8751 VSG

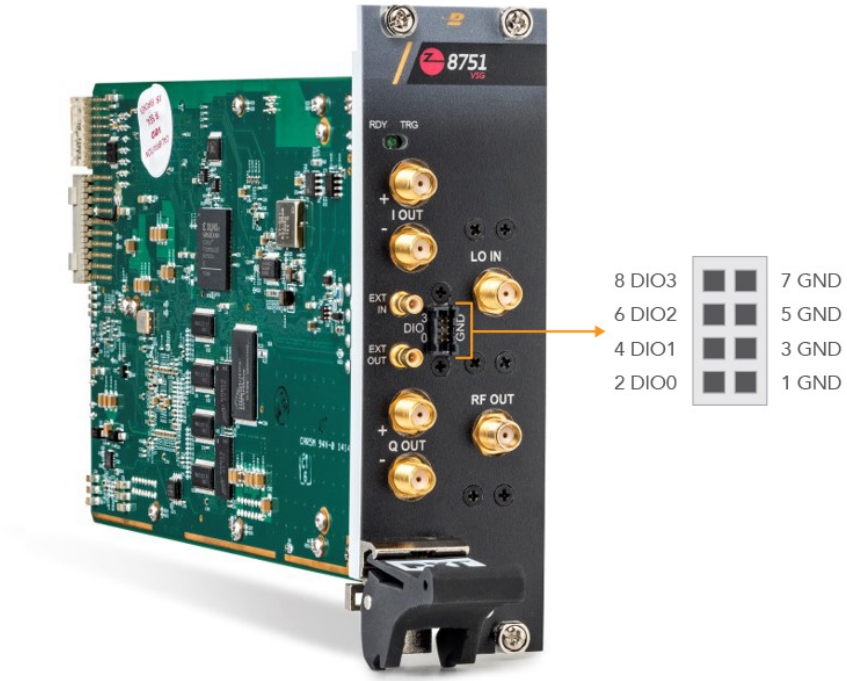
APPLICATION NOTES

z8751 VSG DIO

LITEPOINT

Contents

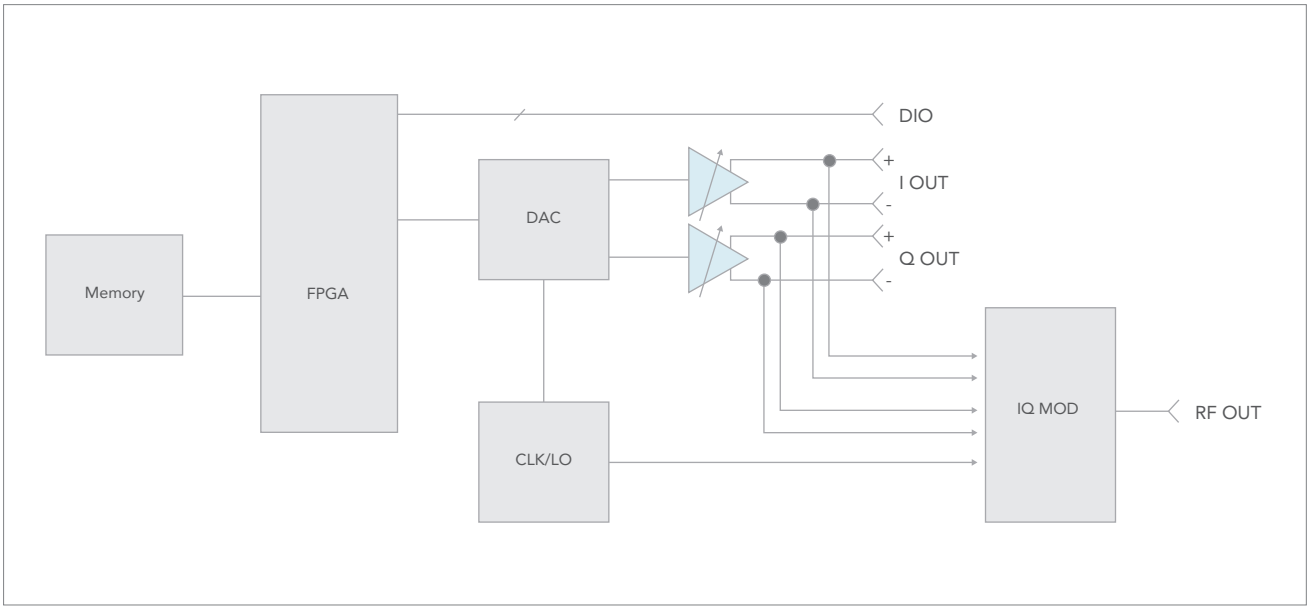
Port Descriptions



Front Panel

Label	Type	Description
I OUT +,-	SMA	Differential baseband I output
Q OUT +,-	SMA	Differential baseband Q output
EXT IN	SMB	External input for trigger or reference
EXT OUT	SMB	External output for trigger, reference or event
RF output	SMA	RF output
LO IN	SMA	Local oscillator input
DIO	Header, 8-pin, 0.05" spacing	Digital input/output, 4-signal (e.g. MIPI, SPI, I^2C)

Block Diagram



Digital Input/Output (DIO)[1]

General Specifications

Specification	Value
Functionality	4-bit bi-directional Digital I/O software programmable. Supports serial interfaces such as MIPI, SPI, I2C, etc.
Programmable Clock Rate	Up to 125 MHz
Programmable Logic	≥ 8 ns resolution
Programmable Direction	Input (52 kΩ pull-down) or Output
Programmable Source/Destination	Backplane triggers, external in/out, trigger event
Output Level	Programmable Level: Default: +1.2V into open load Range: +1.2V to +3.6V into open load Level accuracy: ±5%
Output Drive	≥ ±3 mA @ 1.2V ≥ ±8 mA @ 1.8V ≥ ±12 mA @ 3.6V
Output Enable	Tri-State Output Capability
Connector	Latching Header, 8-pin, 0.05" spacing 4 DIO signals with ground pairs

DIO Clock Rates

Divider	Sample Clock	MIPI Clock	REF Clock	SSBI Clock
1 (no divide)	125 MHz	104 MHz	100 MHz	38.4 MHz
2	62.5 MHz	52 MHz	50 MHz	19.2 MHz
4	31.25 MHz	26 MHz	25 MHz	9.6 MHz
6	20.833 MHz	17.333 MHz	16. 666 MHz	6.4 MHz
8	15.625 MHz	13 MHz	12.5 MHz	4.8 MHz
10	12.5 MHz	10.4 MHz	10 MHz	3.84 MHz
12	10.4166 MHz	8.666 MHz	8.333 MHz	3.2 MHz
…	…	…	…	…
510	245.098 kHz	203.921 kHz	196.078 kHz	75.294 kHz

1 DIO connector available only on PXIe product revision 3 and later.

Pinout

Pin Number	Signal	Pin Number	Signal
1	GND	2	DIO0
3	GND	4	DIO1
5	GND	6	DIO2
7	GND	8	DIO3

MIPI Functionality

Feature	Details
Transactions Supported	Write register 0 Write register Read Register Write Extended Read Extended Write Long Read Long
Sequences	Up to 4 serial data stream sequences Max of 32 RD/WR MIPI byte transactions per sequence, Multi-sequence capture capable, Single transaction pre/post a burst transactions
Triggering	Independent triggers per data stream, armed by previous sequence completion
Buffer Update	Ability to update one buffer while playing other buffer (ping/pong)

Accessories: Mating Connectors & Cables

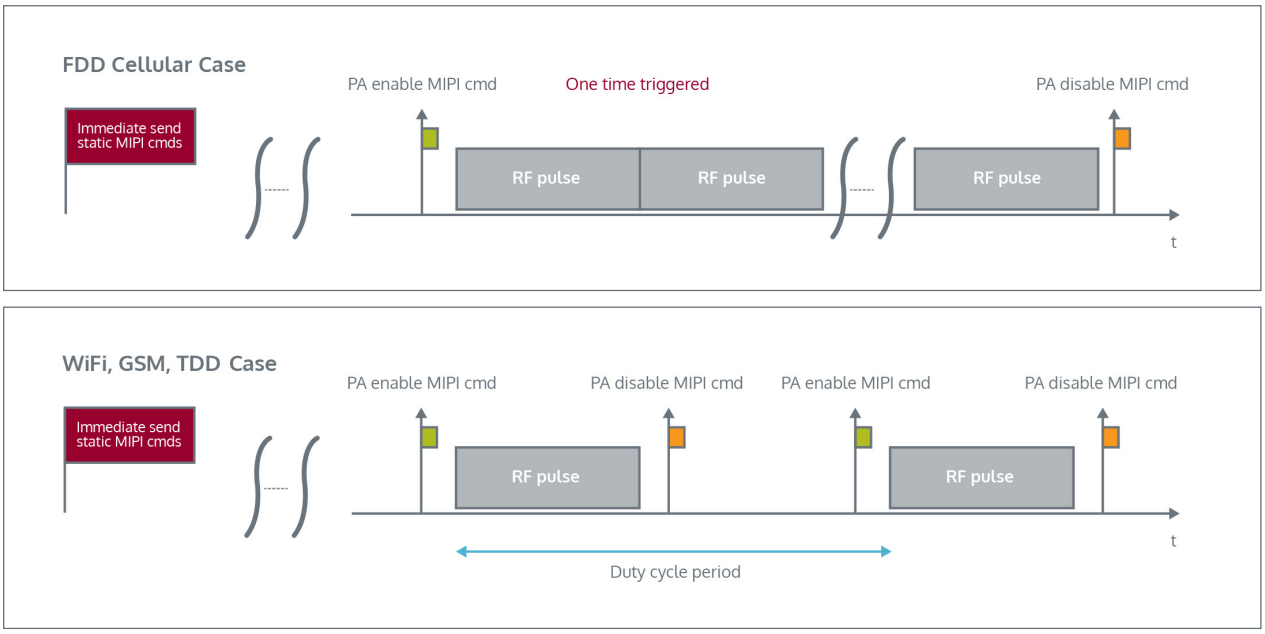
Part	Part Number
Mating Header Connector (requires own cable)	3M, 45108-010030 Strain relief: 3M, 3448-45108
6" Mating Cable with header connector	LitePoint, 0150-ZDIO-002
12" Mating Cable with header connector	LitePoint, 0150-ZDIO-003
24" Mating Cable with header connector	LitePoint, 0150-ZDIO-004
24" Mating Cable with header connector and Breakout Board	LitePoint, 0150-ZDIO-001

Applications

The LitePoint z8751 Vector Signal Generator is suitable for WLAN 802.11 a/b/g/j/n/p/ac/af/ah/ax, Cellular 2G/3G/4G, Bluetooth and ZigBee standards. The z8751 provides RF and baseband I/Q signal outputs corresponding to vector modulated signals. In addition, a DIO connector adds serial interfaces such as MIPI, SPI and I^2C that can be time synchronized to the RF signal.

MIPI-RFFE testing is a specific application for the DIO connector. Preloaded MIPI patterns are generated in a sequence that is time aligned with the RF output signal. This is necessary for PA/FEM testing where the device is enabled/disabled using a MIPI command that must be accurately time-aligned with the RF burst.

Synchronized MIPI testing



Synchronized MIPI testing is a challenging procedure. The time gap between the MIPI command and the RF pulse should be almost negligible. The DIO connector and the RF signal synchronize automatically through the same FPGA. The auto synchronization is successful with burst and modulated RF signals.

SPI is a serial interface method designed for short distance communication. PA/FEM components with SPI capabilities exchange synchronized data with a master device (z8751). The SPI packet determines mode and state of PA/FEM components.

The DIO connector also configures General Purpose Input/output (GPIO) functionalities. GPIO pins are unused by default, but can be configured as input or output depending on the testing scenario. During testing GPIO pins set input/output modes on chipsets, external electronics, switches, etc.

DIO Output Commands

Command	Parameter Form	Response	Notes
:DIO:CLOCk	<frequency>		DIO BB Only, <frequency in Hz>
:DIO:CLOCk?		<frequency>	
:DIO<n>:DATA	<1|0>		<n> = 0-3 DIO pin selection (PXIe_DIO only)
:DIO<n>:DATA?		<1|0>	<n> = 0-3 DIO pin selection. Reads back configured output data. See sense command for read data. (PXIe_DIO only)
:DIO<n>:DIRection	<IN|OUT>		<n> = 0-3 DIO pin selection. (PXIe_DIO only)
:DIO<n>:DIRection?		<IN|OUT>	<n> = 0-3 DIO pin selection. (PXIe_DIO only)
:DIO:ENABle	<ON|OFF,1|0>		Global Enable (PXIe_DIO only)
:DIO:ENABle?		<1|0 >	Query global enable (PXIe_DIO only)
:DIO:LEVel	<voltage>		Voltage is floating point < 3.6V (PXIe_DIO only)
:DIO:LEVel?		<voltage>	Floating point (PXIe_DIO only)
:DIO:RESet			Resets SDS to Defaults/Legacy
:DIO:SEQuence:BURSt	<ON|OFF,1|0>		1=Burst (single-shot), 0 = continuous
:DIO:SEQuence:BURSt?		<ON|OFF,1|0>	1=Burst (single-shot), 0 = continuous
:DIO:SEQuence:CLEar	<sequence_ID>		Clears sequence of commands.
:DIO:SEQuence:DELay	<sequence_ID>, <cycles>		<sequence_ID> specifies the SDS command buffer <cycles> sets the number of clock cycles (periods) to delay
:DIO:SEQuence:DELay?	<sequence_ID>,	<cycles>	<sequence_ID> specifies the SDS command buffer <cycles> returns the currently configured number of delay cycles
:DIO:SEQuence:GAP	<gap>		<gap> sets the number of extra clock cycles at the end of each command that the SDS bus will be left at idle.
:DIO:SEQuence:GAP?		<gap>	<gap> returns the currently configured command gap, def:5
:DIO:SEQuence:STATus?		<status>	Returns the current state machine status register (bits 0:3).

:DIO:SEQuence:TRIGger	<seq_id>,<trig_src>, <trig_mode>, <trig_pol>, <trig_dly_en>		<seq-id> = 1-4, SDS Sequence Number <trig_src> = refer to footnote in page 9 <trig_mode> = <1|0>, 0=edge,1=level <trig_pol> = <1|0>, 0=normal, 1=inverted (low-truth) <trig_dly_en> = <1|0>, 1=enabled delay, 0 = no delay
:DIO:SEQuence:TRIGger?	<seq_id>	<trig_src>,<trig_mode>, <trig_pol>, <trig_dly_en>	
:DIO<n>:SOURce	0-7 or TTLTrg0-7 9 or DIO (default) 10 or SUBModule1 12 or MIPData 13 or MIPClk 14 or EXTernal 8,11, 15: Reserved		<n> = 0-3 DIO pin selection. (PXIe_DIO only)
:DIO<n>:SOURce?		0-7 or TTLTrg0-7 9 or DIO (default) 10 or SUBModule1 12 or MIPData 13 or MIPClk 14 or EXTernal 8,11, 15: Reserved	<n> = 0-3 DIO pin selection. (PXIe_DIO only)
:DIO:MIPI:IMMediate:READ?	<MIPI_Slave_ID>, <half_speed>, <slave_reg_addr>	<slave_response, 1 byte>	Immediately generates MIPI "Write Register" command
:DIO:MIPI:IMMediate:WRITe	<MIPI_Slave_ID>, <half_speed>, <slave_reg_addr>, <slave_reg_data>		Immediately generates MIPI "READ Register" command
:DIO:MIPI:IMMediate:ZERO	<MIPI_Slave_ID>, <slave_reg_data>		Immediately generates the protocol specific MIPI "Write Zero Register" command
:DIO:MIPI:SEQuence:APPend	<SDS_Sequence_ID>, <Read_write>, <MIPI_Slave_ID>, <slave_reg_addr>, <slave_reg_data>		Appends a single MIPI command to the specified command sequence buffer. SDS_Sequence_ID = 1-4
:DIO:MIPI:SEQuence:ENABle	<SDS_Sequence_ID>, <ON|OFF,1|0>		Enables specified buffer and configures internal signal routing for MIPI command generation.
:DIO:MIPI:SEQuence:ENABle?	<SDS_Sequence_ID>	<ON|OFF,1|0>	Returns whether the specified buffer is enabled.
:DIO:MIPI:SEQuence:HALF	<1|0>		Enables/Disables Half-speed clocking during data read.
:DIO:MIPI:SEQuence:HALF?		<1|0>	
:DIO:SSBI:IMMediate:PAIR?	<read_write>, <pair_id>, <slave_reg_addr>, <slave_reg_data>	Read:<slave_response> Write: <0>	<read_write> = 1 for write, 0 for read <pair_id> = 0|1 Immediately generates SSBI register read/write command on specified DIO port pairing.

:DIO:SSBI:IMMediate:READ?	<slave_reg_addr>	<slave_response, 1 byte>	Immediately generates SSBI "READ Register" command
:DIO:SSBI:IMMediate:WRITe	<slave_reg_addr>, <slave_reg_data>		Immediately generates SSBI "Write Register" command
:DIO:SSBI:SEQuence:APPend	<SDS_Sequence_ ID>, <Read_write>, <slave_reg_addr>, <slave_reg_data>		Appends a single SSBI command to the specified command sequence buffer. SDS_Sequence_ID = 1-4
:DIO:SSBI:SEQuence:BLOCk?	<command_count>	<num_responses>	Executes a previously built sequence of commands and returns the number of read commands executed. Read responses are stored in ScratchPad.
:DIO:SSBI:SEQuence:ENABle	<SDS_Sequence_ ID>, <ON|OFF,1|0>		Enables specified buffer and configures internal signal routing for SSBI command generation.
:DIO:SSBI:SEQuence:ENABle?	<SDS_Sequence_ ID>	<ON|OFF,1|0>	Returns whether the specified buffer is enabled.
:DIO:SYNC:CLEAR			Clears Sync config and buffers.
:DIO:SYNC:COERCe	<on|off,1|0>		Enables ITG Gap coercion to synchronize SDS:SYNC and waveform play NOTE: Sample clock only: 125e6, 62.5e6 & 31.25e6
:DIO:SYNC:COERCe?		<1|0>	
:DIO:SYNC:ENABle	<on|off,1|0>, <bytes>		Configures SDS CMD Buffer 4 and routes outputs of RCV1 and RCV2 to DIO 2/3 respectively
:DIO:SYNC:ENABle?		<1|0>	Returns whether a sequence is enabled on SDS CMD buffer 4.

Triggering Options:

"TRIG" = Trigger Even

"GCOM" = Generation Complete (end of waveform)

"EXT" = External Input on instrument

"CONS" = Constant register (manually toggled)

"TSYN" = Delayed trigger, (i.e., PA Enable delayed trigger)

"SYNC1" = SYNC1 output/marker

"SYNC2" = SYNC2 output/marker

"DIO<0-3>" = DIO input pins 0-3
(note: Pins 0 and 1 are used for MIPI, Pins 2 & 3 are used for SSBI,
You will need to use "unused" pins for the protocol you are currently working with)

"TTL<0-7>" = TTL (PXI Backplane triggers 0-7)

MIPI Command Sequencing Example

```
//DIO Port Init, perform once to set up the communications port:
  zbind_send(ztvsg_handle,ZT_TRUE,"outp:dio:reset");
  zbind_send(ztvsg_handle,ZT_TRUE,"outp:dio:enab on");
  zbind_send(ztvsg_handle,ZT_TRUE,"outp:dio:lev 1.8");
  zbind_send(ztvsg_handle,ZT_TRUE,"outp:dio:clock 25e6");
  zbind_send(ztvsg_handle,ZT_TRUE,"outp:dio0:sour 13"); // MIPI Clock pin = DIO0
  zbind_send(ztvsg_handle,ZT_TRUE,"outp:dio1:sour 12"); // MIPI Data pin = DIO1
  // Note: you can specify the DIO pins (0-3)

//Perform immediate commands, when needed:
//MIPI immediate Cmd, writes 1 byte to a slave register address:
  //   Params: Slave Addr,Half-speed,RegAddr,Data
  zbind_send(ztvsg_handle,ZT_TRUE,"outp:dio:mipi:imm:write %d,0,11,3",slave_addr);

//MIPI Immediate Read:
  //   Params: Slave Addr,Half-speed,RegAddr
  zbind_send(ztvsg_handle,ZT_TRUE,"outp:dio:mipi:imm:read? %d,0,11", slave_addr);
  //   Use zbind_receive to get the return data, 1Byte only:
  zbind_receive(ztvsg_handle,ZT_TRUE,"%d",&data);
```

zbind.h information on the zbind_send() and zbind_receive() commands:

```
/*=============================================================================*/
/* Function: zbind_send                                                        */
/* Purpose:  This function sends a command string to the instrument with      */
/*           optional locking and string formatting.                           */
/*         Parameter 1 is the zbind device handle.                             */
/*         Parameter 2 is the lock state. ZT_TRUE will lock subsequent         */
/*           commands until a zbind_receive or zbind_releaselock occurs.       */
/*         Parameter 3 is the format string. This should be in standard        */
/*           formatting, using string conversion specifications.               */
/*           The format string defines whether additional parameters are used. */
/*         Returns the status of the instrument communication.                 */
/*=============================================================================*/
ZT_ERROR _ZBIND_FUNC zbind_send (ZT_HANDLE dev, ZT_BOOL lock, s8 format_str[], ...);
/*=============================================================================*/
/* Function: zbind_receive                                                     */
/* Purpose:  This function returns a response string from the instrument      */
/*           with optional locking and string formatting.                      */
/*         Parameter 1 is the zbind device handle.                             */
/*         Parameter 2 is the lock state. ZT_TRUE will unlock a current        */
/*           lock, ZT_FALSE will only call the instrument if it is not         */
/*           currently locked.                                                 */
/*         Parameter 3 is the format string. This should be in standard        */
/*           formatting, using string conversion specifications.               */
/*           The format string defines whether additional parameters are used. */
/*         Returns the status of the instrument communication.                 */
/*=============================================================================*/
ZT_ERROR _ZBIND_FUNC zbind_receive (ZT_HANDLE dev, ZT_BOOL unlock, s8 format_str[], ...);
```

RESTRICTED RIGHTS LEGEND

DISCLAIMER

LitePoint Corporation makes no representations or warranties with respect to the contents of this manual or of the associated LitePoint Corporation products, and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. LitePoint Corporation shall under no circumstances be liable for incidental or consequential damages or related expenses resulting from the use of this product, even if it has been notified of the possibility of such damages.

If you find errors or problems with this documentation, please notify LitePoint Corporation at the address listed below. LitePoint Corporation does not guarantee that this document is error-free. LitePoint Corporation reserves the right to make changes in specifications and other information contained in this document without prior notice.

TRADEMARKS

CONTACT INFORMATION

LitePoint Corporation
575 Maude Court
Sunnyvale, CA 94085-2803
United States of America

+1.866.363.1911
+1.408.456.5000

LITEPOINT TECHNICAL SUPPORT

www.litepoint.com/support

Doc: 1075-0078-001
July 2017 Rev 2