

# Test Station Documentation

Test station module consists of several functions that will give you access to hardware to send or receive digital and analog signals. Test station consists of one PCI National Instrument's data acquisition card and one homegrown motherboard that will give you access to 40 DAC channels.

Test station hardware profile is given below

- Number of digital I/O channels: 8
- Number of Analog inputs: 8
- Number of analog outputs: 2
- Number of DAC channels: 40

Note: There are 5 DACs with 8 channels each. 5 DACs are daisy-chained.

Test station module consists of different functions that will give you access to different components of test station hardware. Do not play with these Matlab modules if you are not sure what you are doing. There are some global variable used for writing these test station modules so use of these variable name is highly discouraged.

The global variable that are used for writing test station modules are as follows

## **Serial DAC global variables**

- MaxRange - (maximum supply voltage = 4.096)
- MinRange - (minimum supply voltage = 0)
- Dacbits - ( Number of DAC input bits = 16)
- SupplyRange - ( MaxRange – MinRange)

## **NI DAC card global variables**

- Ain - (Analog input channel object): 8 channels
- Aout1 - (Analog input channel 1 object): 1 channel
- Aout2 - (Analog input channel 2 object): 1channel
- diodac – (Digital i/o channel object): 8 channels
- diogen- (Digital i/o channel object): 8 channels \*

\*Currently not in use

All variables are case sensitive. Avoid using these variable names in your program to avoid any kind of conflict.

Test station modules consist of different functions with specific use. Explanation of all written functions with their functionalities is given below.

Note: All function names are case-sensitive so please remember the name of a function when you call it in your program..

## **Initialization and deletion functions**

- **InitializeNiCard()**

- **DeleteNiCard()**

Whenever you want to use any of the test station functions, you must initialize all hardware objects at the start of the program by calling `InitializeNiCard()`. Similarly when you want to finish your work on test station then delete all hardware objects by calling `DeleteNiCard()`. A 1 will be returned on successful execution of these functions. Your code should look like given below

```
status = InitializeNiCard();
```

```
...
```

```
...
```

Your Matlab code

```
...
```

```
...
```

```
status = DeleteNiCard();
```

### **Functions to access NI DAC card hardware**

**Digital I/O channels:** There are 8 digital I/O channels provided by NI data acquisition card, which can be configured as inputs or outputs. The First 4 channels (Channel 1 to 4) are reserved for serial DAC data transfer. So only 4 channels (Channel 5 to 8) are available for general-purpose usage.

You can send data bitwise or in hex (decimal) format by using function given below

- **SendBit (LineNo, BitValue)**

You can send bits by providing channel number (LineNo) and bit value (BitValue). Any number from 5 to 8 can be assigned to LineNo. A 1 will be returned on successful execution. In case if you accidentally assign LineNo with value from 1 to 4 then those numbers will automatically get mapped to channels ranging from 5 to 8.

- **SendHex (HexValue)**

You can send 4 bit data by assigning variable HexValue by any number from 0 to 15. A 1 will be returned on successful execution. In case if you assign value more than 15 then error message will be displayed.

Note: Output of `SendBit (LineNo, BitValue)` and `SendHex (HexValue)` can be observed at header HD0 of daughter card/motherboard at pins 15,13,11 and 9. Pin 15 is channel 5 and pin 9 is channel 8.

### **Analog Input/Output channels**

There are 8 analog inputs and 2 analog outputs that can be accessed using by matlab functions. On motherboard or daughter board 8 analog inputs are located at even numbered pins starting from pin 40 (DAQ\_AI0) to pin 26 (DAQ\_AI7) on connector HD0.2 analog outputs are pin 22 (DAQ\_AO0) and pin 21 (DAQ\_AO1) on connector HD0.

- **GetAnalogInput(Channel)**

This function measures analog input value at any specified channel ranging from 1 to 8. If you give channel number greater than 8 then error will be displayed.

- **SetAnalogOutput(Channel, AnalogValue)**

This function sets up specified DC value at specified channel. You can give any DC value from -10V to +10V. Channel number could be either 1 or 2. If you specify channel number other than 1 or 2 then error will be displayed. A 1 will be returned on successful execution.

### **Functions to access serial DACs**

There are 5 DACs on a motherboard with 8 channels each. They are serially connected in daisy-chained fashion. Each DAC consists of 32-bit shift register, which can be programmed by sending 32 bits serially using SDI pin. Output of 32-bit register is SDO pin that is connected to SDI pin of next DAC. DACs are numbered from 1 to 5 and channels are numbered from 1 to 8.

- **SetSpecificBias (DacNum, ChannelNum, DacValue)**

A specific channel from specific DAC can be programmed using SetSpecificBias function. You can specify any DAC number from 1 to 5 by assigning variable DacNum. A channel number from 1 to 8 can be assigned to ChannelNum variable. You can specify any DC value from 0 to 4.096V to variable DacValue.

e.g. By calling function SetSpecificBias (2,6,3.2), you are programming sixth channel of second DAC with value 3.2V.

DAC has maximum output value of 4.096V. So if you specify DC value greater than 4.096V then it will be rounded off to output value of 4.096V.

- **SetAllBias (dacvalues)**

This function sets all the biases at one time by assigning array of DC values to variable dacvalues.

e.g. dacvalues = [1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4]

SetAllBias (dacvalues) will start programming given DC values from DAC 1 channel1 to DAC 2 channel 8.

You assign up to 40 DC values. If you pass array of values with array size greater than 40 then first 40 values will be programmed and rest will be discarded. DAC has maximum output value of 4.096V. So if you specify DC value greater than 4.096V then it will be rounded off to output value of 4.096V.

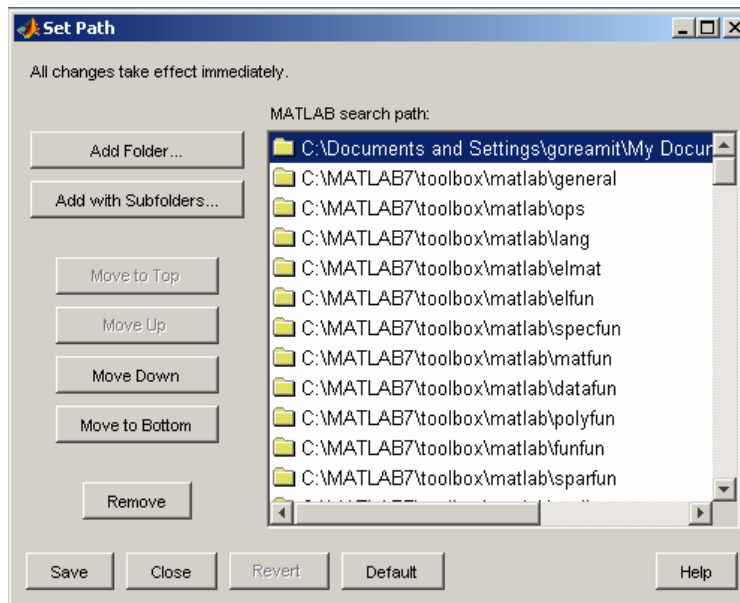
- **SendSerialDacData(SDI)**

This is an internal function used by SetAllBias (dacvalues) and SetSpecificBias (DacNum, ChannelNum, DacValue) functions. It sends 160 bits (32 bit each for all 5 DACs) serially. Do not make changes to this function unless you know what you are doing.

### **Instructions to setup path for test station modules in matlab**

- Download TestStation.zip file and then unzip TestStation.zip file. You will see a folder named "TestStation"
- Start Matlab.
- Set the path for test station modules by selecting File => Set Path =>

- A window will pop up as shown below



- Click on “Add Folder” and select TestStation folder. Save and close this window.
- Now you can use any function from test station folder.
- You ca also view paths set up by typing path on matlab command line.

### **Testing using Spartan-3 FPGA board**

Spartan-3 FPGA board offers lot of flexibility in testing digital domain of chips. You can use FPGA board along with homegrown motherboard if you want to perform high speed testing. Only problem that could be faced while testing with FPGA is time spent in synthesis. Every time you make change in your program you have to go through hardware synthesis process which could be time consuming.

Spartan 3 board has 8 slide switches, 4 pushbuttons, 9LEDs and 4-digit seven-segment display. There are 3 40-pin expansion connectors on FPGA board. Some pins from connectors A1 and A2 are interfaced with homegrown motherboard through connector HD1. All digital inputs and outputs of FPGA have range from 0 to 3.3V so care must be taken when interfacing with 5V digital circuits.

### **Procedure for testing chips using FPGA**

- Write your VHDL/Verilog program.
- Find out input and output pins from your program and map them with available hardware in .ucf file
- Inputs can be mapped with pins from connector HD1 of motherboard. Please refer motherboard documentation for available FPGA port pins. Inputs can also be mapped with 4 push buttons, 8 slide switches, FPGA clock, and JTAG clock.
- Outputs can be mapped with pins from connector HD1 of motherboard. You can also make use of 8 LEDs and 4-digit seven-segment display for digital outputs.
- Program FPGA and run your test program.

## FPGA board hardware port map

- **Switches**

Switch	SW0	SW1	SW2	SW3	SW4	SW5	SW6	SW7
FPGA pin	F12	G12	H14	H13	H14	J13	K14	K13

- **Push Buttons**

Push Button	BTN3 (User Reset)	BTN2	BTN1	BTN0
FPGA pin	L14	L13	M14	M13

- **Clock Sources**

Oscillator Source	FPGA pin
50MHz (IC4)	T9
Socket (IC8)	D9

- **LEDs**

LED	LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
FPGA pin	K12	P14	L12	N14	P13	N12	P12	P11

- For more detailed documentation of FPGA board refer following link  
<http://www.digilentinc.com/Data/Products/S3BOARD/S3BOARD-rm.pdf>
- **HD1 connector of motherboard** (HD1 connector makes connections with some pins from connector A0 and some pins from connector A1 of FPGA board.) gets specific port pins from connectors A1 and A2 of FPGA board.

FPGA pin	HD1 connector pin		FPGA pin
R5	1	2	T5
C1	3	4	C2
M7	5	6	B1
M10	7	8	E6
C5	9	10	D5
C6	11	12	D6
C7	13	14	E7
C8	15	16	D7
C9	17	18	D8
A3	19	20	D10
A4	21	22	B4
A5	23	24	B5
B7	25	26	B6
B8	27	28	A7
A9	29	30	A8
A10	31	32	B10
B12	33	34	B11
B13	35	36	A12
B14	37	38	A13
DGND	39	40	DVDD (3.3V)

Sample entity declaration in VHDL and its hardware port map is given below.

**VHDL entity declaration**

entity ConfigureFPGA is

```
    Port (
        CLKIN : in std_logic;
        PUSHBUTTON: in std_logic;
        SWITCHES: in std_logic_vector( 2 downto 0);
        HD1IN: in std_logic_vector( 2 downto 0);
        LED: out std_logic_vector( 2 downto 0);
        CLKOUT: out std_logic;
        HDOUT: out std_logic_vector (2 downto 0)
    );
end ConfigureFPGA;
```

**FPGA hardware port map (.ucf) file**

```
#PACE: Start of Constraints generated by PACE
#PACE: Start of PACE I/O Pin Assignments
# JTAG clock on FPGA board is located at T9.
NET "CLKIN" LOC = "T9" ;
# Push button on FPGA board is mapped below
NET " PUSHBUTTON" LOC = "M13" ;
# Switches on FPGA board are mapped below
NET "SWITCHES<0>" LOC = "F12" ;
NET "SWITCHES<1>" LOC = "G12" ;
NET " SWITCHES<2>" LOC = "H14" ;
# Pins from connector HD1 on motherboard are mapped as inputs
NET "HD1IN<0>" LOC = "C5" ;
NET "HD1IN<1>" LOC = "C6" ;
NET "HD1IN<2>" LOC = "C7" ;
# LEDs on FPGA board are mapped as outputs
NET "LED<0>" LOC = "K12" ;
NET "LED<1>" LOC = "P14" ;
NET "LED<2>" LOC = "L12" ;
# CLOCK is made available at port C8 pin on connector HD1
NET "CLKOUT" LOC = "C8" ;
# Pins from connector HD1 on motherboard are mapped as outputs
NET "HDOUT<0>" LOC = "E6" ;
NET "HDOUT<1>" LOC = "D5" ;
NET "HDOUT<2>" LOC = "D6" ;
#PACE: Start of PACE Area Constraints
#PACE: Start of PACE Prohibit Constraints
#PACE: End of Constraints generated by PACE
```