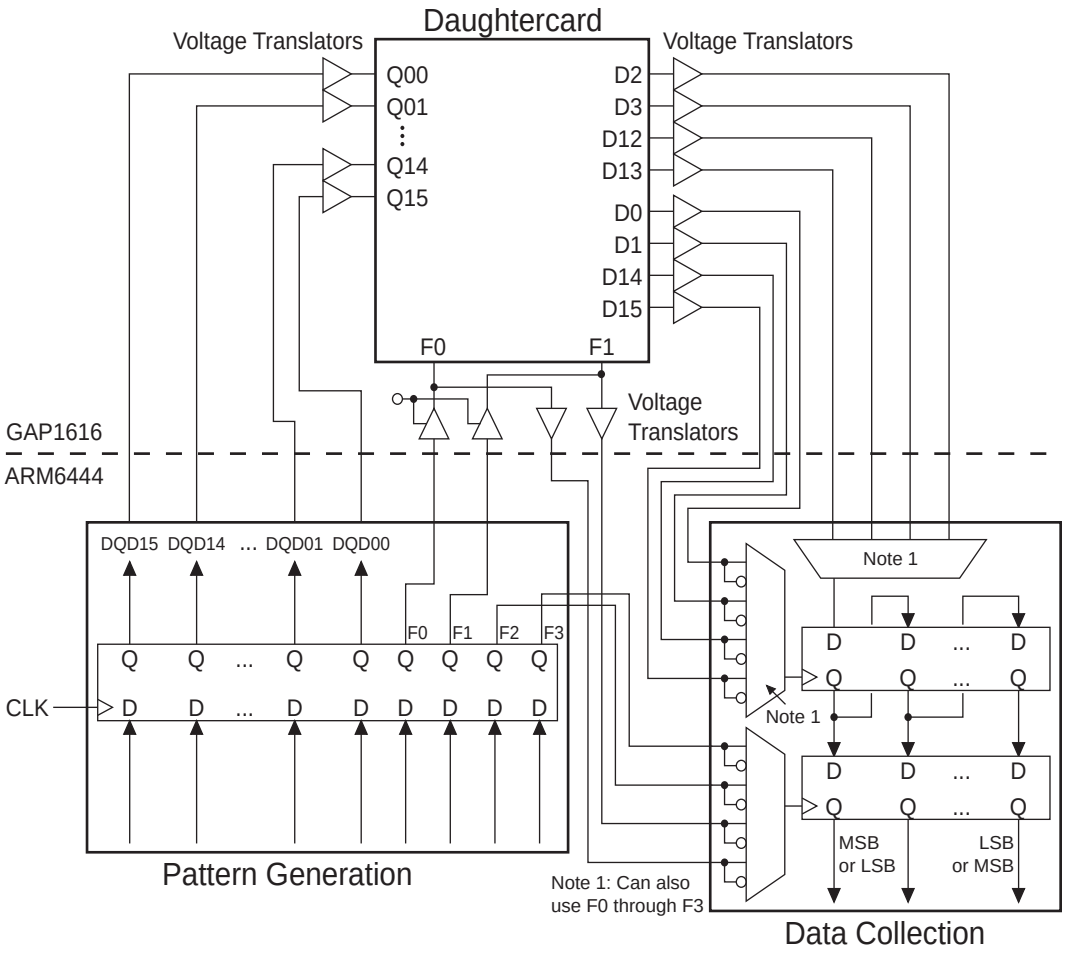

Figure 11. GAP1616 Pattern Generation and Serial Data Collection.

Serial data can also be acquired in addition to parallel data. A block diagram of this functionality is shown in Figure 11. The serial format that is expected is a “three-wire” interface consisting of a clock signal, data signal, and “frame sync” signal. The clock for the serial data must occur on one of the following signals: F0, F1, F2, F3, D0, D1, D14, or D15. The clock edge which latches the serial data is user selectable as either the rising or falling edge. The serial data is restricted to F0, F1, F2, F3, D2, D3, D12, or D13. The frame sync signal must appear on one of F0, F1, F2, or F3 and either the rising or falling edge can be used to latch the parallel data. The serial-to-parallel converter supports ADC resolutions from 6 to 16 bits and the first bit can either be the most significant bit or the least significant bit.

The setup time for each data bit relative to the serial data clock is 10 ns and the hold time is also 10 ns. The setup time for the second stage latch, the latch which follows the serial-to-parallel shift register, is 10 ns and the hold time is 10 ns. In other words, the frame sync signal should not transition within 10 ns of the critical edge of the serial clock (the edge which causes the serial-to-parallel shift register to shift). The shift register is a static design (no dependence on dynamic signals) and has no minimum clock frequency.

There is some flexibility here that may not be immediately apparent, but there are also some limitations. This format can be used for other serial formats, provided that two different sets of signals can be synchronous to each other. For example and I^2C format can be supported by generating signals that conform to the I^2C standard while having another set run the serial-to-parallel converter at the proper time.

On the other hand, those serial devices that generate their own shift clock and that do not provide a convenient signal transition at the end of the serial data may require a serial-to-parallel converter on the daughtercard. Another option might be a small amount of "glue logic" on the daughtercard that would generate the appropriate frame sync signal. For example, an audio ADC whose output data format is I^2S has a frame sync signal that transitions one serial clock cycle before the MSB is generated. By delaying the frame sync by some number of serial clock cycle so that it's transition is just after the LSB, this converter could be made to work with the serial-to-parallel converter included in the GAP1616 hardware. Such circuitry might require just a few flip-flops, or it could be considerably more complicated.

4.3 ADC Input

There is no “connection” in the ATS4000 software between a particular voltage or set of voltages and the input range of the ADC. In addition, newer versions of the ATS4000 software (version 3.8.0 and later) have both an ADC input range and an ADC measurement range. The proper way to think of this is that the ATS4000 software must know what range of voltages will cover the ADC’s input range. The system must also know what voltage will be measured over that range. In many cases, the two are the same while in others they are not. Readers new to the GAP1616 platform will probably find this confusing and these concepts will be developed and explained throughout this section.

4.3.1 Simple ADC Input

At the simplest level and when the input range and the measured range are the same, the user must supply the ideal input voltages for the ADC in the Negative Full Scale Code Center and Positive Full Scale Code Center text entry boxes (see Figure 11). These voltages must represent the ideal voltage that would produce the middle of the negative full-scale output code and the middle of the positive full-scale output code, respectively. For a converter whose digital output is straight binary, the negative full-scale code is 0 and the positive full-scale code is $2^N - 1$, where N is the resolution of the converter in bits. If the ATS4000 hardware is connected directly to the converter or through a simple gain of 1 voltage follower circuit or a gain of 1 inverter, then this is almost all the information that is needed to test the ADC. This section will focus on this concept first.

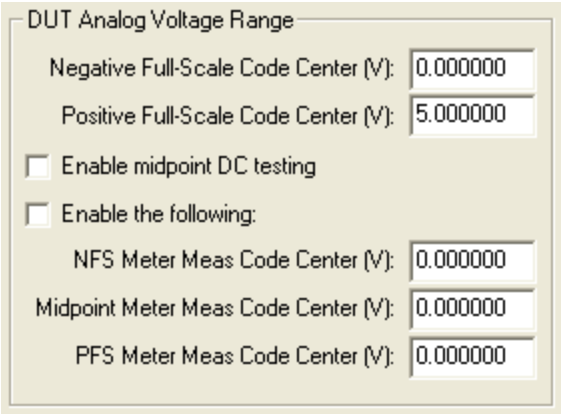

Figure 11. User Interface for ADC Input Voltage Range.

Because there is a limitation to how tightly the voltages will be set for PS 1 through PS 4 and VO 1 through VO 8 (generally ±1 mV), offset error and gain error, as reported in the DC test panel, may show a variance that is similar to the distribution of this error. The error may be as high as several millivolts if the input range is dependent on more than one voltage source. The actual distribution is dependent on both the repeatability of the test and the accuracy of the voltage source or sources. This issue only affects offset error and gain error, it does not affect differential non-linearity or integral non-linearity.

Note that the Number of Output Bits and Output Coding selections for the ADC are contained in the Digital Data Collection section of the user interface. The Number of Output Bits can be 6 to 16. It is possible to select a number lower than the converter's actual resolution, which can be useful in determining the performance of the ADC for a lower number of bits. The Output Coding can be either straight binary ($000\ldots000_B$ = negative full-scale, $111\ldots111_B$ = positive full-scale) or binary two's complement ($100\ldots000_B$ = negative full-scale, $011\ldots111_B$ = positive full-scale).

Source S1 provides one main function and one secondary function. The main function of S1 is to provide an input voltage to the ADC during DC testing. The range of this voltage is defined by the input range of the ADC as shown in Figure 11 and by the setting shown in Figure 12. For a simple ADC test, "Source S1 is inverted" and the top two text entry boxes of Figure 11 provide all of the information that is required regarding the ADC's input range.

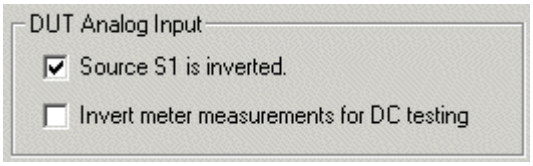

Figure 12. User Interface for S1 Polarity.

If source S1 is connected directly to the ADC or is connected through a buffer amplifier, then the top checkbox shown in Figure 12 should be clear. However, in some cases, it is better to invert S1, particularly in regards to its secondary function that will be discussed shortly. If it is inverted, then the top checkbox shown in Figure 12 should be checked.

At first glance, it might seem that it would be possible to simply change the input range of the part rather than tell the software that the input voltage was inverted. For example, the entries shown in Figure 11 might read 0 V for negative full-scale and –5 V for positive full-scale. However, when the user indicates that S1 is inverted, the software also knows to invert the servo-loop control signal. This is the critical functionality provided by the top interface option ("Source S1 is inverted") shown in Figure 12.

The secondary function of S1 is to provide an offset to the DUT for AC testing. Although some sine generators offer an offset function, most high performance generators do not. The ATS4000 software assumes the AC source does not have an offset function and that it will be provided via S1 or by another voltage that the user has chosen. In either case, S1 is always configured to provide the offset voltage per the converter's specified input range, even if S1 is not being used. Generally, the offset is one-half of the converter's input range. However, the AC setup panel allows the user to target a specific offset value.

When using S1 to provide the offset for the AC signal, it is usually most convenient to simply sum the AC signal and S1 via an inverting buffer. In this case, the functionality shown in Figure 12 allows the ATS4000 to be aware of the inversion. Note that during DC testing, the ATS4000 software turns off the AC source and there is no need for the user to provide a method of disconnecting the AC signal source.

It is possible to accomplish the offset of the AC signal source via other methods. Another possibility is to simply choose one of the VO sources and to set the offset voltage manually. The voltage source might then be summed in with the AC signal through an inverting amplifier. The disadvantage of this method is that the software will not automatically change the offset as the input range of the converter changes. On the other hand, the user may desire a direct method of controlling the offset voltage.

Note that S1 is always the input to the ADC for DC testing because this signal provides the servo-loop functionality and is the only source that does so. If S1 is inverted during DC testing, the checkbox shown in Figure 12 must be checked. If not inverted, the checkbox must be clear. S1 may also be used as an offset for the AC source. If this is the case but the polarity of S1 is different during AC testing than during DC testing, then the checkbox shown in Figure 12 will have to be set appropriately for each test. Obviously, the easiest course of action is to either invert S1 for both AC and DC testing or not. The most common setup is to simply combine the AC source (provided via EXT 1 or EXT 4) with S1 through an inverting buffer.

For the simplest testing scenario, the path from S1 to the ADC input should have a gain of +1 or –1. The S1 hardware and servo-loop circuit can make up some error if the gain is not accurate and the user has some control over this value in the DC SETUP panel ('Worst case maximum or minimum INL expected' text box). It should be possible for S1 to correct for errors up to ±100 mV from the ideal ADC input range (this includes both offset and gain errors).

4.3.2 More Complex ADC Input

As mentioned previously, newer revisions of the ATS4000 software allow more complex ADC configurations than earlier. New options were added that allow the user to specify both the input range of the ADC and the actual measurement range. The need for this may not be apparent at first, so this discussion will focus on why this change was made.

Newer ADCs often support true differential inputs. Unlike the "pseudo-differential" inputs of previous designs, these ADCs support a range of voltage (often from 0 V to V_{CC}) on both inputs. Pseudo-differential ADCs typically require the negative input to be held near ground. Since there are a wide variety of applications for ADCs embedded into microcontrollers, it can be useful to test a fully differential ADC as though it were a single-ended ADC, holding one of the two inputs at a static voltage near ground.

In addition, a differential ADC can support input voltages that are twice the size of the reference voltage. As an example, consider a 13-bit differential ADC with a reference voltage of 3.3 V. If the negative input is at +3.3 V and the positive input is at ground, then the difference in voltage is –3.3 V as seen by the ADC. If the positive input is at +3.3 V and the negative input is ground, then the difference is +3.3 V (this example ignores the fact that positive full-scale would probably be slightly less than +3.3 V in

order to simplify the example). Thus, the ADC has an input range between –3.3 V and +3.3 V, or a full-scale input range of +6.6 V.

The ATS4000 hardware was not designed to support differential ADCs. However, it is not very difficult to provide the circuitry on the GAP1616 daughtercard that will convert a single-ended input to differential. The problem with the older ATS4000 software is that the input range to the ADC had to be the same as the voltage measured on the input pins of the device. Thus, if the input range of the ADC is 0 V to +4.096 V, then older software would expect to provide this range of voltages and to measure the same range at the DUT input pins. There were a few tricks that could be used in order to get both the input voltage and the measured voltage to agree even for a differential ADC, but the end result was not very flexible.

The best solution is to provide the software with two sets of numbers—the input range of the ADC (or daughtercard circuitry) and the values that will be measured at the ADC input pins for the critical points in the converter's transfer function. In Figure 12, the top two numbers represent the voltage range that the ATS4000 hardware will provide. The bottom three numbers represent the ideal voltage that should be measured at negative full-scale, midscale (if enabled), and positive full-scale. The midscale number will be discussed later. For now, let us return to the example.

In the case of testing a differential ADC, there are various ways to convert a single-ended signal to differential. One of the easiest is and most flexible is to sum an offset voltage into two amplifiers while taking the main signal and adding it to the offset on one amplifier and inverting and then adding to the offset on the other. See Figure 13 for an example circuit.

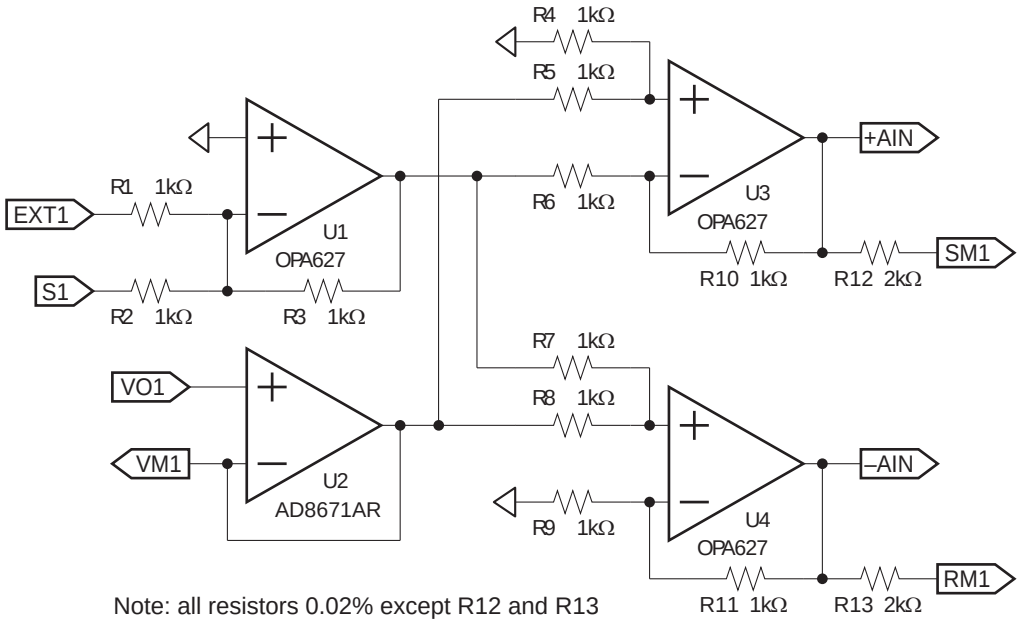

Figure 13. Example Single-ended to Differential Circuit.

In Figure 13, the VO 1 voltage source provides the common-mode voltage for both the positive and negative inputs of the differential ADC. The ATS4000 VO voltage sources have low output drive capability and fairly high output impedance (100 Ω). Thus, U2 provides a simple voltage-follower buffer for the VO 1 voltage source. The ATS4000 S1 output then drives the circuit over the full-scale range of the ADC. This circuit also allows S1 to provide an offset voltage for the EXT 1 input for AC testing (this is discussed in more detail in section 4.3.1 and that discussion applies here as well). The ATS4000 software assumes that S1 can provide an offset function for the AC input signal during AC testing as the S1 input is otherwise not used during AC testing (it is not required that the daughtercard provide this flexibility but it is a good idea for debugging and troubleshooting).

The circuit shown in Figure 13 has a number of features that makes it easy to configure the input voltages for the ADC. Both the EXT 1 and the S1 inputs are inverted and summed by U1. For the ADC's positive analog input (+AIN), U3 divides the output of U1 by two, inverts it, and then gains it up by two. Thus, S1 and EXT 1 end up with normal polarity. U3 also adds the common-mode voltage to this signal. For the negative input (–AIN), U4 divides the output of U1 by two and then gains it up by two while also adding the common-mode voltage.

This circuit allows the common-mode voltage to be entered into the VO 1 text entry box directly, without having to worry about inversion or gain. Likewise, the ADC's differential input range can be entered directly into the expected measurement text entry boxes without the added complexity of adding in the common-mode voltage. Finally, the negative full-scale and positive full-scale text entry boxes are used to enter the range of a single ADC input pin as though the ADC were a single-ended ADC. This is probably the most difficult item to remember but it is consistent with the function of the S1 analog signal (which is a single-ended signal).

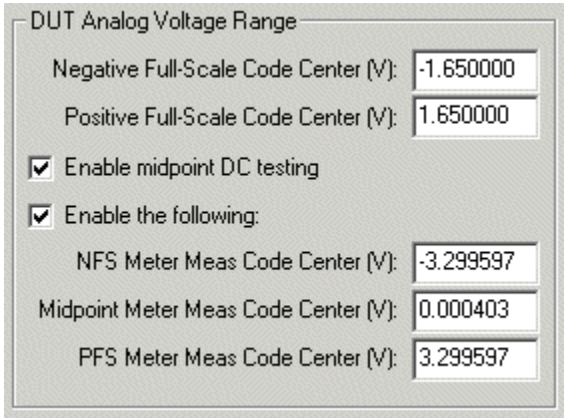

Figure 14. DUT Analog Voltage Range Configuration Example for a 13-bit Differential ADC with a 3.3 V Reference.

Figure 14 provides an example configuration for the hypothetical 13-bit differential ADC that has a 3.3 V reference. The VO 1 text entry box (not shown) is configured for 1.65 V. Note that the negative full-scale and positive full-scale code center entries do not have to

be very precise in this case. These voltages are used as rough estimates for the S1 analog voltage. They should be within a few millivolts of the correct value, but they do not need to be more precise than that. In this case, the bottom three text entry boxes correspond to the values measured at the ADC's endpoints and they must be precise.

Single-ended ADCs typically have an offset error instead of a negative full-scale error. The ATS4000 software uses the more precise terminology of negative full-scale error to indicate that offset was measured at negative full-scale. Differential ADCs typically measure offset at the midpoint of the converter's transfer function. Thus, differential ADCs have three critical points: a negative full-scale error, an offset error, and a positive full-scale error. The ATS4000 software now allows the user to indicate that the ADC has three critical points rather than two. This function was added in such a way that it could be used even with single-ended converters.

The circuit of Figure 13 allows a great deal of flexibility in testing a differential ADC and this flexibility may not be readily apparent. In the example that has been used so far, the common-mode voltage was defined as 1.65 V. However, a differential ADC does not care, within some reasonable range, what value of common-mode voltage is being used on both inputs. Ideally, the converter would work exactly the same regardless of the common-mode voltage. In reality, a change in the common-mode voltage can make a difference in the conversion results of the ADC.

If the 13-bit differential ADC is being tested with a supply voltage of 5 V but a reference voltage of 3.3 V, then the common-mode voltage range extends from 1.65 V (3.3 V divided by two) to 3.35 V (5 V minus 1.65 V). It may also be desirable to see what happens when the common-mode voltage is set lower or higher than this range, which drives some of the analog inputs slightly below ground or above the supply rail.

It is also very common to test a differential ADC as though it were a single-ended ADC. This can be done in two ways—using only half the converter's input range or using a smaller reference and holding one analog input at a static voltage. For example, it is possible to test the 13-bit ADC at a supply voltage of 5 V, a reference of 2.5 V, the negative input held constant at +2.5 V, and with an input range of 0 V to +5 V on the positive input (which appears to the ADC as an input range of –2.5 V to +2.5 V). Another option is to tie the negative input to ground and to simply use one-half of the converter's range on the positive input.

These types of experiments cannot be performed directly with the circuit shown in Figure 13. However, the inclusion of some additional jumpers and hardware can make this very easy. One addition is a simple static voltage. For example, the VO 5 voltage source could provide a static voltage. A jumper could then be used to switch –AIN from U4 to the buffered VO 5 source. In addition, a different jumper might allow –AIN to be tied directly to ground. With just a few jumpers and a couple of buffered voltage sources, a wide range of test configurations can be realized.

The ATS4000 GAP1616 software also makes these types of experiments easy to perform. However, there are some important considerations. In the case of the 13-bit ADC, the straight binary output code that occurs at negative full-scale is 0 while the straight binary output code that occurs at positive full-scale is 8,191. If only half of the converter's output range is being tested, then the ATS4000 needs to see a 12-bit result, not a 13-bit result. For an embedded ADC, this can be done by limiting the output codes to only 12-bits. When testing on the positive channel, the range of output codes should be limited in software between 4,096 and 8,191 and these codes should be mapped to a 12-bit range of codes between 0 and 4,095.

In the case of doing the same type of test on the negative channel (testing the lower half of the converter's input range), things get even more complicated. In this case, it may be desirable to "reverse" the input range of the ADC. Rather than go from negative full-scale (which is the maximum voltage on the negative channel) to midscale (which would be 0 V on the negative channel assuming the positive channel was set to 0 V), it might be desirable to test the converter as though it were a "normal" converter, with the input going from 0 V to the maximum voltage.

This situation causes the ATS4000 software some difficulty in regards to the circuit of Figure 13. The reason that a problem exists is because 0 V represents the minimum input voltage to the ADC but the maximum voltage is lower (as seen by the digital multimeter looking at the ADC's input) and is the negative value of the ADC's reference voltage (assuming the differential ADC has an input range of twice the reference voltage). The ATS4000 software was written in such a way that higher output codes must always correspond to higher measured voltages, so simply entering a more negative value for positive full-scale than for negative full-scale will not work.

One solution is to allow a swap of the SM1 and RM1 signals in the daughtercard circuitry and that's a perfectly acceptable solution that could be implemented with jumpers. Another solution is to enable (place a check) in the "Invert meter measurements for DC testing" checkbox shown in Figure 12. Typically, the checkbox would have to be checked and the state of the "Source S1 is inverted" changed to the opposite configuration. As in the case of testing the positive input channel, the microcontroller's software will have to limit the ADC's output codes to the range expected by the ATS4000 software.

Note that in the case of a discrete 13-bit differential ADC, testing the converter as though it were a lower resolution device requires additional circuitry to move the lower bits up and to limit their range. The digital circuitry is typically not difficult and can be implemented in discreet logic or a CPLD.

The topics covered in this section are very advanced. If you are new to the ATS4000 GAP1616 platform or have just started testing differential ADCs, then the discussion may have been difficult to follow. If so, the best option is to review the tests that you expect to be performing on the differential ADC and to start a preliminary schematic. Once you have some of the basics covered, Lynium will be glad to review the material, answer questions, and offer guidance on the best way to accomplish your goals.

5. Daughtercard Interface

Figure 15 provides the pin-out for the daughtercard interface on the GAP1616.

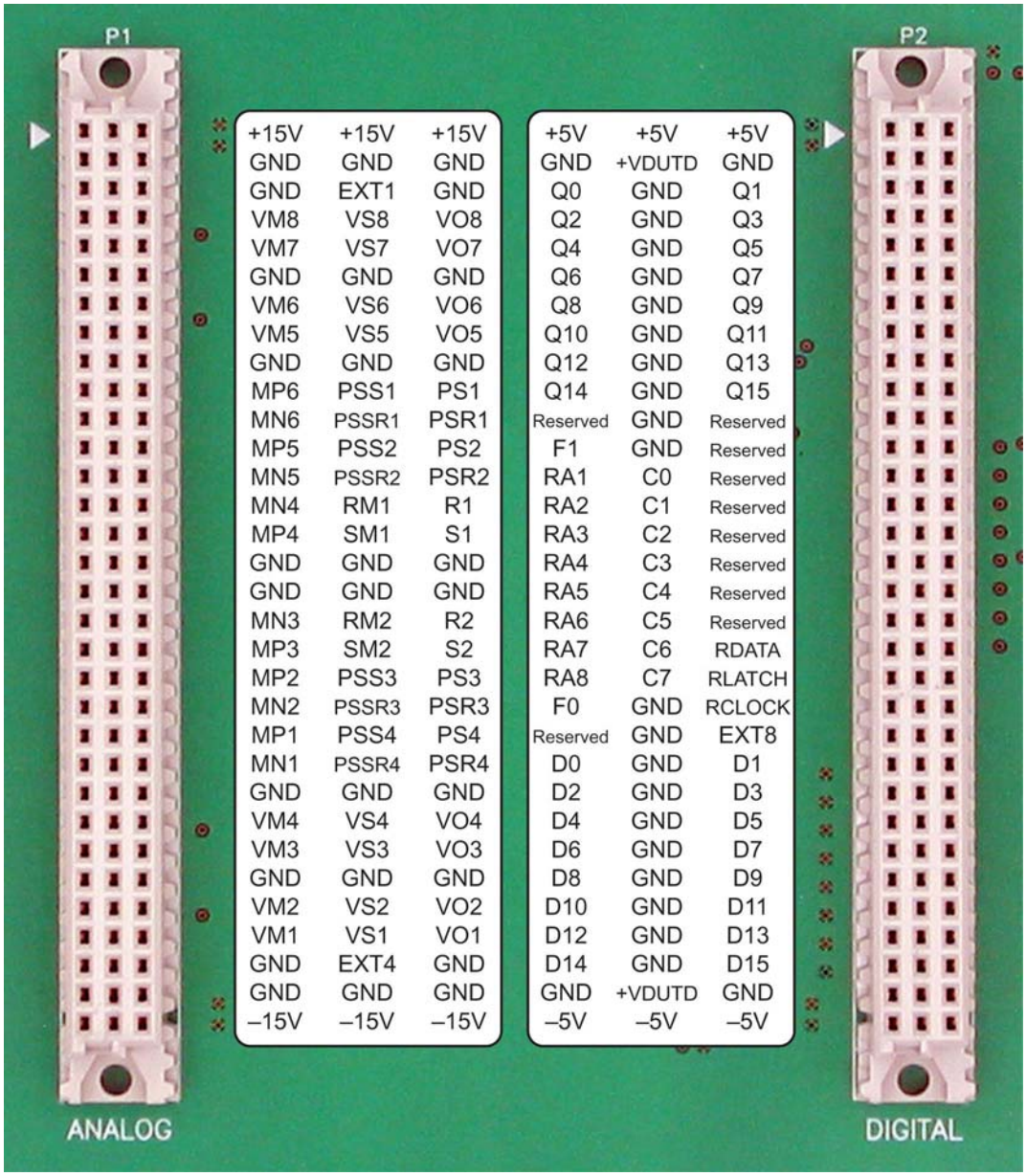

Figure 15. GAP1616 Daughtercard Interface.

The left connector (P1) provides the analog interface between the GAP1616 and the daughtercard. The leftmost pin labels are for this interface. The right connector (P2) provides the digital interface with the rightmost pin labels providing the pin-out.

The analog interface provides numerous ground connections (marked “GND”) and two power supplies: +15V and –15V. It is recommended that the daughtercard should make use of as many ground connections as possible (preferably all of them).

Likewise, the digital connector provides a number of ground connections (marked "GND") and two power supplies: +5V and –5V. The daughtercard should make use of as many ground connections as possible, particularly those grounds near high-speed (>10MHz) digital signals.

Table VIII provides a detailed description of each pin on the analog connector (P1) while Table IX is for the digital connector (P2).

+15	+15 V Power, maximum current source: 500mA.
EXT1	Direct connection to EXT1 input on rear of ATS4000. Normally used for AC signal.
VM 5 – VM 8, VS 5 – VS 8, VO 5 – VO 8	Voltage sources 5 through 8, range of –10.25V to +10.25V, 1mA maximum source and sink current, VO is the output, VS the sense pin (connect to VO at the daughtercard), VM is the measurement channel (connect to VO or, if VO is unity-gain buffered, connect to buffer output).
VM 1 – VM 4, VS 1 – VS 4, VO 1 – VO 4	Voltage sources 1 through 4, programmable range, 1mA maximum source and sink current, VO is the output, VS the sense pin (connect to VO at the daughtercard), VM is the measurement channel (connect to VO or, if VO is unity-gain buffered, connect to buffer output). Note that the range for sources 1 and 2 is independent of the range for sources 3 and 4. Maximum range is –10.25V to +10.25V.
MP 1 – MP 6, MN 1 – MN 6	Additional differential measurement channels. Current software does not support these measurement channels.
PS1, PSS1, PSR1, PSSR1, PS2, PSS2, PSR2, PSSR2	Power supplies 1 and 2, range of +0.035V to +19.75V, 100mA maximum source and sink current, PS is the supply voltage, PSS is the voltage sense pin (connect to PS at the daughtercard load), PSR is the supply return (connect to ground at the daughtercard; note: already connected to ground on the GAP1616 board), PSSR is the return sense (connect to ground near daughtercard load ground point).
PS3, PSS3, PSR3, PSSR3, PS4, PSS4, PSR4, PSSR4	Power supplies 3 and 4, range of –19.75V to –0.035V, 100mA maximum source and sink current, PS is the supply voltage, PSS is the voltage sense pin (connect to PS at the daughtercard load), PSR is the supply return (connect to ground at the daughtercard; note: already connected to ground on the GAP1616 board), PSSR is the return sense (connect to ground near daughtercard load ground point).
S1, R1	Source channel 1, S1 is the voltage output, range is –10.25V to +10.25V, 10mA maximum source and sink current, R1 is the return path (connect to ground at the daughtercard). This is the main voltage source for DC measurements and servo-loop functions. It can also serve to offset the AC input when summed with an external AC signal. This source is single-ended and must be converted to differential for those ADCs with true differential inputs.
SM 1, RM 1	Differential measurement channel 1. This is the main differential measurement channel for DC measurements. Connect SM 1 to the positive input of the converter and RM 1 to the negative input (or ground if no negative input).
S2, R2	Source channel 2, S2 is the voltage output, range is –10.25V to +10.25V, 10mA maximum source and sink current, R2 is the return path (connect to ground at the daughtercard). Current software does not support this source.
SM 2, RM 2	Differential measurement channel 2. This is an alternate differential measurement channel. Current software does not support this measurement channel.
EXT4	Direct connection to EXT4 input on rear of ATS4000.
–15	–15 V Power, maximum current sink: 500mA.

Table VIII. Pin Descriptions for the GAP1616's Analog Connector (P1).

+5	+5 V Power, maximum current source: 1A.
+VDUTD	Connect either one of these pins to the voltage source that sets the DUT's I/O voltage levels. On the GAP1616, this voltage sets the HIGH input/output level of the CMOS voltage translators for pins D0-D15, Q0-Q15, F0, and F1. Minimum voltage on +VDUTD is TBD (should be 2.5V or less), maximum is 7V.
Q0 – Q15	Level translated digital signals driven by GAP1616 and used for arbitrary digital "pattern" generation. Pattern data is defined by user in user specified text file. Maximum source/sink current is 5mA.
Reserved	These pins are reserved for future use. Do not make any connection to these pins.
F0, F1	These pins may be sourced by the GAP1616 or the daughtercard (user configurable). If sourced by the GAP1616, they can be used as additional digital signals in a similar manner to Q0-Q15. Regardless of the source, they are primarily intended to drive the data collection side of the GAP1616 (see Figure 10). They can also be used to indicate when to acquire data or for serial-to-parallel conversion (see Figure 11). If the acquire signal originates from the DUT, it must be on F0 or F1. Maximum source/sink current is 5mA.
RA 1 – RA 8	Relay drivers 1 through 8. These sources can control continuous relays or momentary relays such as latching relays. Minimum output voltage is 3.8V when supplying 120mA of current. Maximum power dissipation for the relay driver itself is 1W at 75°C (no more than two outputs should be driving 120mA at any one time). See the MIC5891 data sheet for more information.
C0 – C7	Static control signals 0 through 7. These digital signals are static during any test and are NOT level translated (HIGH = 5V). When tri-stated, they are pulled HIGH (to 5V) with 15kΩ (range is 8kΩ to 20kΩ). Maximum source/sink current is 5mA.
RDATA, RLATCH, RCLOCK	These signals are designed to control additional UCN5895A or MIC5891 relay drivers which are placed on the daughtercard (RA1-RA8 are sourced by a UCN5895A on the GAP1616). Up to three additional drivers can be used. The Serial Data In of the first should be tied to RDATA and the Serial Data Out tied to the Serial Data In of the next (daisy-chained). The first driver corresponds to RB1-RB8 in the ATS4000 software. The second and third to RC1-RC8 and RD1-RD8, respectively. RLATCH should be tied to STROBE on all the devices and RCLOCK to CLOCK.
EXT8	Direct connection to EXT8 input on rear of ATS4000. If the master clock is not a sine source (see the main SETUP panel of the ATS4000 software), then the signal on EXT8 is the master clock for pattern generation (shown as CLK in Figures 10 and 11). This signal may also be used as a clock source on the daughtercard. However, the voltage of EXT8 is not level translated. The LOW value is approximately 0.0 V and the HIGH level approximately 4.0 V.
D0 – D15	These signals are driven by the daughtercard and should, at some point, represent the result of an analog-to-digital conversion. If the data is parallel, D15 should be the most significant bit (MSB). For serial data, D0, D1, D14, or D15 can be used for the serial clock while D2, D3, D12, or D13 can be used for the serial data (F0–F3 must be the serial latch or frame sync signal). Note that the serial clock or serial data can also be on one of F0, F1, F2, or F3. These signals are level translated.
–5	–5 V Power, maximum current sink: 1A.

Table IX. Pin Descriptions for the GAP1616's Digital Connector (P2).

6. The RTB1616 Self Test Daughtercard

The RTB1616 is a Lynium supplied GAP1616 daughtercard that can be used to run a limited self test on the GAP1616 setup. To use this board, install the daughtercard, turn power on to the ATS4000, turn on all external equipment, and start the ATS4000 software. Select the File menu and then Open and navigate to the folder containing the most recent ATS4000 software (for example: C:\ATS4000\REV2_1_2). In the file list, select RTB1616.P4K and click OK. It will take a few moments for the software to load the pattern file.

When this is done, go to the AC TEST panel of the ATS4000 software and click Run. If the GAP1616, ARM6444, ATS4000, and external equipment are working properly, the AC test should be successful and provide the result shown in Figure 16.

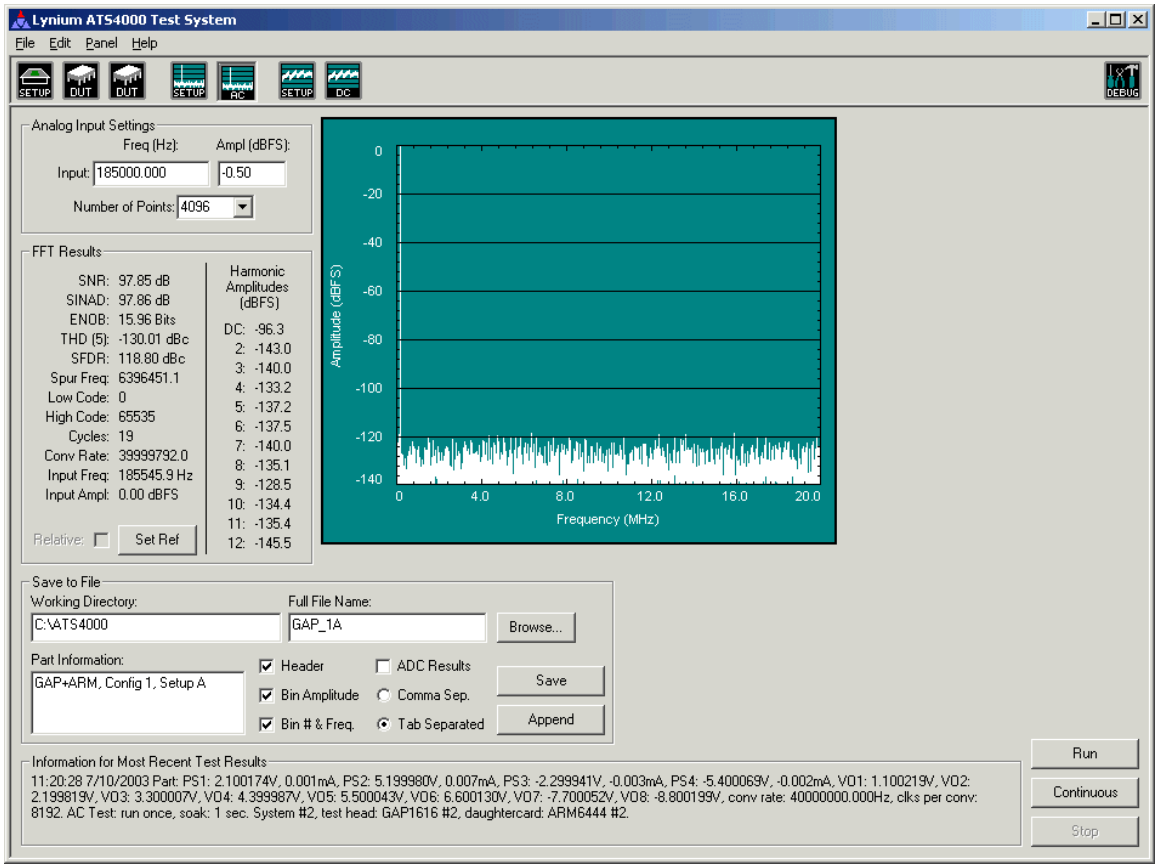

Figure 16. Result of an AC Test with the RTB1616 Installed.

A successful test requires the following GAP1616 hardware to be operating properly: Q0 through Q15, D0 through D15, PS 1 through PS 4, VO 1 through VO 8, VDUTD, and F2. For this test, the pattern generator is operating at 80 MHz and the data collection module at 40 MHz. (If the external clock source connected to EXT 8 does not go up to 80 MHz, enter it's maximum frequency in the Pattern Frequency text box of the first DUT SETUP panel, then run the AC test.)

The values reported in the FFT Results and the Information for Most Recent Test Results sections should be very close to those shown in Figure 16. The difference should only be in the last digit or last few digits of each result shown (supply and voltage source settings should be within 2 mV of the values shown). If this is not the case, then something may be wrong with the GAP1616 setup.

The functionality of DAC 1 (AWG) and DAC 2 (Y) can also be checked with the RTB1616. Simply click the Continuous button in the AC TEST panel. While the AC test is running, monitor the DAC 1 and DAC 2 outputs with an oscilloscope. Each should show a sine wave of with an amplitude of ±2.5 V and a frequency of approximately 185.6 kHz (see Figure 17).

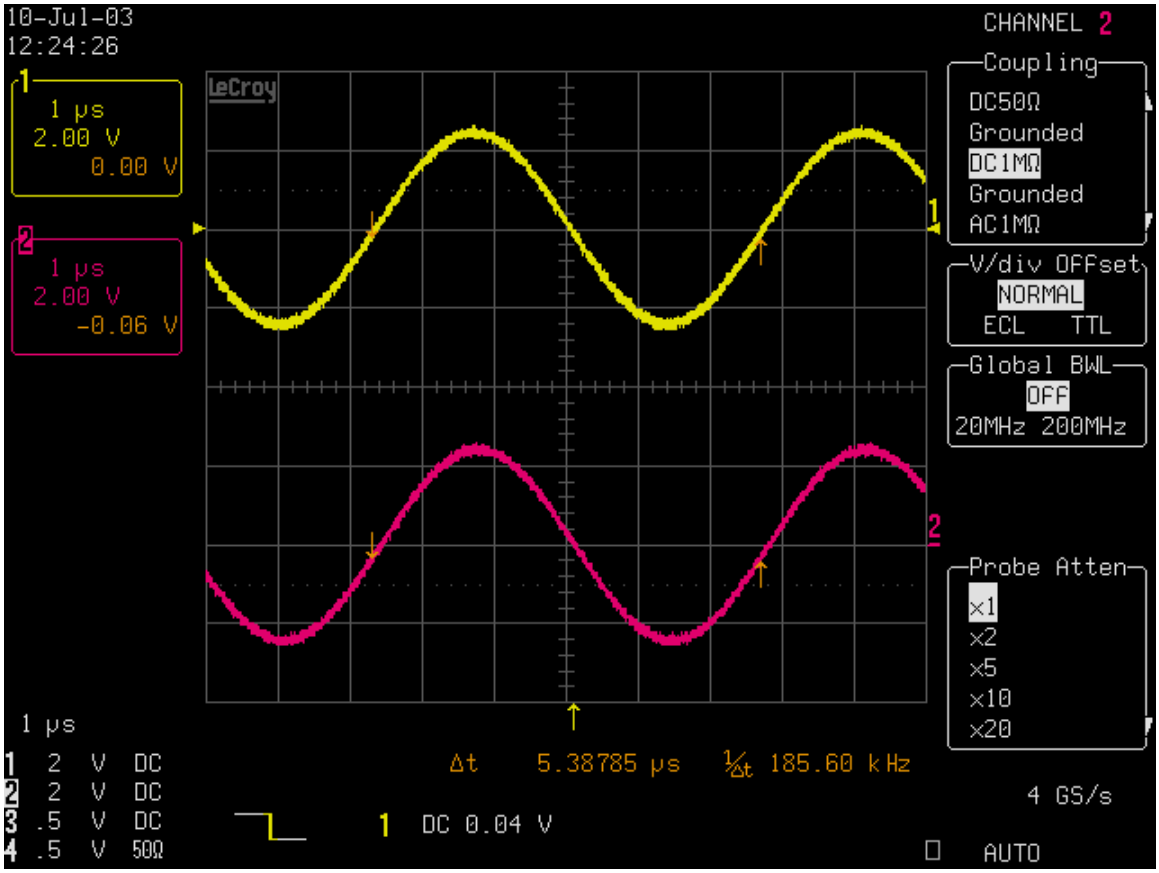

Figure 17. DAC 1 (top trace) and DAC 2 (bottom trace) Outputs when Continuous AC Testing is Selected with the RTB1616 Installed.

The RTB1616 has also been designed to test RA 1 through RA 8, C0 through C7, MP 1 through MP 6, MN 1 through MN 6, RDATA, RCLOCK, RLATCH, F3, F1, and F0, but this functionality is not supported with the current software (version 2.1.2). Do not change the DUT PANEL settings for these items when using the RTB1616—the board cannot used to check the items manually (a number of the resources are interconnected and must be configured carefully). For testing S1 and S2, use the RTH4000 board (which also tests the ATS4000 system supplies, PS 1 through PS 4, and the servo-loop circuitry).

(This page intentionally left blank)

7. VERSION HISTORY

Ver.	Date	Comment
A	July 10, 2003	Initial release.
B	February 12, 2004	Changed “vector” references to “pattern” and “acquisition” to “collection.”
C	April 2, 2010	Added details regarding software options for testing differential ADCs (mainly section 4.3.2). Updated some of the GUI screen captures.

