User Manual

SC4410 Scout

Number: SCT-UM126DVC

Date: 2017-04-27

SIGNAL CRAFT TECHNOLOGIES

6815 – 8 Street NE, Suite 295
Calgary, AB T2E7H7 Canada
www.signalcraft.com

This document contains proprietary information and shall not be disclosed, in whole or in part, to any party without the expressed and written consent of an authorized representative of SignalCraft Technologies Inc. ©2013

Revision History

Revision	Date	Author	Description
0.5	2012-11-20	L.Bratvold	Initial Draft
1.0	2012-12-07	L.Bratvold	Release
2.0	2013-04-19	B. Balla	Updates to reflect added features SPI / Dual Edge Triggering
2.1	2013-10-21	B. Balla	Add FCC compliance info
2.2	2015-10-30	A. Pineda	Added RFFE v2.0 spec commands
2.3	2016-01-23	B. Balla	Update SPI Commands
2.4	2016-06-25	B. Balla	Update RFFE v2.0 spec commands
2.5	2016-11-03	B. Balla	Updated clk command to include 38.4 & 19.2 MHz commands
2.5.1	2017-04-27	M.Puppato	Minor edits

Table of Contents

REVISION HISTORY **2**

TABLE OF CONTENTS **3**

1. FCC COMPLIANCE **6**

2. QUICK START **6**

3. USAGE **7**

3.1 Block Diagram 7
3.2 Power Supply 8
3.3 Installation 8
3.4 Control Interface 9
3.4.1 Command Prompt 10
3.4.2 Parameter Format 11
3.4.3 Script Comments 11
3.4.4 Status Register 11
3.4.5 Count Value 12
3.5 LED Control 12
3.5.1 Command Details 12
3.6 VIO Control 13
3.6.1 Internal 13
3.6.2 External 13
3.6.3 Command Details 13
3.7 SCLK Control 14
3.7.1 Command Details 14
3.8 Delay Control 14
3.8.1 Command Details 14
3.9 Serial Commands 15
3.9.1 Serial Read Configuration (SET_READ) 15
3.9.2 Register Zero Write (RZW) 15
3.9.3 Register Write (RW) 16
3.9.4 Register Read (RR) 16
3.9.5 Extended Register Write (ERW) 16
3.9.6 Extended Register Write (ERR) 17

3.9.7 Extended Register Write Long (EWL) 17
3.9.8 Extended Register Read Long (ERL) 17
3.10 GPIO Commands 18
3.10.1 I/O Usage 18
3.10.2 I/O Default State 18
3.10.3 GPIO Direction Write (GDW) 19
3.10.4 GPIO Direction Read (GDR) 19
3.10.5 GPIO Write (GPW) 19
3.10.6 GPIO Read (GPR) 19
3.10.7 GPIO Examples 20
3.11 SPI Commands 20
3.11.1 SPI Configuration 21
3.11.2 SPI Mode (SPI_MODE) 21
3.11.3 Display SPI Status (SPI_STATUS) 21
3.11.4 SPI Configuration (SPI_CFG) 22
3.11.5 SPI Write Data (SPI_WR) 22
3.11.6 SPI Read Data (SPI_RD) 23
3.11.7 SPI Slave Device Select (SPI_SELECT) 24
3.11.8 SPI Slave Select Polarity (SPI_POL) 24
3.11.9 SPI SCLK RATE (SPI_CLOCK) 24
3.11.10 SPI PROTOCOL MODE (SPM) 26
3.11.11 SPI COMMAND WORD WIDTH (COM_WIDTH) 26
3.11.12 SPI ADDRESS WIDTH (ADD_WIDTH) 26
3.11.13 SPI WRITE DATA WIDTH (WR_WIDTH) 27
3.11.14 SPI READ DATA WIDTH (RD_WIDTH) 27
3.11.15 SPI READ WAIT CYCLES (SPIWC) 27
3.12 Operating Mode 28
3.12.1 Normal Mode 28
3.12.2 Buffer Mode 28
3.13 Triggering 28
3.13.1 Software Trigger 28
3.13.2 Hardware Trigger 28
3.13.3 Command Details 29
3.14 Read Buffer 29
4. CONNECTIVITY 30
4.1 Alternate Power Connector 30
4.1.1 Interface Specs 30

4.1.2 MATING CONNECTOR 30
4.2 USB CONNECTOR 30
4.2.1 INTERFACE SPECS 30
4.3 ETHERNET CONNECTOR 30
4.4 TRIGGER INPUT CONNECTOR 31
4.4.1 INTERFACE SPECS 31
4.5 DUT INTERFACE CONNECTOR 31
4.5.1 INTERFACE SPECS 31
4.5.1.1 VIO (INTERNAL): 31
4.5.1.2 VIO (EXTERNAL): 31
4.5.1.3 DUT I/O: 31
4.5.2 MATING CONNECTOR 31
4.5.3 PIN-OUT RFFE MODE 31
4.5.4 PIN-OUT SPI MODE 32
5. FIRMWARE UPDATE 32
5.1 FIRMWARE UPDATE PROCEDURE 33
6. APPENDIX A: USING SCOUT WITH A LINUX MACHINE 34

1. FCC Compliance

This device is exempt from the technical requirements of Part 15 pursuant to paragraph 15.103 (c) as being a digital device used exclusively as industrial or commercial test equipment. Its operation is subject to the following conditions:

1. This device may not cause harmful interference.

2. This device must accept any interference received, including interference that may cause undesired operation.

NOTE: Although this equipment is exempt from the technical requirements of Part 15, it has still been tested following good practice and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy, and if it is not installed and used in accordance with the instruction manual, it may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.

2. Quick Start

1. Connect the Device Under Test to the Scout DUT Interface Connector: VIO, GND, SCLK (IO1) and SDATA (IO2)

2. Connect Scout to a USB port on the host Windows or Linux PC.

3. Front panel Status LED will turn green when Scout is ready (approximately 5 seconds after power-up).

4. If Windows prompts for a driver, browse to Scout INF file (request by emailing support@signalcraft.com)

5. Check Windows Device Manage (Computer>Properties>Device Manager) to determine the COM port assigned to Scout

6. Open a terminal emulator, select the COM port and configure with settings 115200-8-N-1

7. Enter ‘help’ at the command prompt (->) for a list of support commands

8. Command specific help is available as well

   Example - **help rw**

9. To write to the DUT, issue one of the write commands

   Example - rw 1 0x10 0x5A

10. To read from the DUT, issue one of the read commands, then display the read buffer

    Example - **rr 1 0x10**

3. Usage

The following document describes the usage of the SC4410 Scout I/O Adapter. To simplify the examples, this manual presents how to use Scout from a terminal emulator application. All of the concepts and commands described here would also apply to an external software application, such as Labview or 'C'.

Questions should be addressed to support@signalcraft.com if additional support is required.

3.1 Block Diagram

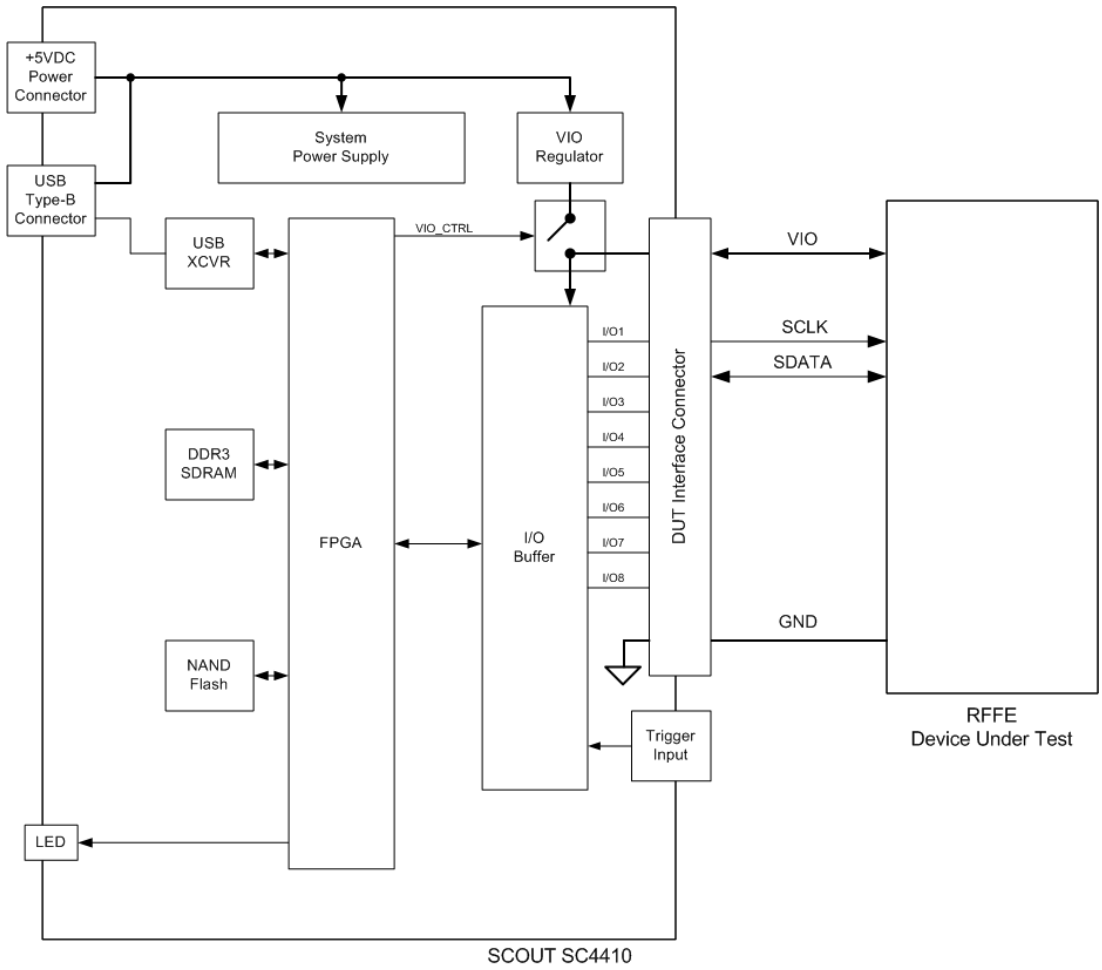

Figure 1- Block Diagram

*Note: The block diagram is a simplified representation of Scout. Some details have been omitted.

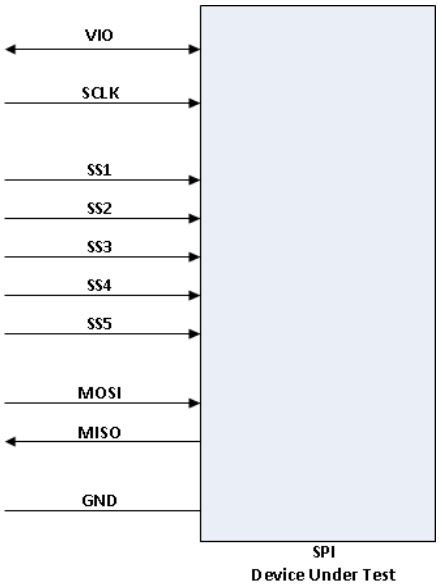

Figure 2- SPI Interface Block Diagram

3.2 Power Supply

Scout is USB bus-powered device.

Alternatively a +5V, 500mA external power supply can be connected at the 2-pin connector on the front panel. When an external power supply is detected, the USB port power is bypassed.

3.3 Installation

Scout uses a Windows or Linux built-in driver to provide a COM port control interface over USB between it and the host PC. For simplicity this manual describes the Windows Installation Process, but the Linux process will be similar.

When Scout is initially connected to a host PC, the Windows Driver Installer may prompt you for the location of an INF file. This INF file defines which Windows driver to associate with the Scout device. The INF file is included with the device or can be requested by emailing support@signalcraft.com.

Once installed Scout appears in the Windows Device Manager as a COM port. Make note of the COM port assigned to Scout (COM10 in the example below) as it will be required when configuring the terminal emulator COM port settings. The Device Manager can be launched by right-clicking your mouse on the 'My Computer' icon, selecting 'Properties' and then choosing the 'Device Manager' tab.

IMPORTANT NOTE: The COM port assignment will change if Scout is moved to a different USB port.

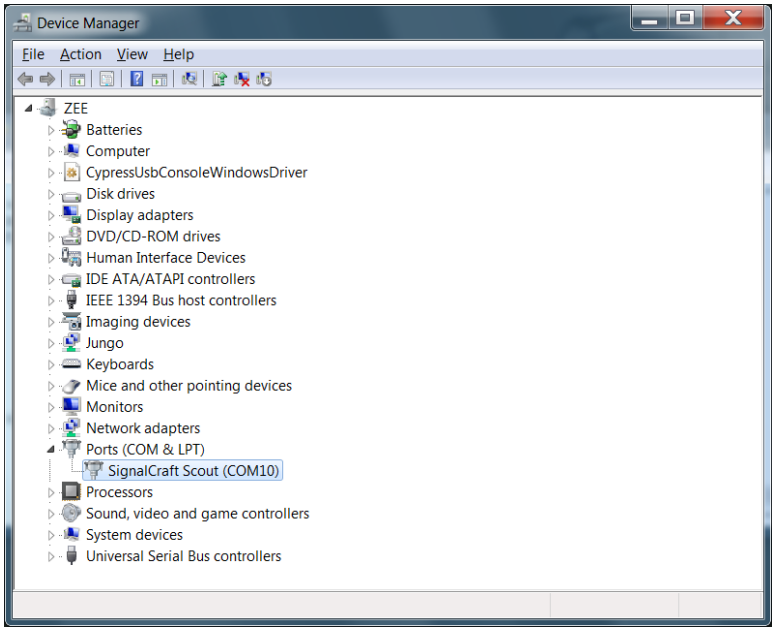

Figure 3- Windows Device Manager, Scout COM Port Assignment

3.4 Control Interface

Scout is controlled by predefined commands that are issued to it on a terminal interface. This allows simple terminal emulator programs such as Tera Term to be used to communicate with Scout. Alternatively, an external software application, such as Labview or 'C', can access and control Scout through this same COM interface. An example Labview GUI is provided with Scout or can be obtained from support@signalcraft.com.

The terminal program settings for the COM port assigned to Scout must be configured as 115200-8-N-1.

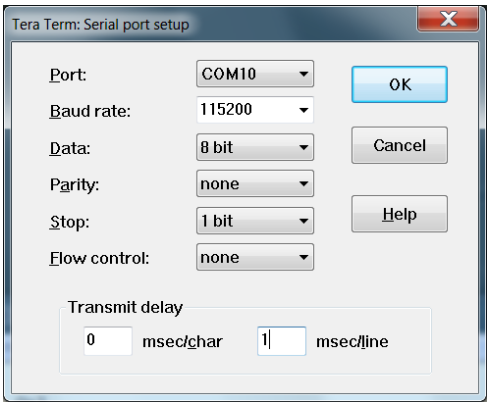

Figure 4- Example Tera Term COM Port Settings

Once the COM connection is established the user is able to issue any of the predefined commands listed in the help menu. Entering 'help' on the command line at any time during an active session displays a list of supported commands and a description of each. Commands are *not* case-sensitive.

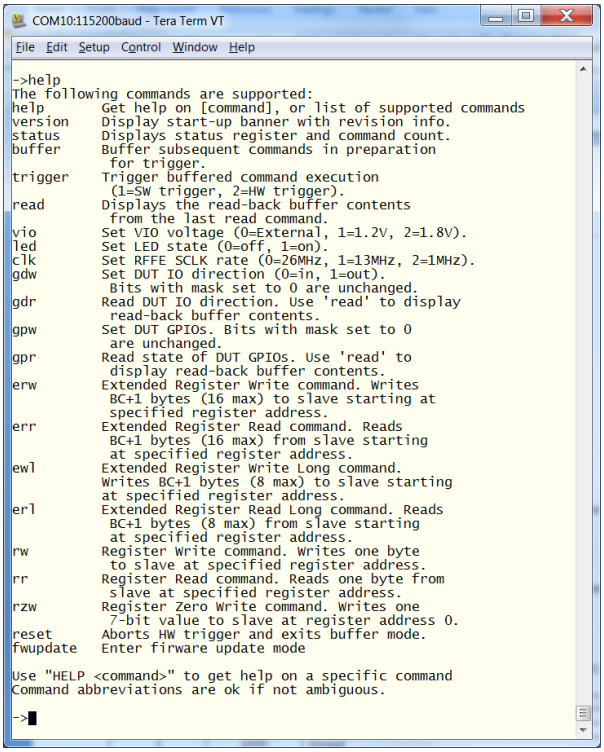

Figure 5- Help Command List

Command-specific help is also available by entering 'help' followed by a space and a specific command name. In addition to the description, command-specific help lists the parameters and syntax required when issuing the command.

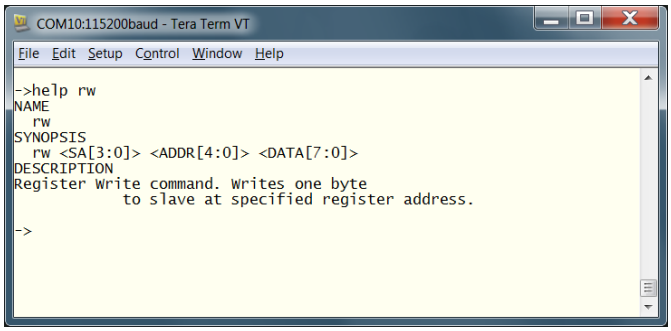

Figure 6- Example Command-Specific Help

IMPORTANT NOTE: USB COM port connections are not automatically released by Tera Term. Ensure the Scout COM port connection is closed before connecting Scout to the PC.

3.4.1 **Command Prompt**

Generally speaking, the only response to successful execution of a command is the command prompt '->'. However commands issued with incorrect parameter values, an incorrect number of parameters or otherwise cannot be successfully processed by Scout will result in an error.

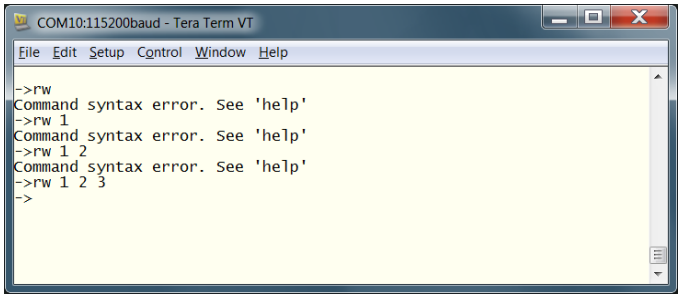

Figure 7- Example Command Entry Errors

3.4.2 Parameter Format

Command parameter values may be entered in decimal or hex notation, or both. Hex values require a'0x' prefix. For example, all three of the following commands are valid and equivalent.

rw 1 2 3
rw 0x01 0x02 0x03
rw 1 0x02 0x03

3.4.3 Script Comments

Comments in scripts are allowed if proceeded by the # character and white space. Comments must be on their own line, they cannot follow a command. For example, a comment insert for the previous commands.

 # Same Command in Different Entry Formats

rw 1 2 3
rw 0x01 0x02 0x03
rw 1 0x02 0x03

3.4.4 Status Register

A Status register provides insight into the device for use during system operation. The status flags provide the ability for user software to implement specific command-to-command delays, trigger timeouts, etc.

The Status register is displayed when the 'status' command is issued.

See Section 3.4.5 for details on the Count value.

IMPORTANT NOTE: Not all Status register bits are used. Unused bits are set to zero.

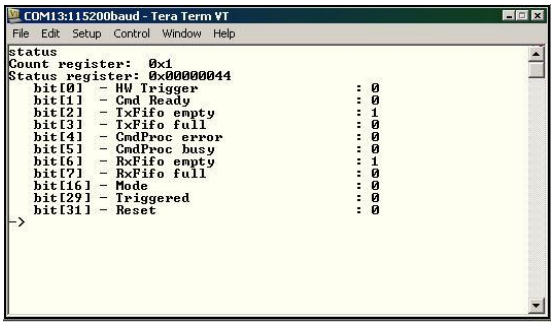

Figure 8 - Status Register Flags

The following table contains a description of each of the flags.

BIT	FLAG	DESCRIPTION (ACTIVE STATE)
0 (LSB)	HW Trigger	Current level at TRIGGER Input connector
1	Cmd Ready	One or more commands waiting in Tx FIFO
2	TxFIFO Empty	Tx FIFO is empty
3	TxFIFO Error	Tx FIFO over-flow error. (If asserted, power cycle is required)
4	CmdProc Error	A Command Processor error has occurred. (If asserted, power cycle is required)
5	CmdProc Busy	Command Processor is busy processing a command
6	RxFIFO Empty	Rx FIFO is empty
7	RxFIFO Error	Rx FIFO over-flow error. (If asserted, power cycle is required)
16	Mode	Indicates whether in RFFE or SPI Mode
29	Triggered	Trigger event has occurred. (*Normal mode*: Always asserted. *Buffer mode*: Cleared until trigger event occurs then set.)
31 (MSB)	Reset Error	System is in reset. (If asserted, power cycle is required)

Table 1- Status Register Flags

3.4.5 Count Value

A Count Register value is also displayed when the 'status' command is issued. The Count value is dependent on the current operating mode:

Normal mode – Count value indicates the number of commands executed since most recent 'reset' command.

Buffer mode – Count value indicates the number of commands executed since most recent 'buffer' command.

The Count value is not reset by a trigger event. Therefore issuing commands in Normal mode after sending commands in Buffer mode will continue to increment the Count value.

3.5 LED Control

On initial power up, the Status LED on the front panel remains off for approximately 5 seconds during boot at which time it turns green to indicate Scout is functional.

Once functional, the LED state can be controlled by the user. This facilitates debugging and can be used as a user indicator during a user session. The LED is controlled using the 'led' command.

The current LED setting can be polled by issuing the 'led' command with no parameters.

3.5.1 Command Details

Syntax: LED <state>

Description: Set LED state

Supported: All Versions

Parameters: state = 0; LED disabled

state = 1; LED enabled, green (default)

Example: Disable the LED

led 0

3.6 VIO Control

The VIO pin on the DUT Interface Connector can be used to provide power to a Slave device (Internal) or to accept an external supply voltage to set the logic levels of the DUT I/O buffers (External).

The VIO power supply pin function is controlled using the 'vio' command.

The current VIO setting can be polled by issuing the 'vio' command with no parameters.

By default the Internal VIO power supply OFF/EXTERNAL.

3.6.1 **Internal**

The Internal VIO power supply voltage is user-selectable and can be set to 1.2V, 1.8V or OFF/EXTERNAL when in RFFE Mode. When in SPI mode the Internal VIO power supply voltage is user-selectable and can be set to 1.2V, 1.8V, 2.5V, 3.3V or OFF/EXTERNAL.

The Internal VIO power supply is able to source a minimum of 75mA @ 1.2V.

3.6.2 **External**

An External VIO supply voltage can be provided to power Scout's DUT I/O Interface.

When using an External VIO supply, the Internal VIO must be set to OFF/EXTERNAL.

The External VIO voltage is applied to the VIO pin of the DUT Interface Connector.

The externally applied voltage must be in the range of 1.2V to 3.3V.

3.6.3 **Command Details**

Syntax: VIO <mode>

Description: Set VIO power supply mode

Supported: All Versions

Parameters: mode = 0; OFF / EXTERNAL. Internal VIO regulator disabled (default)

mode = 1; Internal VIO regulator enabled, output set to 1.2V

mode = 2; Internal VIO regulator enabled, output set to 1.8V

mode = 3; **SPI Mode Only** Internal VIO regulator enabled, output set to 2.5V

mode = 4; **SPI Mode Only** Internal VIO regulator enabled, output set to 3.3V

Example: Enable VIO regulator for 1.2V output

vio 1

3.7 SCLK Control

The Serial SCLK signal is output on DUT IO1 pin and provides the clock signal for the serial interface.

The SCLK frequency is selectable between 52MHz, 26MHz, 13MHz and 1MHz.

The current SCLK frequency setting can be polled by issuing the 'clk' command with no parameters.

3.7.1 Command Details

Syntax: CLK <rate>

Description: Set SCLK toggle rate

Supported: Application Versions 3.01 or earlier - Rates 0 - 2
Application Versions 3.02 or later - Rates 0 - 3
Application Versions 3.05 or later - Rates 0 - 5

Parameters:
rate = 0; SCLK = 26 MHz (default)

rate = 1; SCLK = 13 MHz

rate = 2; SCLK = 1 MHz

rate = 3; SCLK = 52 MHz

rate = 4; SCLK = 38.4 MHz

rate = 5; SCLK = 19.2 MHz

Example: Set SCLK to 13MHz toggle rate

clk 1

3.8 Delay Control

The Delay control is implemented to insert a delay between commands. Delay in itself is a command that is written to the TX FIFO (Command FIFO). The delay is based on the clock frequency set by the CLK command.

3.8.1 Command Details

Syntax: Delay <cycles>

Description: Set the delay SCLK cycles

Supported: All Versions

Parameters: cycles = 0x00000000 - 0xFFFFFFFF

Example: SCLK = 1 (13 MHz), Set a Delay of 255 uSec

delay 3315

3.9 Serial Commands

The Serial commands are used for two-wire serial communication with a slave device connected to the SCLK and SDATA signals (IO1 and IO2) on the DUT Interface Connector.

Serial commands can be buffered in a FIFO using the 'buffer' command. Serial commands stored in the FIFO are sent to the DUT when a trigger event occurs.

With the release of the RFFE v2 specification, only its mandatory features such as the half-speed read operation, sReads, and extended frequency range (52 MHz) have been implemented.

COMMAND	DESCRIPTION
SET_READ	Set whether the read operation is in full-speed or half-speed
RZW	Write one 7-bit value to Register 0
RW	Write one byte to specified register address
RR	Read one byte from specified register address
ERW	Write up to 16 consecutive bytes starting at specified base register address
ERR	Read up to 16 consecutive bytes starting at specified base register address
EWL	Write up to 8 consecutive bytes starting at specified base register address
EWR	Read up to 8 consecutive bytes starting at specified base register address

Table 2- Serial Command Summary

IMPORTANT NOTE: Data returned by the Serial Read commands is stored in the Read Buffer and must be accessed using the 'read' command after the serial command has been executed.

3.9.1 Serial Read Configuration (SET_READ)

Syntax: SET_READ <mode>

Description: Set whether the read operation is in full-speed or half-speed.

Supported: Application version 3.02 or newer - RFFE Mode Only

Parameters: mode = 0; full-speed read operation (normal)

mode = 1; half-speed read operation

Example: **set_read 1**

3.9.2 Register Zero Write (RZW)

Syntax: RZW <SA[3:0]> <DATA[6:0]>

Description: Write one 7-bit value to slave at register address 0.

Supported: All Versions - RFFE Mode Only

Parameters: SA[3:0] = Slave Address

DATA[6:0] = Register Value

Example: **rzw 11 0x7F**

3.9.3 Register Write (RW)

Syntax: RW <SA[3:0]> <ADDR[4:0]> <DATA[7:0]>

Description: Write one byte to slave at specified register address.

Supported: All Versions - RFFE Mode Only

Supported: RFFE Mode Only

Parameters: SA[3:0] = Slave Address

ADDR[4:0] = Register Address

DATA[7:0] = Register Value

Example: **rw 11 0x1F 0xFF**

3.9.4 Register Read (RR)

Syntax: RR <SA[3:0]> <ADDR[4:0]>

Description: Read one byte from slave at specified register address.

Supported: All Versions - RFFE Mode Only

Parameters: SA[3:0] = Slave Address

ADDR[4:0] = Register Address

Example: Read the value of register 0x1F from Slave 11.

Example: **rr 11 0x1F**

3.9.5 Extended Register Write (ERW)

Syntax: ERW <SA[3:0]> <BC[3:0]> <ADDR[7:0]> <DATA{0}> … <DATA{*BC*}>

Description: Write up to 16 bytes to slave starting at specified register address.

Supported: All Versions - RFFE Mode Only

Parameters: SA[3:0] = Slave Address

BC[3:0] = Byte Count (*Note: number of bytes written is BC + 1*)

ADDR[7:0] = Register Address

DATA{0:*BC*} = Register Values

Example: Write 8 bytes, 0x01 thru 0x08, to Slave 11 starting at register address 0x10.

erw 11 0x07 0x10 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08

3.9.6 Extended Register Write (ERR)

Syntax: ERR <SA[3:0]> <BC[3:0]> <ADDR[7:0]>

Description: Read up to 16 bytes from slave starting at specified register address.

Supported: All Versions - RFFE Mode Only

Parameters: SA[3:0] = Slave Address

BC[3:0] = Byte Count (*Note: number of bytes read is BC + 1*)

ADDR[7:0] = Register Address

Example: Read 16 bytes from Slave 11 starting at register address 0x00.

err 11 0x0F 0x00

3.9.7 Extended Register Write Long (EWL)

Syntax: EWL <SA[3:0]> <BC[2:0]> <ADDR[15:8]> <ADDR[7:0]> <DATA{0}> … <DATA{*BC*}>

Description: Write up to 8 bytes to slave starting at specified extended register address.

Supported: All Versions - RFFE Mode Only

Parameters: SA[3:0] = Slave Address

BC[2:0] = Byte Count (*Note: number of bytes written is BC + 1*)

ADDR[15:8] = Register Address MSB

ADDR[7:0] = Register Address LSB

DATA{0:*BC*} = Register Values

Example: Write 8 values, 0x01 thru 0x08, to Slave 11 starting at register address 0x1234.

erw 11 0x07 0x12 0x34 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08

3.9.8 Extended Register Read Long (ERL)

Syntax: ERL <SA[3:0]> <BC[2:0]> <ADDR[15:8]> <ADDR[7:0]>

Description: Read up to 8 bytes to slave starting at specified extended register address.

Supported: All Versions - RFFE Mode Only

Parameters: SA[3:0] = Slave Address

BC[2:0] = Byte Count (*Note: number of bytes read is BC + 1*)

ADDR[15:8] = Register Address MSB

ADDR[7:0] = Register Address LSB

DATA{0:*BC*} = Register Values

Example: Read 4 values from Slave 2 starting at register address 0x8000.

err 2 0x03 0x80 0x00

3.10 GPIO Commands

The GPIO commands are used to control the general-purpose DUT Interface Connector I/O signals IO3 thru IO8. The GPIO commands are only supported in RFFE Mode.

IMPORTANT NOTE: IO1 and IO2 are reserved for use as the Serial SCLK and SDATA signals respectively and are therefore not directly controllable by the user. The GPIO commands have no effect on these signals.

COMMAND	DESCRIPTION
GDW	Set GPIO direction (0 = INPUT, 1 = OUTPUT)
GDR	Read GPIO direction (0 = INPUT, 1 = OUTPUT)
GPW	Set GPIO level (0 = GND, 1 = VIO)
GPR	Read GPIO level (0 = GND, 1 = VIO)

Table 3- GPIO Command Summary

3.10.1 I/O Usage

For software control, each DUT I/O signal is mapped to one bit of an eight-bit word as shown in the following table.

DUT I/O	FUNCTION	BIT	VALUE
IO1	SCLK	0	0x01
IO2	SDATA	1	0x02
IO3	GPIO3	2	0x04
IO4	GPIO4	3	0x08
IO5	GPIO5	4	0x10
IO6	GPIO6	5	0x20
IO7	GPIO7	6	0x40
IO8	GPIO8	7	0x80

Table 4- DUT I/O Bit Map

3.10.2 I/O Default State

On power-up, the DUT I/O signal directions and levels are defaulted as shown in the following table.

I/O	POWER-UP DIRECTION (STATE)
SCLK (IO1)	Output (low)

SDATA (IO2)	Input (internally pulled low)
IO3-IO8	Output (low)

Table 5- I/O Power-up Defaults

3.10.3 GPIO Direction Write (GDW)

Syntax: GDW <mask> <value>

Description: Set GPIO directions

Supported: All Versions - RFFE Mode Only

Parameters: mask: Specifies the I/Os to update (0 = ignore, 1 = update)

value: Specifies the I/O directions (0 = INPUT, 1 = OUTPUT)

Example: Set IO4 as output, IO8 as input and leave others unchanged

gdw 0x88 0x08

3.10.4 GPIO Direction Read (GDR)

Syntax: GDR

Description: Read GPIO directions. (Read-back value is stored in Read Buffer).

Supported: All Versions - RFFE Mode Only

Parameters: None

Example: Read current I/O directions

Gdr

3.10.5 GPIO Write (GPW)

Syntax: GPW <mask> <value>

Description: Set GPIO levels

Supported: All Versions - RFFE Mode Only

Parameters: mask: Specifies the I/Os to update (0 = ignore, 1 = update)

value: Specifies the I/O levels (0 = GND, 1 = VIO)

Example: Set IO3, IO4 and IO5 high and all others low.

gpw 0xFF 0x1C

3.10.6 GPIO Read (GPR)

Syntax: GPR

Description: Read GPIO levels. (Read-back value is stored in Read Buffer)

Supported: All Versions - RFFE Mode Only

Parameters: None

Example: Read current I/O levels

Gpr

3.10.7 GPIO Examples

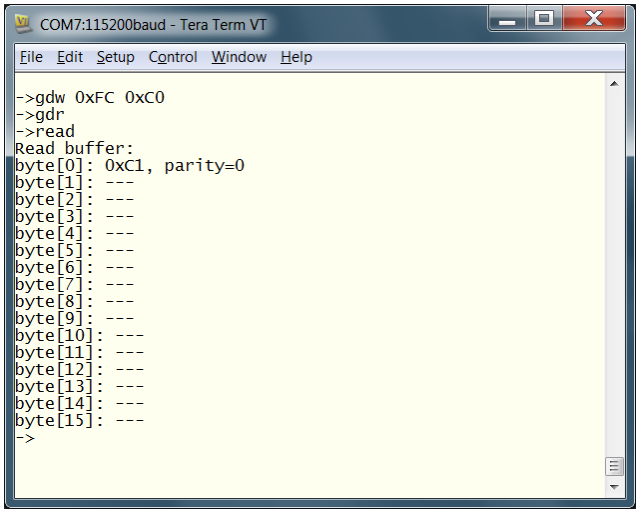

Figure 9 - Example GPIO Direction Write and Read Commands

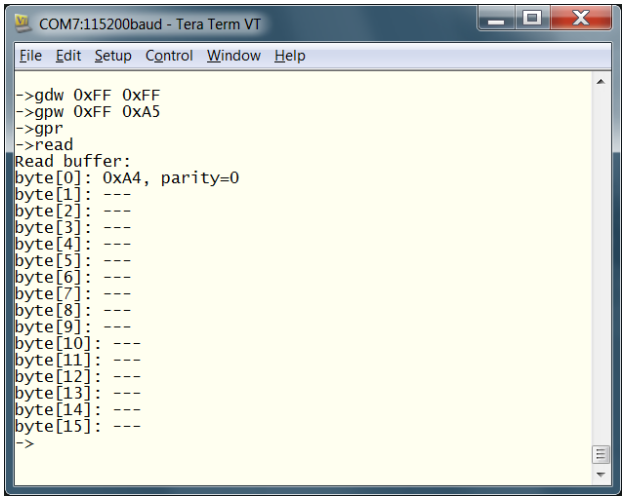

Figure 10- Example GPIO Write and Read Commands

Note: The 'parity' value displayed in the Read buffer is not valid for the GPIO commands and should be ignored.

3.11 SPI Commands

The SPI commands are used for 4 -wire serial communication with a slave device connected to the SCLK, MOSI, MISO and one of the slave select signals (SS1 – SS5). All of the SPI commands with the exception of SPI_WR and SPI_RD are SPI configuration that is determined by the DUT slave device.

SPI commands can be buffered in a FIFO using the 'buffer' command. Serial commands stored in the FIFO are sent to the DUT when a trigger event occurs.

COMMAND	DESCRIPTION
SPI_MODE	Change Scout Personality Mode (RFFE or SPI)
SPI_STATUS	Display the SPI Status Register
SPI_CFG	Configure the SPI Interface & Protocol
SPI_WR	Perform single byte or burst write to the specified address (starting address for burst)
SPI_RD	Perform single byte or burst read from the specified address (starting address for burst)
SPI_SELECT	Select one of the SPI Slave Selects
SPI_POL	Set the Polarity of the Slave Select to Active Low or Active High
SPI_CLOCK	Set the SPI SCLK Rate (200 kHz – 26 MHz)
SPM	Set the SPI Mode (supports Mode 0 – Mode 3)
COM_WIDTH	Define the Width of the Command Word (determined by the slave device)
ADD_WIDTH	Define the Width of the Address (determined by the slave device)
WR_WIDTH	Define the Width of the Write Data Word (determined by the slave device)
RD_WIDTH	Define the Width of the Read Data Word (determined by the slave device)
SPIWC	Set the Required Number of Wait Cycles (determined by the slave device)

Table 6- SPI Command Summary

3.11.1 SPI Configuration

For software control, all of the configuration can be completed with a single the SPI_CFG command. Subsequent modifications to the configuration can be completed with a configuration attribute command, i.e. change SPI clock rate can be altered with the SPI_CLOCK command which alters the clock rate but leaves all other attributes unchanged.

3.11.2 SPI Mode (SPI_MODE)

Syntax: SPI_MODE <mode>

Description: Set Scout to RFFE or SPI Mode

Supported: Application Version 2.10 or newer

Parameters: Mode: 0=RFFE (default), 0xCF=SPI

Example: Set Scout to SPI Mode

SPI_MODE 0xCF

3.11.3 Display SPI Status (SPI_STATUS)

Syntax: SPI_STATUS

Description: Displays the SPI Status Register

Supported: Application Version 2.10 or newer

Parameters: None

Example: Display SPI Status

SPI_STATUS

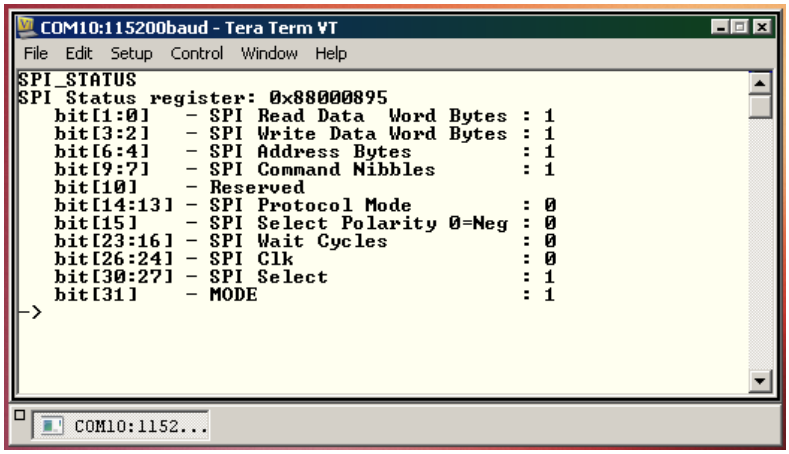

Figure 11- Example SPI STATUS

3.11.4 SPI Configuration (SPI_CFG)

Syntax: SPI_CFG <SPI_SELECT[3:0]> <CLK_RATE[2:0] <SPI_SELECT_POLARITY[0]> <SPI_MODE[1:0]> <COM_WIDTH_BITS[4:0]> <ADDR_WIDTH_BITS[5:0]> <WR_WIDTH[1:0]> <RD_WIDTH[1:0]> <WAIT_CYCLES[7:0]>

Description: Configure the SPI Interface & Protocol to match DUT

Supported: Application Version 2.10 or newer - SPI Mode Only

Parameters: Refer to following commands for each of the parameters

3.11.5 SPI Write Data (SPI_WR)

Syntax: SPI_WR <BURST[7:0]> <COMMAND[15:0] <ADDRESS[31:0]> <DATA0[31:0]> - <DATA5[31:0]>

Description: Perform Single or Burst Write Operation. A minimum of DATA0 is required, for burst operation Data2 – Data5 are utilized. If there is not enough data provided for a burst operation the defined data will be repeated to perform the burst operation.

Supported: Application Version 2.10 or newer - SPI Mode Only

Parameters:

Burst: If Burst = 0 the operation is a single byte write otherwise burst = the number of data bytes + 1 in burst.

Command: Determined by the SPI Slave DUT, command for write or burst write.

Address: SPI Slave DUT register address.

DATA: Data to be written to the SPI address.

Example:

Configuration: Command 4-bits

Address 8-bits

Write Data Width 1 Byte

Single Byte Write – write 0XDE to address 0x3A

SPI_WR 0x00 0x01 0x3A 0xDE

Burst Write of 7 Bytes – write data to starting address 0x1A

SPI_WR 0x06 0x11 0x1A 0x12345678 0x9ABCDE

Address	Data
0x1A	0x78
0x1B	0x56
0x1C	0x34
0x1D	0x12
0x1E	0xDE
0x1F	0xBC
0x20	0x9A

Notes: When performing a burst operation, if there isn't enough data provided the provided data pattern will repeat.

3.11.6 SPI Read Data (SPI_RD)

Syntax: SPI_WR <BURST[7:0]> <COMMAND[15:0] <ADDRESS[31:0]>

Description: Perform Single or Burst Read Operation

Supported: Application Version 2.10 or newer - SPI Mode Only

Parameters:

Burst: If Burst is 0 the operation is a single byte read. Number of data bytes in burst.

Command: Determined by the SPI Slave DUT, command for read or burst read.

Address: SPI Slave DUT register address.

Example:

Configuration: Command 1 Nibble

Address 8-bits

Write Data Width 1 Byte

Single Byte Read – read the contents of address 0x17

SPI_RD 0x00 0x00 0x17

The results of the SPI read are stored in the RX_FIFO, refer to READ command to retrieve data.

3.11.7 SPI Slave Device Select (SPI_SELECT)

Syntax: SPI_SELECT <SPI_SELECT[3:0]>

Description: Choose SPI Slave Select (SS1 – SS5)

Supported: Application Version 2.10 or newer - SPI Mode Only

Parameters: SPI_SELECT 0x0 = No Device Select active
0x1 = Select line SS1 Active
0x2 = Select line SS2 Active
0x3 = Select line SS3 Active
0x4 = Select line SS4 Active
0x5 = Select line SS5 Active

Example: Set SPI Device Select to SS3

Spi_select 0x3

3.11.8 SPI Slave Select Polarity (SPI_POL)

Syntax: SPI_POL <SPI_SELECT_POLARITY[0]>

Description: Set SPI Slave Select Polarity (Active Low/ Active High)

Supported: Application Version 2.10 or newer - SPI Mode Only

Parameters: SPI_SELECT_POLARITY 0x0 = Active Low
0x1 = Active High

Example: Set SPI Device Select to Active High

SPI_POL 0x1

3.11.9 SPI SCLK RATE (SPI_CLOCK)

Syntax: SPI_CLOCK <CLK_RATE[2:0]>

Description: Set SPI SCLK Frequency

Supported: Application Version 2.10 or newer - SPI Mode Only

Parameters: CLK_RATE: SPI Clock Select the clock divisor of the SPI base rate of 26 MHz
0x0 = 26.000 MHz SPI Clock Rate
0x1 = 13.000 MHz SPI Clock Rate
0x2 = 6.250 MHz SPI Clock Rate
0x3 = 3.125 MHz SPI Clock Rate
0x4 = 1.625 MHz SPI Clock Rate
0x5 = 0.813 MHz SPI Clock Rate
0x6 = 0.406 MHz SPI Clock Rate
0x7 = 0.203 MHz SPI Clock Rate

Example: Set SPI SCLK Frequency to3.125 MHz

SPI_CLOCK 0x3

3.11.10 SPI PROTOCOL MODE (SPM)

Syntax: SPM <SPI_PROTOCOL_MODE[1:0]>

Description: Set SPI Protocol Mode

Supported: Application Version 2.10 or newer - SPI Mode Only

Parameters: SPI_PROTOCOL_MODE:
0x0 = MODE 0
0x1 = MODE 1
0x2 = MODE 2
0x3 = MODE 3

Example: Set SPI Protocol Mode to Mode 0

SPM 0

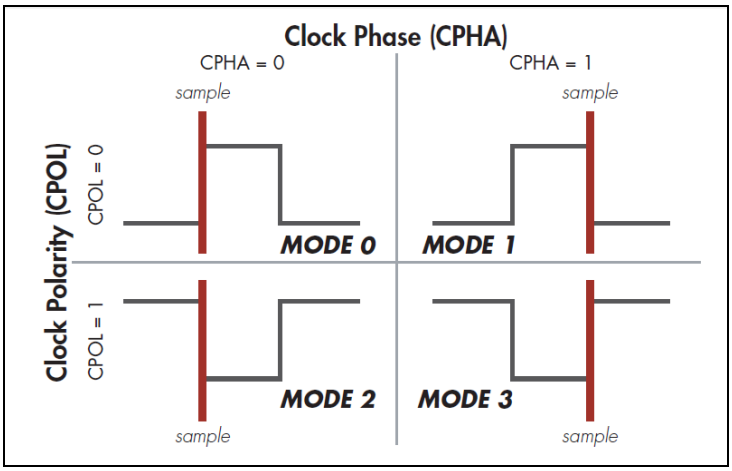

Figure 12- SPI Modes

3.11.11 SPI COMMAND WORD WIDTH (COM_WIDTH)

Syntax: COM_WIDTH <COM_WIDTH_BITS[4:0]>

Description: Set SPI Command word width

Supported: Application Version 3.01 or newer - SPI Mode Only

Parameters: COM_WIDTH_BITS:
Value represented in # of bits from 0 to 16

Example: Set SPI Command width to 4-bits

COM_WIDTH 0x04

3.11.12 SPI ADDRESS WIDTH (ADD_WIDTH)

Syntax: ADD_WIDTH < ADDR_WIDTH_BITS[5:0]>

Description: Set SPI Address Width

Supported: Application Version 3.02 or newer - SPI Mode Only

Parameters: ADDR_WIDTH_BYTES:
Value represented in # of bits from 0 to 32

Example: Set SPI Address Width to 16-bits

ADD_WIDTH 0x10

3.11.13 SPI WRITE DATA WIDTH (WR_WIDTH)

Syntax: WR_WIDTH <WR_DATA_WIDTH[1:0]>

Description: Set SPI Write Data Width

Supported: Application Version 2.10 or newer - SPI Mode Only

Parameters: WR_DATA_WIDTH:
Value represented in # of bytes [0, 1, 2, or 4]

Example: Set SPI Write Data Width to 4 Bytes

WR_WIDTH 0x03

3.11.14 SPI READ DATA WIDTH (RD_WIDTH)

Syntax: RD_WIDTH <RD_DATA_WIDTH[1:0]>

Description: Set SPI Read Data Width

Supported: Application Version 2.10 or newer - SPI Mode Only

Parameters: RD_DATA_WIDTH:
Value represented in # of bytes [0, 1, 2, or 4]

Example: Set SPI Read Data Width to 8-bits

RD_WIDTH 0x01

3.11.15 SPI READ WAIT CYCLES (SPIWC)

Syntax: SPIWC <WAIT_CYCLES[7:0]>

Description: SPI Read Wait Cycles To Be Inserted Between Address and Read Data if Required.
Valid Data is from 0 - 255 sclk wait cycles

Supported: Application Version 2.10 or newer - SPI Mode Only

Parameters: WAIT_CYCLES:
0x00 – 0xFF

Example: Set SPI Read Wait Cycles to 2-wait cycles

SPIWC 0x02

3.12 Operating Mode

3.12.1 Normal Mode

Scout defaults to Normal mode on power-up.

In Normal mode, all commands are executed immediately as they are entered. The '->' command prompt indicates Normal mode operation.

3.12.2 Buffer Mode

Scout may be placed into Buffer mode using the 'buffer' command. Scout supports two buffers, the primary and an alternate buffer. The 'B>' command prompt indicates Primary Buffer mode operation and The 'b>' command prompt indicates Alternate Buffer mode operation. Commands written to the Primary Buffer are executed first in Software trigger mode or on a rising edge trigger when in hardware triggered mode. Similarly, Commands written to the Alternate Buffer are executed following the commands in the Primary Buffer or on a falling edge trigger when in hardware triggered mode. Each buffer can hold 127 commands providing enough buffer space to buffer 254 commands in software triggered mode or 127 positive edge triggered and 127 negative edge triggered commands in hardware triggered mode.

IMPORTANT NOTE: When in software triggered mode there must be at least 1 command in the primary buffer before commands in the alternate buffer can be triggered by software. The general rule for software triggered mode is to always write 1 dummy command to the primary buffer (i.e. delay 0 – zero delay) then commands cam be palced in either the primary or alternate buffers.

When in Buffer mode Serial and GPIO commands are written to a FIFO, and not executed until the 'trigger' command is issued and a trigger event occurs.

IMPORTANT NOTE: All non-Serial and non-GPIO commands are executed immediately and cannot be buffered.

3.13 Triggering

Buffered command execution (FIFO Dump) is triggered by a Software or Hardware trigger event.

3.13.1 Software Trigger

A software trigger event occurs when the user issues the 'trigger 1' command.

The software trigger causes command execution to begin immediately and all commands are executed first from the primary buffer and then from the alternate buffer in the order they were buffered.

On completion of the trigger event, the Count value in the Status register will indicate the total number of commands executed from the FIFO and the Triggered flag will transition from low to high.

3.13.2 Hardware Trigger

A hardware trigger is configured when the user issues the 'trigger 2'.

The hardware trigger causes Scout to monitor the Trigger Input connector for a valid trigger event. A valid hardware trigger event is a rising-edge signal transition from 0V to VIO on the Trigger Input connector.

Once a positive edge trigger event occurs, command execution begins immediately and all commands are executed in the order they were buffered in the primary buffer. Similarly, when a negative edge trigger event occurs, command execution begins immediately and all commands are executed in the order they were buffered in the alternate buffer.

A hardware trigger can be configured to repeat when the user issues the command ‘trigger 2 1’ which sets the repeat bit. The Repeat feature has been implemented to allow the same command sets to be executed repeatedly until stopped by the user. For example, if Repeat mode was enabled with Hardware Trigger mode, if both FIFOs had at least one command, once the 'trigger' command is issued Scout waits for the first rising edge, then sends commands in the Primary FIFO. Once all Primary FIFO commands have completed, Scout waits for the next falling edge then sends the commands in the Alternate FIFO. Then because Repeat mode is enabled, Scout would wait for the next rising-edge and send the same Primary FIFO commands again, and so on. The user must issue a RESET command to halt the system and clear the FIFO contents. The Repeat Mode must be enabled by the user, which sets the Repeat Mode bit in the Trigger Control & Status register

The user may issue the ‘reset’ command to abort the Hardware trigger. The ‘reset’ command causes Scout to cease monitoring the Trigger Input and clear all commands from the buffer

On completion of the trigger event, the Count value in the Status register will indicate the total number of commands executed from the FIFO and the Triggered flag will transition from low to high.

Note that the REPEAT function is not valid for the software trigger mode.

3.13.3 **Command Details**

Syntax:	trigger <mode>
Description:	Set up trigger to execute commands when in Buffer mode
Parameters:	mode = 1; Software trigger. Buffered commands are executed immediately.
Example:	mode = 2; Hardware trigger. Buffered commands are executed when a trigger event (rising edge) occurs on the Trigger Input. On the next rising edge at the Trigger Input connector, execute the commands that have been buffered **trigger 2**

3.14 Read Buffer

Scout uses a 16-byte Read Buffer to store data from the most recently executed read commands when in RFFE mode. When operating in SPI_Mode personality Scout uses a 32-byte Read Buffer to store data from the most recently executed read commands.

The Read Buffer contents are displayed by issuing the ‘read’ command.

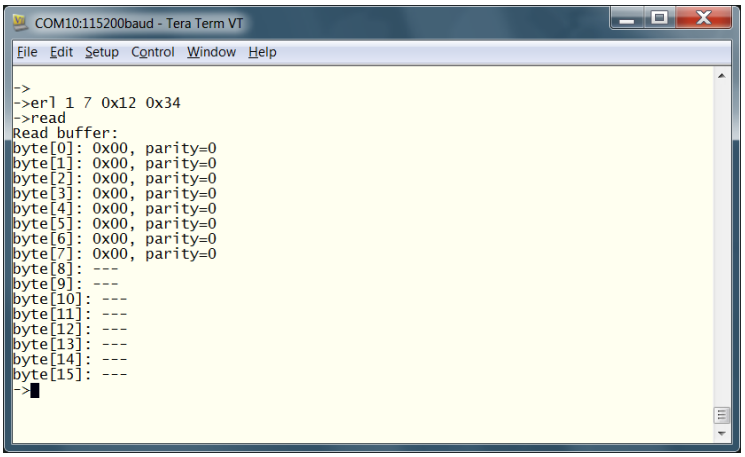

Figure 13 - Read Buffer contents - Extended Read Long command

Operation of the Read Buffer is dependent on the current operating mode.

Normal mode – The Read Buffer contains only the results of the most recently issued read command (i.e. previous read command results are over written).

Buffer mode – The Read Buffer contains the most recent 16 bytes read from the slave. For example, if 16 single-byte read commands were buffered and executed, the Read Buffer would contain all sixteen bytes.

When the Read Buffer contains less than 16 bytes, the '---'symbol indicates empty buffer locations.

Bytes are displayed in the order they were read, with byte[0] representing the oldest read value and byte [15] the newest.

4. Connectivity

4.1 Alternate Power Connector

The 2-pin connector allows Scout to be powered from an external 5V power supply. This may be required when Scout is connected to a USB port that cannot source 500mA, such as on an unpowered USB hub.

4.1.1 **Interface Specs**

Supply Voltage: +5VDC +/-5%. Damage *may occur if limits are exceeded.*

Supply Current (min.): 500mA

4.1.2 **Mating Connector**

Manufacturer: Phoenix Contact

Part Number: 1881325

4.2 USB Connector

The USB connector provides the Host PC-to-Scout control interface. It also normally provides power to the unit.

4.2.1 **Interface Specs**

Supply Voltage: +5V (VBUS)

Supply Current (min): 500mA

Interface: USB2.0

Connector: Type B

4.3 Ethernet Connector

The Ethernet connector is not currently support and is intended for future applications

IMPORTANT NOTE: Connecting the Ethernet port may cause unreliable operation of Scout.

4.4 Trigger Input Connector

The Trigger Input is a high-impedance, logic-level input. ***It is not a 50-ohm input***.

4.4.1 Interface Specs

Input Voltage Limits: -0.5V (min), +4.6V (max). *Damage may occur if limits are exceeded.*

Connector: BNC

4.5 DUT Interface Connector

The 12-pin DUT Interface connector provides eight user-controllable General Purpose I/O pins, one VIO power connection and multiple Ground connections.

All DUT Interface signals use VIO signaling levels.

4.5.1 Interface Specs

4.5.1.1 VIO (Internal):

Supply Voltage: 1.2V, 1.8V (User-selectable)

Current Sourcing Capacity: 75mA (min) @ 1.2V

4.5.1.2 VIO (External):

Supply Voltage: 1.2V (min), 3.3V (max). (VIO setting = OFF/EXTERNAL). *Damage may occur if limits are exceeded.*

Current Consumption: 20mA (max) @ 1.2V

4.5.1.3 DUT I/O:

Input Voltage Limits: -0.5V (min), +4.6V (max). *Damage may occur if these limits are exceeded.*

Output Voltage: Low = 0V, High = VIO

4.5.2 Mating Connector

Manufacturer: Phoenix Contact

Part Number: 1881422

4.5.3 Pin-out RFFE Mode

PIN	SIGNAL	DESCRIPTION
1	VIO	VIO power supply input / output
2	IO1	Serial SCLK signal
3	GND	Ground
4	IO2	Serial SDATA signal

5	IO3	General-purpose I/O #3
6	GND	Ground
7	IO4	General-purpose I/O #4
8	IO5	General-purpose I/O #5
9	IO6	General-purpose I/O #6
10	GND	Ground
11	IO7	General-purpose I/O #7
12	IO8	General-purpose I/O #8

Table 7 - DUT Interface Connector Pin-out

4.5.4 **Pin-out SPI Mode**

PIN	SIGNAL	DESCRIPTION
1	VIO	VIO power supply input / output
2	IO1	Serial SCLK signal
3	GND	Ground
4	IO2	Slave Select SS1
5	IO3	Slave Select SS2
6	GND	Ground
7	IO4	Slave Select SS3
8	IO5	Slave Select SS4
9	IO6	Slave Select SS5
10	GND	Ground
11	IO7	MOSI (Serial Data Out)
12	IO8	MISO (Serial Data Inrff)

Table 9 - DUT Interface Connector Pin-out

5. Firmware Update

The following procedure provides instructions on how to update the Scout firmware.

Note: Failure to follow the exact steps in this procedure can cause Scout to become non-functional!

ZMODEM protocol is used to transfer the new firmware image file from the host PC to the Scout unit. A terminal emulator that has ZMODEM file transfer capability must be used (e.g. Tera term). ZMODEM Binary mode must be enabled for the file transfer.

Depending on the size of the image, the programming operation may take up to 60 seconds to complete. The programming operation must not be interrupted during this time or the image will most likely be corrupted and the Scout unit will no longer be functional. If this occurs, contact support@signalcraft.com.

5.1 Firmware Update Procedure

Before proceeding with updating check the version to ensure the Platform (either 7010 or 7020) matches the update file set which should have either 7010 or 7020 in the filename. Note that if there is not a Platform field or the file set does not contain 7010 or 7020 in the name this indicates a 7020 variant.

Step 1: With Scout powered up, issue the 'fwupdate' command. A Firmware Update menu is displayed.

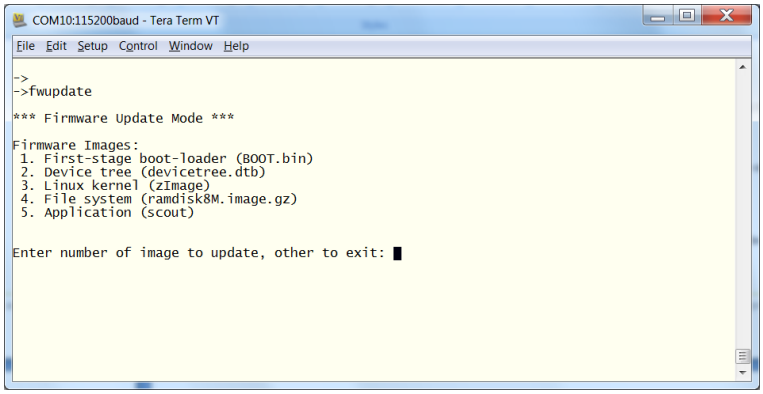

Figure 14 - Firmware Update Mode Menu

Step 2: Enter '5' at the prompt to update the Scout firmware.

Scout will respond with:

Use ZMODEM and select 'scout' image to transfer.

**B0100000023be50

Step 3: In the Host PC terminal program, use the ZMODEM file transfer capability to browse to the new Scout firmware image and begin the transfer.

On Tera Term, this is found under File>Transfer>ZMODEM>Send

Step 4: Wait for the transfer to complete.

Step 5: If the file transfer is successful, Scout will automatically begin reprogramming the new firmware.

Step 6: Wait for the programming to complete. ***DO NOT INTERRUPT THE PROCESS AT THIS POINT!***

On successful completion of the upgrade process, you will be notified of such and be presented with the FW Update mode menu. If you are done with the updates, press 'ENTER' to exit the upgrade mode.

Step 7: Cycle power to the unit.

6. Appendix A: Using SCOUT with a Linux Machine

If the host machine is running Linux, Scout can be used in a similar fashion to that described in this manual. In lieu of a Windows USB CDC generic driver, a Linux Host ACM driver is used instead.

```
Installing the Linux Host ACM Driver
------------------------------------
To use the Linux ACM driver you must configure the Linux host side
kernel for "Support for Host-side USB" and for "USB Modem (CDC ACM)
support".

Once the gadget serial driver is loaded and the USB device connected
to the Linux host with a USB cable, the host system should recognize
the gadget serial device.  For example, the command

  cat /proc/bus/usb/devices

should show something like this:

T:  Bus=01 Lev=01 Prnt=01 Port=01 Cnt=02 Dev#=  5 Spd=480 MxCh= 0
D:  Ver= 2.00 Cls=02(comm.) Sub=00 Prot=00 MxPS=64 #Cfgs=  1
P:  Vendor=0525 ProdID=a4a7 Rev= 2.01
S:  Manufacturer=Linux 2.6.8.1 with net2280
S:  Product=Gadget Serial
S:  SerialNumber=0
C:* #Ifs= 2 Cfg#= 2 Atr=c0 MxPwr=  2mA
I:  If#= 0 Alt= 0 #EPs= 1 Cls=02(comm.) Sub=02 Prot=01 Driver=acm
E:  Ad=83(I) Atr=03(Int.) MxPS=   8 Ivl=32ms
I:  If#= 1 Alt= 0 #EPs= 2 Cls=0a(data ) Sub=00 Prot=00 Driver=acm
E:  Ad=81(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=02(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms

If the host side Linux system is configured properly, the ACM driver
should be loaded automatically.  The command "lsmod" should show the
"acm" module is loaded.


Installing the Linux Host Generic USB Serial Driver
---------------------------------------------------
To use the Linux generic USB serial driver you must configure the
Linux host side kernel for "Support for Host-side USB", for "USB
Serial Converter support", and for the "USB Generic Serial Driver".

Once the gadget serial driver is loaded and the USB device connected
to the Linux host with a USB cable, the host system should recognize
the gadget serial device.  For example, the command

  cat /proc/bus/usb/devices

should show something like this:

T:  Bus=01 Lev=01 Prnt=01 Port=01 Cnt=02 Dev#=  6 Spd=480 MxCh= 0
D:  Ver= 2.00 Cls=ff(vend.) Sub=00 Prot=00 MxPS=64 #Cfgs=  1
P:  Vendor=0525 ProdID=a4a6 Rev= 2.01
S:  Manufacturer=Linux 2.6.8.1 with net2280
S:  Product=Gadget Serial
S:  SerialNumber=0
```

```
C:* #Ifs= 1 Cfg#= 1 Atr=c0 MxPwr=  2mA
I:  If#= 0 Alt= 0 #EPs= 2 Cls=0a(data ) Sub=00 Prot=00 Driver=serial
E:  Ad=81(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=02(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms

You must explicitly load the usbserial driver with parameters to
configure it to recognize the gadget serial device, like this:

  modprobe usbserial vendor=0x0525 product=0xA4A6

If everything is working, usbserial will print a message in the
system log saying something like "Gadget Serial converter now
attached to ttyUSB0".
```