Blade Runner System Operators Manual

Author: Product Development

Revision 0.2

10/11/2005

DRAFT

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLE

1 System Architecture

The BladeRunner Wireless System is designed to bridge or backhaul Voice-TDM, Wireless-CDMA, or Data traffic for a large variety of carrier class service providers. In its most fundamental configuration a bi-directional wireless link is facilitated between two nodes in a point to point architecture.

Due to the complexity of the system the PC701 must be professionally installed.

1.1 Targeted Network Applications

- A basic PTP backhaul platform
- Multiple Off the Shelf (OTS) user Interfaces
- Frequency agility (independent RF)
    - *5.8GHz*
- RF Blade Runner Modular Server Architecture
    - *C-OFDM (NLOS)*
    - *Single carrier (LOS)*
- Supports concurrent operations
    - *Integrated Micro cells*
    - *Fixed Local Loop (TelCo grade)*
    - *HotSpot backhaul*
- Cost effective enough for SOHO/MDU market

1.2 Point -To -Point Operation

Transmitting and receiving of the OFDM signals, carried over the 5.8GHz band in minimum configuration ,requires the use of the following modular components.

1. Two Antenna
2. Two Digital Interface Transceiver Devices (DITs)
3. Two OFDM Modem cards
4. Minimum of two carrier cards
5. Two Tributary cards (T1/E1)
6. Two host cards
7. Two power supplies

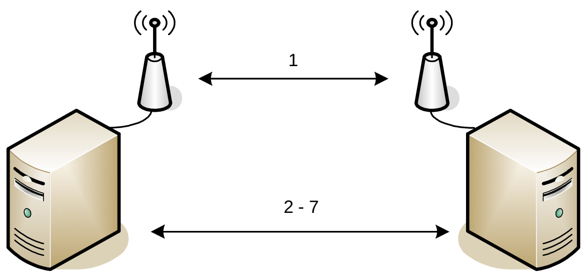


Figure 1 Point-to-Point configuration

2 Equipment Start up

2.1.1 System Boot Parameters and Configuration

The WHYS7020 boot parameters are maintained in the hard drive of HOST CARD. They comprise the following:

- System IP address, netmask, default gateway
- User name and password information
- Source for the software image that is to run in HOST CARD
- Console baud rate definition

2.1.2 File System

WHWYS7020 is equipped with regular hard drive running Linux RTOS. All the program and data are stored in standard Linux file formats:

1. Images for the processor on the Host Card hard drive
    a. HOST CARD can maintain several copies of the Linux code images.
    b. Each code-image file will have a header that has the following fields:
        i. Size
        ii. Version
        iii. Date
        iv. Checksum
2. Saved system startup configuration
    a. Several (TBD) files may be used to save the system configuration
    b. The files may be compressed to save space
    c. The files are protected from corruptions by checksum
3. Fatal logs

2.1.3 System Initialization

For the processors equipped with flash (i.e. PCI/SBE CPUs), a portion of that also acts as a boot-ROM, which should be write-protected. Upon power-on or reset of a board, control is given to the ROM code, which will be responsible for loading the operational code and transferring control to it. The ROM code in the control processor of the PCI/SBE will support the following functionality:

- Diagnostics on the hardware controlled by it. The scope is defined such that the CPU in each board controls all the hardware excepting the ones that are directly controlled by some other processor in the board. For example, the RF mezzanine may have its own CPU, rather than the PCI/SBE CPU, to control the RF hardware.
- Initialization of the board hardware. This could involve initializing the different hardware devices, booting the FPGAs, booting other devices like co-processors, Ethernet MAC, etc.
- Initializing the UART and making it operational, if applicable.
- Transfer of control from ROM image to the executable RAM software image

For the processors with the hard drive and no Flash for the OS image, it just initializes like a standard PC

2.1.4 HOST CARD Initialization

First, the HOST CARD will go through the standard Linux power-up initialization sequence, since the hardware is Pentium based standard PC motherboard. Device drivers for the non-transparent PCI-to-PCI Bridge interface will be installed during the initialization process and build the communication channels for management messages and PPP/SLIP links.

After the Linux initialization is done, control will be transferred to ioWave tasks, which will perform the following actions:

1. Initialize the hardware and software components of the board – this includes system software modules, which are not part of the Linux OS.
2. If failure or any errors detected in above steps, display error message onto the UART console and reset

3. Check the hard drive for system boot configuration and obtain the NMS IP address and other mandatory configuration parameter in the system boot configuration. If this information is not available, display the message in console and continue the initialization process.
4. If failure or any errors detected in above steps, display error message onto the debug port. The error message can later be displayed by "dmesg" command.
5. Obtain the system boot configuration from hard drive configuration files.
6. If configuration is to be restored, read the required configuration files from hard drive or from remote and restore them. If any errors occur in this process, generate an error message on debug port as well as a trap to NMS, disable the configuration restoration, by writing in the system boot configuration area, and reset. The board will come up with default configuration after this.
7. Give enough time delay (TBD) so that all PCI/SBEs are ready to be probed.
8. Probe the PCI/SBEs and enable the PCI shared memory interface stack so that message from PCI/SBE boards can be received and processed.
9. After this, the HOST CARD board is completely operational.
10. The HOST CARD module can now communicate with all PCI/SBE boards. Following are some of the messages exchanged:
    - Software download from HOST CARD to PCI/SBE. The operator manually controls the download process.
    - Keep-alive messages.
    - Board status messages.
    - Alarm (or trap) messages from other boards indicating errors on the board becoming operational. HOST CARD will take action based on the message.

The operator can also do configuration and manage the system. In particular, the system administrator can perform the following operations related to system initialization:

- Initiate download of files from external file server by specifying the server IP address and filename to download. The corresponding file will be downloaded into hard drive file system.
- Configure board parameters for PCI/SBE boards, including SLI table entries, specifying the slot number and board number of the board. These parameters include whether the board should download and execute code from its flash or from HOST CARD, the filename in case code is to be downloaded and whether any general configuration should be restored for the board when it comes up.
- Reset any PCI/SBE board in the system

2.1.5 Initialization of PCI/SBE boards

After a power-on or reset, all the non-HOST CARD boards begin to execute from ROM. Following are the sequence of steps performed by the ROM code of these boards:

1. Perform power-on (diagnostic) tests on the board hardware if required. This is done upon a cold start only.
2. Initialize the hardware and software components of the board – this includes setting up of interrupt tables, initialization of memory modules, initialization of the OS (if there is one), and other system software modules, PCI shared memory interface (for IBC), Flash memory, asynchronous port as well as the software stacks.
3. If failure or any errors detected in above steps, display error message onto the debug port and reset. A terminal or console should be connected to the debug port to observe these messages.
4. Copy the contents of flash from the predefined location to RAM. The header of the file in flash will give information on the size and version of the software. If any errors during this process, generate a trap/fault message onto debug port.
5. Transfer control to the start of the RAM code for the operational code to initialize.

After the control has been transferred from ROM to RAM, the following actions are performed by the RAM code of the non-HOST CARD boards:

1. Initialize the hardware and software components of the board – this is specific to the board and includes setting up of interrupt tables, initialization of memory modules, initialization of the OS and other system software modules, Ethernet interfaces, flash memory, asynchronous port as well as the protocol software residing on this board.
2. If failure or any errors detected in above steps, display error message onto the debug port and reset.
3. Wait the HOST CARD to probe this board. Then complete the handshaking and initialize the PCI shared memory interface. If everything passes, go into the normal operation mode, otherwise (i.e. timeout on HOST CARD probing) generate a trap/fault message onto debug port and stop all operations.
4. Send BOARD_UP message to HOST CARD indicating that this board is active. HOST CARD will restore the configuration for this board if so configured. Periodic polling (keep-alive) will also be initiated between HOST CARD and this board.
5. The board is now completely operational.
6. Periodically perform the keep-alive handshaking with the HOST CARD during normal operation. In case the handshaking fails, continue the operation to process user payload.

2.1.6 Initialization of PTMC boards

The CPLD/FPGA is responsible to initialize the PMC boards since the HOST CARD is not directly interfacing with the PMCs After a power-on or reset, all the non-HOST CARD boards begin to execute from ROM. Following are the sequence of steps performed by the ROM code of these boards:

1. Perform power-on (diagnostic) tests on the board hardware if required. This is done upon a cold start only.
2. Initialize the hardware and software components of the board – this includes setting up of interrupt tables, initialization of memory modules, initialization of the OS (if there is one), and other system software modules, PCI shared memory interface (for IBC), Flash memory, asynchronous port as well as the software stacks.

3. If failure or any errors detected in above steps, display error message onto the debug port and reset. A terminal or console should be connected to the debug port to observe these messages.
4. Initialize the hardware and software components of the board – this is specific to the board and includes setting up of interrupt tables, initialization of memory modules, initialization of the OS and other system software modules, Ethernet interfaces, flash memory, asynchronous port as well as the protocol software residing on this board.
5. If failure or any errors detected in above steps, display error message onto the debug port and reset.
6. Wait the HOST CARD to probe this board. Then complete the handshaking and initialize the PCI shared memory interface. If everything passes, go into the normal operation mode, otherwise (i.e. timeout on HOST CARD probing) generate a trap/fault message onto debug port and stop all operations.
7. Send BOARD_UP message to HOST CARD indicating that this board is active. HOST CARD will restore the configuration for this board if so configured. Periodic polling (keep-alive) will also be initiated between HOST CARD and this board.
8. The board is now completely operational.

Periodically perform the keep-alive handshaking with the HOST CARD during normal operation. In case the handshaking fails, continue the operation to process user payload..

2.1.7 Software Downloading and Upgrade Procedure

This section describes how the software is downloaded into the WHWYS7020 system and saved into the hard drive and flash memory of the various boards.

The administrator can initiate the download of new software into the hard drive of HOST CARD board at any time during the normal operation. An installation script program will then put the files into the corresponding directories. Next time when the board is reset, this new image is thus used.

The operator will use HOST CARD to download the software for all the boards of the system. Since HOST CARD is equipped with hard drive, it has the space to store the old copies of

HOST CARD and PCI/SBE images. The HOST CARD gets the software from the file server by FTP and passes it to the individual board. The board will verify the software and save it to its flash. The new software will take effect after an automatic reset.

The HOST CARD board uses FTP to retrieve the files from the server. The data received from the server is copied into the hard drive. Then run a script file to complete the upgrade procedure.

Boards other than HOST CARD will obtain their software from HOST CARD over the PCI bus during upgrade. The HOST CARD is the initiator of the PCI/SBE download process when operator issues the CLI download command in HOST CARD. Each time the WHWYS7020 system resets, it will verify the version of software within every module. If version mismatch is detected, it will send alarm traps to notify the operator to download new version of software. The data received from HOST CARD will be copied into RAM until the complete file is downloaded. If successful, the file will be saved to flash memory at a predefined location. If not successful, the HOST CARD will retry the download for a fixed number of times (typically 3 times) and if still unsuccessful, the board will display an error message on its debug port and reset.

2.2 Operational Configuration and Test Set up

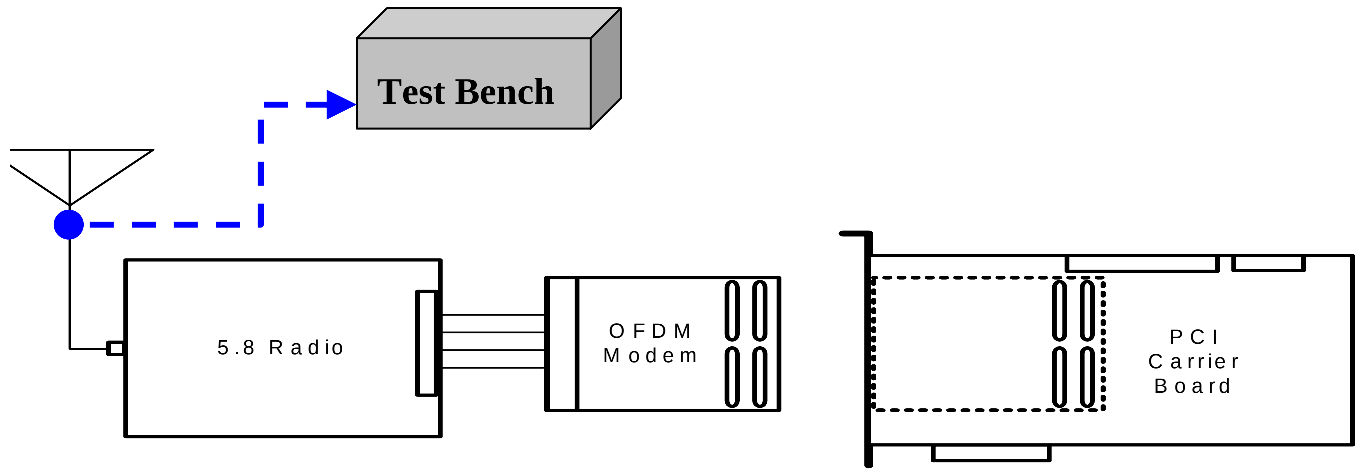


Figure 2 An Example of Test configuration

For RF loop back considerations the RF loop back is enabled between the ports of the DIT output. For monitoring purposes and signal testing the DIT out put is padded and directed into a test component.

2.2.1 Default Configuration Parameters

Table 1: Default Power & Gain setting

DESCRIPTION	VALUE
RF Transmit Power Max	10 dBm
RF Transmit Power Typ	7 dBm
RF Transmit Gain Default	35 dB

Table 2 Default Configuration Parameters

DESCRIPTION	VALUE
RF Transmit Frequency	5750 MHz
RF Receive Frequency	5800 MHz
IF Transmit Frequency	2400 MHz
IF Receive Frequency	2420 MHz
RX Voltage Attenuation	2000 (45 dB Gain)
DAC Voltage Reference	130 (1.2 Volts)
TCXO Reference	10MHz

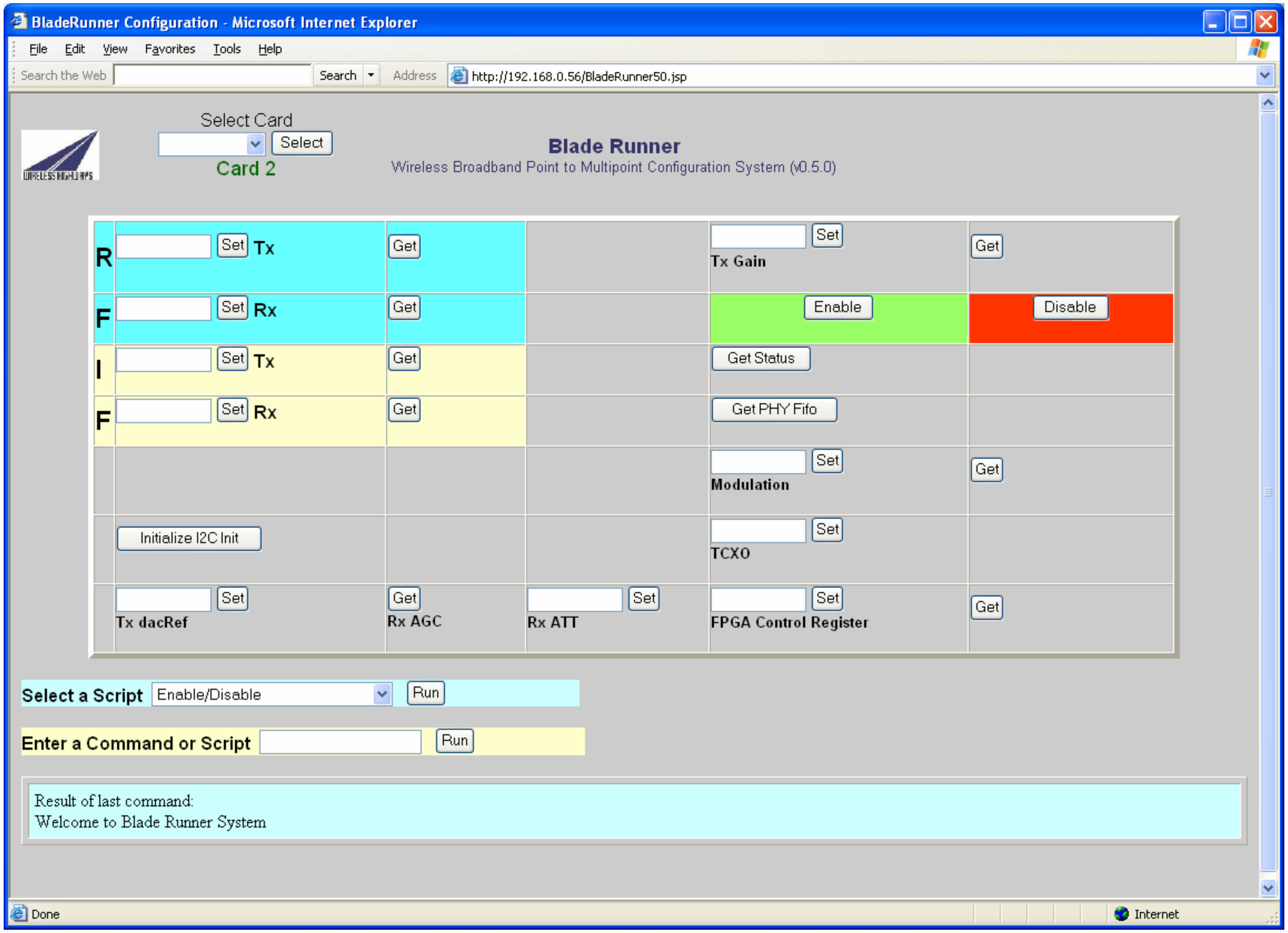
3 User Interface via I2C and GUI

3.1 BladeRunner GUI

Blade Runner is the internal name given to an application that provides the capability to configure Wireless Highway's point-to-point wireless communications system. The system employs a graphical user interface and is designed to operate with any client utilizing a standard internet browser such as Internet Explorer, Netscape and Mozilla.

The application operates in a web centric environment utilizing java server pages in a Linux server environment, and will execute in any environment that has implemented the java virtual machine specification without further modification. Blade Runner will automatically load upon connection of the browser to the host in a non secure test environment. In a secure environment the system will request a valid userid and a password to gain access. If the application fails to load automatically, use the following URL.

"http://host.IP.address/BladeRunner.jsp.



Result of last command:
Welcome to Blade Runner System

Figure 3 BladeRunner Screen Shot

3.2 I2C Command Line Format

The WHWYS7020 supports a Command Line as an operator interface. The list of commands and their description are contained within the Command Set section. The format of the user command is as follows:

cli [Command] [arg1] [arg2]….[arg6]

Each command should respond with a result of the issued command. In example the user wants to initialize the OFDM modules'I2C interface and configure it with the default parameters the following command is entered:

cli i2c_init

If the OFDM module configure properly the response should be:

Welcome to Blade Runner System

Received a message from card # : i2c init done

3.3 Command Set and Description

3.3.1 PHY commands

phy_show_status_reg

Description: Displays the value of the Status Register.
Format: phy_show_status_reg
Parameters: no.
Return: the current value in hexadecimal of the phy status register.

phy_show_rx_link_active_status

Description: Displays the status of the Rx Link. The status setting start default to Idle. When the first REF symbol is detected it become active. Note that it will stay active forever after becoming active.
Format: phy_show_rx_link_active_status
Parameters: no.
Return: the current Rx Link status. (Rx Link currently Active Or Rx Link currently Idle)

phy_show_rx_link_configured

Description: Displays the status of the Rx Link. The status setting start default to Idle. When the first REF symbol is detected it become active. Note that it will stay active forever after becoming active.
Format: phy_show_rx_link_configured
Parameters: no.
Return: current Rx Link status. (Rx Link currently Configured or Rx Link Not currently Configured)

phy_show_rx_data_status

Description: Displays the Status of the Rx FIFO. If the Rx FIFO is 3/4 full or more, this will show Data Ready.

Format: phy_show_rx_data_status
Parameters: no.
Return: current status. (Rx Data Not Ready Or Rx Data Ready)

phy_show_tx_data_status

Description: Displays the Status of the Tx FIFO. If the Tx FIFO is 1/4 full or less, this will show Data Ready.
Format: phy_show_tx_data_status
Parameters: no.
Return: current status. (Tx Data Not Ready Or Tx Data Ready)

phy_show_control_reg

Description: Displays the value of the Control Register.
Format: phy_show_control_reg
Parameters: no.
Return: the current value in hexadecimal of the PHY control register

phy_enable_tx_gain_force

Description: The phy_enable_tx_gain_force command activates transmit gain written by the phy_set_tx_gain command.
Format: phy_enable_tx_gain_force
Parameters: no.
Return: the status.

phy_disable_tx_gain_force

Description: This command de-activates transmit gain written by the phy_set_tx_gain.
Format: phy_disable_tx_gain_force
Parameters: no.
Return: the status.

phy_show_tx_gain_force_status

Description: This command shows the status of the transmit.
Format: phy_show_tx_gain_force_status
Parameters: no.
Return: the status.

phy_show_tx_gain

Description: This command shows the value written by the phy_set_tx_gain command.
Format: phy_show_tx_gain
Parameters: no.
Return: the value ranging from 0 to 255.

phy_set_tx_gain

Description: This command is used to write the tx gain to the PHY.
Format: phy_set_tx_gain [value]
Parameters: value Range are from 0 to 2^$_8$-1 in absolute unit.
Return: none

phy_show_agc

Description: This command will show the value of the power level of the Reference symbols of the symbols.
Format: phy_show_agc
Parameters: no.
Return: the value ranging from 0 to 255.

phy_show_tx_enable

Description: This command shows the status of the tx enable bit. If this bit is OFF, it will mute all data on the transmit section of the PHY.
Format: phy_show_tx_enable
Parameters: no.
Return: status. (Tx Enable is currently ON or Tx Enable is currently OFF)

phy_enable_tx

Description: This command activates transmit by un-muting the output of the PHY. No data will be transmitter.
Format: phy_enable_tx
Parameters: no.
Return: status. (enabled)

phy_disable_tx

Description: This command de-activates transmit by muting the output of the PHY. No data will be transmitted.
Format: phy_disable_tx
Parameters: no.
Return: status. (disabled)

phy_show_tx_config_continuous

Description: This command shows the status of the tx config continuous bit.
Format: phy_show_tx_config_continuous
Parameters: no.
Return: status. (ON or OFF)

phy_show_rx_config_continuous

Description: This command shows the status of the rx continuous bit.
Format: phy_show_rx_config_continuous
Parameters: no.
Return: status. (ON or OFF)

phy_enable_tx_config_continuous

Description: This command activates the tx using the current configuration word in continuous mode. If no data it will send idle data.
Format: phy_enable_tx_config_continuous
Parameters: no.

phy_disable_tx_config_continuous

Description: This command de-activates the tx continuous mode. This mode is the default.
Format: phy_disable_tx_config_continuous
Parameters: no.
Return: status. (disabled)

phy_enable_rx_config_continuous

Description: This command activates the rx using the current configuration word in continuous mode.
Format: phy_enable_rx_config_continuous
Parameters: no.
Return: status. (enabled)

phy_disable_rx_config_continuous

Description: This command de-activates the rx continuous mode. This mode is the default. ***Format:*** phy_disable_rx_config_continuous
Parameters: no.
Return: status. (disabled)

phy_show_operate_mode
Description: This command shows if the PHY is in operate/standby mode.
Format: phy_show_operate_mode
Parameters: no.
Return: status. (operate mode or standby mode)

phy_enable_operate_mode
Description: This command puts the PHY in operate mode.
Format: phy_enable_operate_mode
Parameters: no.
Return: status. (enabled)

phy_disable_operate_mode
Description: This command puts the PHY in standby mode. In this mode, the PHY is in FFT feedback and the Data FIFO is reset.
Format: phy_disable_operate_mode
Parameters: no.
Return: status. (disabled-> standby mode)

phy_show_tx_station_mode
Description: This command shows if the Tx section of the PHY is in base station or in subscriber mode.
Format: phy_show_tx_station_mode
Parameters: no.
Return: status. (base station mode or subscriber mode)

phy_show_rx_station_mode
Description: This command shows if the Rx of the PHY is in base station or in subscriber mode.
Format: phy_show_rx_station_mode
Parameters: no.
Return: status. (base station mode or subscriber mode)

phy_enable_tx_bs_mode
Description: This command activates the Tx of the PHY in base station mode and de-activates the subscriber mode.
Format: phy_enable_tx_bs_mode
Parameters: no.
Return: status. (base station enabled)

phy_disable_tx_bs_mode
Description: This command de-activates the Tx of the PHY from the base station mode and activates the subscriber mode.
Format: phy_disable_tx_bs_mode
Parameters: no.
Return: status. (base station disabled)

phy_enable_rx_bs_mode
Description: This command activates the Rx of the PHY in base station mode and de-activates the subscriber mode.
Format: phy_enable_rx_bs_mode
Parameters: no.

Return: status. (base station enabled)

phy_disable_rx_bs_mode

Description: This command de-activates the Rx of the PHY from the base station mode and activates the subscriber mode.

Format: phy_disable_rx_bs_mode

Parameters: no.

Return: status. (base station disabled)

phy_show_revision_code

Description: This command shows the revision of the PHY.

Format: phy_show_revision_code

Parameters: no.

Return: revision code.

For example,

PHY: Revision Code = 0x5701003a

Where:

-57 is the Company.

-01 is the Version. Version 00 = Software SAR FPGA or Version 01 = Software SAR ASIC

-00 is the revision.

-3a is the build number.

phy_show_interrupt_reg

Description: This command Displays the value of the interrupt register.

Format: phy_show_interrupt_reg

Parameters: no.

Return: the current value in hexadecimal of the PHY interrupt register.

For example,

PHY: Interrupt Register = 0x01

Where:

- Bit 0: = Receive data buffer ready.
- Bit 1: = Transmit data buffer ready.
- Bit 2: = RX end of frame interrupt.
- Bit 3: = Receive data buffer overflow.
- Bit 4: = TX end of frame interrupt.
- Bit 5: = CFG symbol received.
- Bits (7:6): spare

***Note that A .1. written to the associated bit enables that interrupt.

phy_enable_rx_irq

Description: This command enables the receive data buffer ready interrupt. This interrupt occurs when the Rx buffer is at ¾ capacity.

Format: phy_enable_rx_irq

Parameters: no.

Return: status (enabled)

phy_disable_rx_irq

Description: This command enables disables the receive data buffer ready interrupt.

Format: phy_disable_rx_irq

Parameters: no.

Return: status (disabled)

phy_enable_tx_irq

Description: This command enables the Transmit data buffer ready interrupt. This interrupt occurs when the buffer is at ¼ capacity.

Format: phy_enable_tx_irq
Parameters: no.
Return: status (enabled)

phy_disable_tx_irq

Description: This command disables the Transmit data buffer ready interrupt.
Format: phy_disable_tx_irq
Parameters: no.
Return: status (disabled)

phy_enable_rx_end_of_frame_irq

Description: This command enables the RX end of frame interrupt.
Format: phy_enable_rx_end_of_frame_irq
Parameters: no.
Return: status (enabled)

phy_disable_rx_end_of_frame_irq

Description: This command disables the RX end of frame interrupt.
Format: phy_disable_rx_end_of_frame_irq
Parameters: no.
Return: status (disabled)

phy_enable_tx_end_of_frame_irq

Description: This command enables the TX end of frame interrupt.
This interrupt indicates the end of a transmit frame.
Format: phy_enable_tx_end_of_frame_irq
Parameters: no.
Return: status (enabled)

phy_disable_tx_end_of_frame_irq

Description: This command disables the TX end of frame interrupt.
Format: phy_disable_tx_end_of_frame_irq
Parameters: no.
Return: status (disabled)

phy_enable_rx_buffer_overflow_irq

Description: This command enables the Receive data buffer overflow interrupt. This interrupt indicates that a Rx Buffer overflow occurred and that data in the receive buffer is overrun.
Format: phy_enable_rx_buffer_overflow_irq
Parameters: no.
Return: status (enabled)

phy_disable_rx_buffer_overflow_irq

Description: This command disables the Receive data buffer overflow interrupt.
Format: phy_disable_rx_buffer_overflow_irq
Parameters: no.
Return: status (disabled)

phy_enable_config_symbol_irq

Description: This command enables the CFG symbol received interrupt. This interrupt indicate that a CFG symbol is received.
Format: phy_enable_config_symbol_irq
Parameters: no.
Return: status (enabled)

phy_disable_config_symbol_irq
Description: This command disables the CFG symbol received interrupt.
Format: phy_disable_config_symbol_irq
Parameters: no.
Return: status (disabled)

phy_show_dma_reg
Description: This command shows the value of the DMA control Register.
Format: phy_show_dma_reg
Parameters: no.
Return: console returns the current value in hexadecimal of the PHY DMA control register.
For example,
PHY: Status Register = 0x00
Where:
- Bit 30: = Receive DMA.
- Bit 31: = Transmit DMA.

phy_show_rx_dma_status
Description: This command shows the value of the RX DMA bit status.
Format: phy_show_rx_dma_status
Parameters: no.
Return: status (enabled or disabled)

phy_show_tx_dma_status
Description: This command displays the value of the TX DMA bit status.
Format: phy_show_rx_dma_status
Parameters: no.
Return: status (enabled or disabled)

phy_enable_rx_dma
Description: This command enables DMA requests for received data. Request will be temporarily halted on empty receive buffer.
Format: phy_enable_rx_dma
Parameters: no.
Return: status (enabled)

phy_enable_tx_dma
Description: This command enables DMA requests for transmitted data. Request will be temporarily halted on full transmit buffer.
Format: phy_enable_tx_dma
Parameters: no.
Return: status (enabled)

phy_show_rx_data_count
Description: This command shows the number of bytes in the RX buffer of the PHY.
Format: phy_show_rx_dma_count
Parameters: no.
Return: current value

phy_show_tx_config_buffer1_status
Description: This command shows the status of the Tx config buffer 1. The status FRESH means set by the PPC for the PHY to use and DIRTY means cleared after use by the PHY.
Format: phy_show_tx_config_buffer1_status

Parameters: no.
Return: current value (FRESH or DIRTY)

phy_show_tx_config_buffer2_status
Description: This command shows the status of the Tx config buffer 2. The status FRESH means set by the PPC for the PHY to use and DIRTY means cleared after use by the PHY.
Format: phy_show_tx_config_buffer2_status
Parameters: no.
Return: current value (FRESH or DIRTY)

phy_show_tx_config_buffer3_status
Description: This command shows the status of the Tx config buffer 3. The status FRESH means set by the PPC for the PHY to use and DIRTY means cleared after use by the PHY.
Format: phy_show_tx_config_buffer3_status
Parameters: no.
Return: current value (FRESH or DIRTY)

phy_show_rx_config_buffer1_status
Description: This command shows the status of the Rx config buffer 1. The status FRESH means set by the PPC for the PHY to use and DIRTY means .cleared after use by the PHY.
Format: phy_show_rx_config_buffer1_status
Parameters: no.
Return: current value (FRESH or DIRTY)

phy_show_rx_config_buffer2_status
Description: This command shows the status of the Rx config buffer 2. The status FRESH means set by the PPC for the PHY to use and DIRTY means .cleared after use by the PHY.
Format: phy_show_rx_config_buffer2_status
Parameters: no.
Return: current value (FRESH or DIRTY)

phy_show_base_station_sid
Description: This command shows the SID value.
Format: phy_show_base_station_sid
Parameters: no.
Return: current value

phy_set_base_station_sid
Description: This command sets the base station SID value.
Format: phy_set_base_station_sid [value]
Parameters: [value] This value is used to identify the base station. Value Default is 0x01.
Return: current value

phy_show_local_station_sid
Description: This command shows the SID value.
Format: phy_show_local_station_sid
Parameters: no
Return: current value

phy_set_local_station_sid
Description: This command sets the local station SID value.
Format: phy_set_local_station_sid [value]

Parameters: [value] This value is used to identify the subscriber station. Value range is from 3 to 2047. (1 is reserved to the base station and 2 is for the contention slot)
Return: current value

phy_write_tx_word

Description: This command writes a hex or decimal value to the PHY Tx Data fifo.
Format: phy_write_tx_word [value]
Parameters: [value] is a 32 bit words
Return: none

phy_write_tx_word_loop

Description: This command writes 32 bit words to the Tx fifo with values starting at zero and incrementing, as many times as described by the number of iterations specified by the first argument. Note that if the Tx fifo is full, the function will continue to write because there is no way to tell if the fifo is full or empty.

Format: phy_write_tx_word_loop [nb]

Parameters: [nb] number of iterations.
Return: none

phy_read_rx_word

Description: This command reads the next 32 bits from the PHY Rx FIFO.

Format: phy_read_rx_word
Parameters: none
Return: the current value

phy_read_rx_word_loop

Description: This command reads 32 bits of data from the Rx fifo for the number of times given by the argument. If the amount of data is less than the amount of data requested to read, an error is printed. If the data on each read iteration does not match the iteration count, an error is printed and the remainder of the data is not read.
Format: phy_read_rx_word_loop
Parameters: none
Return: the current value

phy_read_prep_header

Description: This command reads the bytes contained in the prepended header.

Format: phy_read_prep_header
Parameters: none
Return: the current value

phy_flush_rx

Description: This command flushes the Rx FIFO by reading out a 32 bit value if the count is greater than or equal to 4 bytes.
Format: phy_flush_rx
Parameters: No parameters are required. If an argument is passed, the progress is printed.
Return: status

phy_flush_print_rx

Description: This command flushes the Rx fifo by reading out 32 bit value if the count is greater than or equal to 4 bytes. It also print out the value read.

Format: phy_flush_print_rx
Parameters: No parameters are required.
Return: read value

phy_show_delay_correction

Description: This command displays the range delay correction in number of samples clock.

Format: phy_show_delay_correction
Parameters: No parameters are required.
Return: delay correction value

phy_set_delay_correction

Description: This command sets the range delay correction in number of samples clock. It is required to advance or compensate for SS range delay. Note that this command must be execute on the sub station. For reference, the typical values are 1250 for a digital link and 1390 for an analog link. For the RF link it increase as the distance increase between the sub and the base station.

Format: phy_set_delay_correction [delay]

Parameters: [delay] Delay value in decimal
Return: none

phy_show_lost_packet_count

Description: This command shows the number of PDUs discarded by the FEC.

Format: phy_show_lost_packet_count
Parameters: No parameters are required.
Return: Lost Packet Count

phy_show_frame_number

Description: This command displays the current frame number.

Format: phy_show_frame_number
Parameters: No parameters are required.
Return: Frame Number

phy_show_prefix_thresh

Description: This command shows the prefix threshold value.

Format: phy_show_prefix_thresh
Parameters: No parameters are required.
Return: Prefix Threshold

phy_set_prefix_thresh

Description: This command sets the prefix threshold to a given value. This value is usually calibrated and should not be changed. If this value is decreases the number of false syncs will increase and if the value is increased the number of missed syncs will increase.
Format: phy_set_prefix_thresh [threshold]

Parameters: [threshold] 16 bits value
Return: Threshold

phy_show_force_prefix

Description: This command shows the status of the force prefix bit. If it is enabled, then the value set by the phy_set_prefix_thresh will be used. If disabled, an internal default value will be used.
Format: phy_show_force_prefix
Parameters: No parameters are required.
Return: status (enabled or disabled)

phy_enable_force_prefix

Description: This command enables the current prefix threshold value programmed in the PHY.
Format: phy_enable_force_prefix
Parameters: No parameters are required.
Return: status (enabled)

phy_disable_force_prefix

Description: This command disables the current prefix threshold value programmed in the PHY.
Format: phy_disable_force_prefix
Parameters: No parameters are required.
Return: status (disabled)

phy_show_feedback

Description: This command shows the feedback controls.
Format: phy_show_feedback
Parameters: No parameters are required.
Return: status

phy_disable_feedback

Description: This command disables all the enabled feedback if there are any.
Format: phy_disable_feedback
Parameters: No parameters are required.
Return: status

phy_enable_data_feedback

Description: This command enables the data and fft feedback bit and disables all others.
Format: phy_enable_data_feedback
Parameters: No parameters are required.
Return: status

phy_enable_fft_feedback

Description: This command enables the fft feedback bit and disables all others.
Format: phy_enable_fft_feedback
Parameters: No parameters are required.
Return: status

phy_enable_packet_feedback

Description: This command enables the packet and fft feedback bits and disables all others.
Format: phy_enable_packet_feedback
Parameters: No parameters are required.
Return: status

phy_enable_codeword_feedback

Description: This command enables the codeword and fft feedback bits and disables all others.
Format: phy_enable_codeword_feedback
Parameters: No parameters are required.
Return: status

phy_enable_sample_feedback

Description: This command enables the sample and disables all others.
Format: phy_enable_sample_feedback
Parameters: No parameters are required.
Return: status

3.3.2 POLL driver command

poll_phycfg_stats

Description: This command is used to display poll_driver statistics.
Format: poll_phycfg_stats [-r]
Parameters: The only optional parameter allows the user to reset the counters.
Return: statistics
The following statistics is displayed:

```
Tx Frames: 3
Rx Frames: 0
Rx Cfgs: 3
Rx Lost Frame Sync: 0
Rx UL Maps: 0
Tx Buf1 Sets: 1  Tx Buf1 Unavailable: 0
Tx Buf2 Sets: 1  Tx Buf2 Unavailable: 0
Tx Buf3 Sets: 1  Tx Buf3 Unavailable: 0
Rx Buf1 Sets: 2  Rx Buf1 Unavailable: 0
Rx Buf2 Sets: 1  Rx Buf2 Unavailable: 0
```

poll_add_user

Description: This command is used to add a subscriber to the schedulers' database. The scheduler will then allocate bandwidth to the given subscriber. Once a user is added to the database, the only operations permitted on the subscriber are poll_modify_user (used to modify an existing users' parameters) and poll_remove_user (used to remove an existing user from the database).
Format: poll_add_user SID [DL-Modulation] [DL-Slots] [UL-Modulation] [DL-Slots]
Parameters:
[SID] This is the subscriber ID. Valid values are from 4 to 49. Some SID values are reserved. Values 0 to 3 are explicitly reserverd.
[DL-Modulation] The is the modulation scheme to be used for this subscriber's traffic which is transmitted from the Base Station in the DL sub-frame. Value values are {QPSK, QAM16, QAM64}.

[DL-Slots] This is the number of OFDM Data symbols allocated to this subscriber for transmission in the DL sub-frame. Valid values are 0 to 30.
[UL-Modulation] The is the modulation scheme to be used for this subscriber's traffic which is transmitted from the Subscriber Station to the Base Station in the UL sub-frame. Valid values are {QPSK, QAM16, QAM64}.
[UL-Slots] This is the number of OFDM Data symbols allocated to this subscriber for its burst in the UL sub-frame. Valid values are 0 to 24.
Return: none
For example,
poll_add_user 6 QAM16 12 QPSK 8

poll_modify_user

Description: This command is used to modify and existing users' scheduling and allocation parameters. The format is the same as that for the poll_add_user command.
Format: poll_modify_user SID [DL-Modulation] [DL-Slots] [UL-Modulation] [DL-Slots]
Parameters: see poll_add_user.
Return: none.

poll_remove_user

Description: This command is used to remove an existing user from the scheduler's database.
Format: poll_remove_user SID
Parameters:
[SID] The subscriber's ID. Valid values are 4 to 49.
Return: none.

poll_display_list

Description: This command is used to print out the list of subscribers currently in the scheduler's database.
Format: poll_display_list
Parameters: none
Return: list of subscribers.

poll_trace_config

Description: This command is a debug command that is used to display the last 30 writes to the Tx and Rx Configuration control buffers.
Format: poll_trace_config [tx | rx]
Parameters:
[tx] This command is used to display the last 30 writes to the tx config word buffer.
[rx] This command is used to display the last 30 writes to the rx config word buffer.

Return: trace message

poll_rand_alloc

Description: This command is to randomly change the allocation parameters of an existing user in the database. The only required parameter is the subscriber id (SID).
Format: poll_rand_alloc SID
Parameters:
[SID] The subscriber station identifier (SID). Valid values are 2 to 49.
Return:

3.3.3 Subscriber command

ss_driver_stats Command

Description: This command is used to display information about the ssProc (Subscriber Station MAP processor) statistics.

Format: ss_driver_stats [-r]

Parameters:

The only parameter allows the user to reset the counters

Return:

The following statistics is displayed:

Late writes: 0
Late MAPS: 0
Missing Cfgs: 0

agc_driver Command

Description: This command is used at initialization or in a script file to configure the automatic Rx AGC loop at the subscriber station. This is a software driver which monitors received power levels and makes periodic adjustments to the receiver gain control block. This command is only applicable for the Subscriber Station.

Format: agc_driver [init | start | status | debug | auto | manual | sp [value] | mv | output [value] | max [limit] | min [limit]]

Parameters:

Separate keywords are used for each parameter. Some keywords have an optional second parameter which is used to set a value for a given variable (parameter name). The parameter keywords are as follows:

[init] This parameter allocates memory and initializes the driver's control block. It should be called only one after power up from either a script file or from the command line.

[start] This parameter starts the agc_driver with the values contained in the driver's control block. Normally the keywords are used to initialize these values.

[status] This parameter causes the driver to display its current state.

[debug] This parameter invokes the printing of debug messages on the display. This parameter is used for software testing. The parameter toggles the display of debug messages on and off every time the command is invoked.

[auto] This places the agc_driver in closed loop control. This means it will attempt to maintain the received power level at the threshold defined by the contents of sp.

[manual] This places the agc_driver in open-loop control. This means it will not attempt any adjustments to the gain control block. It will remain at the last setting. The output level can be displayed using the agc_driver output command.

[sp] [value] Displays the current desired value for received power. The optional value parameter allows you to change the desired setpoint. mv Displays the current received power level from the PHY as measured over the reference symbol. Units are in dB * 10. For example, if the value of mv is 117, then this is 11.7 dB.

[output] [value] Displays the current output value being applied to the Rx DAC. The output value can be changed using the optional value parameter. This only has an effect when the driver is in manual mode.

[max] [value] Displays the current maximum value that can be applied to the Rx gain control DAC. The output value will be pinned at this level. The max value can be changed using the optional value parameter.

[min] [value] Displays the current minimum value that can be applied to the Rx gain control DAC. The output value will be pinned at this level. The min value can be changed using the optional value parameter.

[window] [value] Displays the current window value. This creates a window around the sp that is +- this value. As long as the mv is within this range, no adjustments to the output Rx DAC will be attempted. The default window size is 20. The window value can be changed using the optional parameter, value.
Return:
For example, to initialize and start the driver, type the following commands:

```
agc_driver init
agc_driver start
```

Here is an example of an initialization script:

```
agc_driver sp 74
agc_driver output 128
agc_driver min 50
agc_driver max 200
agc_driver window 10
agc_driver auto
```

This initializes the operating parameters and sets the control mode to automatic.

3.3.4 FPGA command

fpga_enable_drive_netref1

Description: This command is used to enable netref1 clock.
Format: fpga_enable_drive_netref1
Parameters: none
Return: status

fpga_enable_drive_netref2

Description: This command is used to enable netref2 clock.
Format: fpga_enable_drive_netref2
Parameters: none
Return: status

fpga_disable_drive_netref1

Description: This command is used to disable netref1 clock.
Format: fpga_disable_drive_netref1
Parameters: none
Return: status

fpga_disable_drive_netref2

Description: This command is used to disable netref2 clock.
Format: fpga_disable_drive_netref2
Parameters: none
Return: status

f_e_tx_bridge

Description: This command is used to enable tx CT bridge.
Format: f_e_tx_bridge
Parameters: none
Return: status

f_e_rx_bridge

Description: This command is used to enable rx CT bridge.
Format: f_e_rx_bridge
Parameters: none
Return: status

f_d_tx_bridge

Description: This command is used to disable tx CT bridge.
Format: f_d_tx_bridge
Parameters: none
Return: status

f_d_rx_bridge

Description: This command is used to disable rx CT bridge.
Format: f_d_rx_bridge
Parameters: none
Return: status

fpga_get_tx_ctframe_per_pdu

Description: This command is used to get number of ctframes per pdu.
Format: fpga_get_tx_ctfrme_per_pdu
Parameters: none
Return: number of ctframes per pdu

fpga_set_tx_ctframe_per_pdu

Description: This command is used to set number of ctframes per pdu.
Format: fpga_set_tx_ctfrme_per_pdu [ctframe]
Parameters:
[ctframe] is number of ctframes per pdu.
Return: number of ctframes per pdu

fpga_get_tx_fifo_level

Description: This command is to used to get tx fifo level.
Format: fpga_get_tx_fifo_level
Parameters: none
Return: number of double words left in TX FIFO

fpga_get_rx_fifo_level

Description: This command is to used to get rx fifo level.
Format: fpga_get_rx_fifo_level
Parameters: none
Return: number of double words left in RX FIFO

show_tx_lut

Description: This command is used to show TX look up table.
Format: show_tx_lut
Parameters: none
Return: Look up table

fpga_set_test_mode

Description: This command is used to show RX look up table.
Format: show_rx_lut

Parameters: none
Return: Look up table

fpga_disable_tx_all_slot

Description: This command is used to disable all the tx timeslots.
Format: fpga_disable_tx_all_slot
Parameters: none
Return: none

fpga_disable_rx_all_slot

Description: This command is to used disable all the rx timeslots.
Format: fpga_disable_rx_all_slot
Parameters: none
Return: none

fpga_enable_tx_tm_slot

Description: This command is used to enable one timeslot in tx look up table.
Format: fpga_enable_tx_tm_slot [timeslot] [line]
Parameters:
[timeslot] timeslot number (0-127)
[line] CT line number (0-7)
Return: none

fpga_disable_tx_tm_slot

Description: This command is used to disable one timeslot tx look up table.
Format: fpga_disable_tx_tm_slot [timeslot] [line]
Parameters:
[timeslot] timeslot number (0-127)
[line] CT line number (0-7)
Return: none

fpga_enable_rx_tm_slot

Description: This command is used to enable one timeslot rx look up table.
Format: fpga_enable_rx_tm_slot [timeslot] [line]
Parameters:
[timeslot] timeslot number (0-127)
[line] CT line number (8-15)
Return: none

fpga_disable_rx_tm_slot

Description: This command is used to disable one timeslot rx look up table.
Format: fpga_disable_rx_tm_slot [timeslot] [line]
Parameters:
[timeslot] timeslot number (0-127)
[line] CT line number (8-15)
Return: none

fpga_get_reg

Description: This command is used to get hexadecimal value in a register.
Format: fpga_get_reg [reg]
Parameters:
[reg] address of the register

Return: value in register

fpga_set_reg

Description: This command is used to set hexadecimal value in a register.

Format: fpga_set_reg [reg] [value]

Parameters:

[reg] address of the register

[value] value in hexadecimal that want to set to the register

Return: value in register

fpga_get_ctrl_reg

Description: This command is used to get hexadecimal value in control register.

Format: fpga_get_ctrl_reg

Parameters: none

Return: value in control register

fpga_set_ctrl_reg

Description: This command is used to set hexadecimal value in control register.

Format: fpga_set_ctrl_reg [value]

Parameters:

[value] value in hexadecimal that want to set to the control register

Return: value in control register

3.3.5 Radio command

i2c_init

Description: This command is used to initialize the i2c Bus.

Format: i2c_init

Parameters: NONE

Return: Success

s_tx_rffreq

Description: This command is used to set the TX RF frequency for the radio through i2c Bus.

Format: s_tx_rffreq [arg1]

Parameters:

[arg1] TX RF frequency in the range of [3200, 3400] Mhz.

Return: Value of the set TX RF frequency .

s_rx_rffreq

Description: This command is used to set the RX RF frequency for the radio through i2c Bus.

Format: s_rx_rffreq [arg1]

Parameters:

[arg1] RX RF frequency in the range [3200, 3400] Mhz.

Return: Value of the set RX RF frequency .

s_rx_att

Description: This command is used to set the RX attenuation for the radio through i2c Bus.

Format: s_rx_att [arg1]

Parameters:

[arg1] RX attenuation in the range of [1400, 2400]

Return: Value of the set RX attenuation.

g_tx_power
Description: This command is used to get the TX power from the radio through i2c Bus.
Format: g_tx_power
Parameters: None.
Return: Value of the current TX power for the radio***.***

s_tcxo
Description: This command is used to set the 10Mhz reference VCO for the radio through i2c Bus.
Format: s_tcxo [arg1]
Parameters:
[arg1] 10Mhz reference in the range of [0, 255].
Return: Value of the set 10Mhz reference VCO.

g_tx_rffreq
Description: This command is used to get the TX RF frequency from the radio through i2c Bus.
Format: g_tx_rffreq
Parameters: None
Return: Value of the current TX RF frequency.

g_rx_rffreq
Description: This command is used to get the RX RF frequency from the radio through i2c Bus.
Format: g_rx_rffreq
Parameters: None.
Return: Value of the current RX RF frequency.

s_tx_iffreq
Description: This command is used to set the TX IF frequency for the radio through i2c Bus.
Format: s_tx_iffreq [arg1]
Parameters:
[arg1] TX IF frequency in the range of [2400, 2500] Mhz.
Return: Value of the set TX IF frequency.

s_rx_iffreq
Description: This command is used to set the RX IF frequency for the radio through i2c Bus.
Format: s_rx_iffreq [arg1]
Parameters:
[arg1] RX IF frequency in the range of [2400, 2500] Mhz.
Return: Value of the set RX IF frequency.

g_tx_iffreq
Description: This command is used to get the TX IF frequency from the radio through i2c Bus.
Format: g_tx_iffreq
Parameters: None
Return: Value of the current TX IF frequency.

g_rx_iffreq
Description: This command is used to get the RX IF frequency from the radio through i2c Bus.
Format: g_rx_iffreq

Parameters: None
Return: Value of the current RX IF frequency

s_tx_dacRef
Description: This command is used to set the TX DAC reference for the radio through i2c Bus.
Format: s_tx_dacRef [arg1]
Parameters:
[arg1] TX DAC ref in the range of [0, 130].
Return: Value for the set TX DAC ref.

s_tx_gain
Description: This command is used to set the TX gain for the modem card.
Format: s_tx_gain [arg1]
Parameters:
[arg1] TX gain in the range of [0, 80]. The power-up default value is 40.
Return: Value for the set TX gain

g_tx_gain
Description: This command is used to get the TX gain for the modem card.
Format: g_tx_gain
Parameters: None
Return: Value for the current TX gain

get_hw_version
Description: This command is used to get the FPGA version for the modem card.
Format: get_hw_version
Parameters: None
Return: The FPGA version number

get_sw_version
Description: This command is used to get the software version for the modem card.
Format: get_sw_version
Parameters: None.
Return: Software version number.

e_auto_reset
Description: This command is used to enable the auto reset function.
Format: e_auto_reset
Parameters: None
Return: Success.

d_auto_reset
Description: This command is used to disable the auto reset function
Format: d_auto_reset
Parameters: None
Return: Success.

g_auto_reset
Description: This command is used to display the status of the auto reset function.
Format: g_auto_reset

Parameters: None
Return: The status of the auto reset function: enabled | disabled

e_txpdu_check
Description: This command is used to enable TX pdu check. If enabled, the zero TX PDU case will trigger the auto reset function.
Format: e_txpdu_check
Parameters: None
Return: Success

d_txpdu_check
Description: This command is used to disable the TX PDU check.
Format: d_txpdu_check
Parameters: None
Return: Success.

g_txpdu_check
Description: This command is used to display the status of TX PDU check
Format: g_txpdu_check
Parameters: None
Return: Status of TX PDU check: enabled |disabled.

board_reset
Description: This command is used to manually reset the modem card
Format: board_reset
Parameters: None
Return: None

g_plr_max
Description: This command is used to get the PLR threshold for triggering auto reset function
Format: g_plr_max
Parameters: None
Return: PLR threshold.

s_plr_max
Description: This command is used to set the PLR threshold for triggering auto reset function
Format: s_plr_max [arg1]
Parameters:
[arg1] PLR threshold value in the range of [0, 100]
Return: Value for the set PLR threshold

g_plr_rep
Description: This command is used to get the value for PLR_REP. If the packet loss rate is bigger than PLR_MAX for PLR REP consecutive seconds, the auto reset function will be triggered when enabled.
Format: g_plr_rep
Parameters: None
Return: Value for PLR_REP.

s_plr_rep

Description: This command is used to set the value for PLR_REP
Format: s_plr_rep [arg1]
Parameters:
[arg1] value for PLR_REP.
Return: The set value for PLR_REP.

g_reset_count
Description: This command is used to get the number of board resets.
Format: g_reset_count
Parameters: None
Return: Number of board resets

clr_reset_count
Description: This command is used to clear the reset counter
Format: clr_reset_count
Parameters: None
Return: Success.

g_errchk_cycles
Description: This command is used to get the period for error (PLR) checking.
Format: g_errchk_cycles
Parameters: None
Return: Period for error (PLR) checking.

s_errchk_cycles
Description: This command is used to set the period for error (PLR) checking
Format: s_errchk_cycles [arg1]
Parameters:
[arg1] Period for period for error (PLR) checking
Return: the set value for error (PLR) checking period

3.4 T1/E1 mode selection

ict config t1e1 <slot(integer)> {/(string)} <port(integer)> [**mode**(string) {t1|e1}(string)]
default value: t1

1. Framing Method
ict config t1e1 <slot(integer)> {/(string)} <port(integer)> [**framing**(string) {sf |esf | crc4 | no-crc4} (string)]
default value : esf (t1) crc4(e1)

2. Line coding
ict config t1e1 <slot(integer)> {/(string)} <port(integer)> [**linecode**(string) {ami | b8zs | hdb3}(string)]
default value: ami

3. Clock Source selection

ict config t1e1 <slot(integer)> {/(string)} <port(integer)> [**clock**(string) {internal | line }(string)]
default value : internal

4. Cable length
ict config t1e1 <slot(integer)> {/(string)} <port(integer)> [**cablelength**(string) {e1_75_ohm | e1_120_ohm | t1_0_db | t1_7.5_db | t1_15_db | t1_22.5_db | t1_110_ft | t1_220_ft | t1_330_ft | t1_440_ft | t1_550_ft | t1_660_ft }(string)]
default value: t1_7.5_db(t1), e1_120_ohm(e1)

5. Loopback
ict config t1e1 <slot(integer)> {/(string)} <port(integer)> [**loopback**(string) {none | line | payload | digital}(string)]
default value: none

6. All ones signal (AIS)
ict config t1e1 <slot(integer)> {/(string)} <port(integer)> [**allones**(string) {on | off}(string)]
default value: off

7. show Current configuration and/or status
ict config t1e1 <slot(integer)> {/(string)} <port(integer)> [**show**]

3.4.1 T8110 Configuration commands

1. Clear all the connections
ict config t8110 <slot> [**reset**(string)]

2. Manufacture default
ict config t8110 <slot> [**default**(string)]

3. Add connection
ict config t8110 <slot> [**add**(string)] {<Hbus|Lbus|FPGA>, <stream>,<timeslot>} {<Hbus|Lbus|FPGA>, <stream>,<timeslot>}

4. Delete connection
ict config t8110 <slot> [**del**(string)] {<Hbus|Lbus|FPGA>, <stream>,<timeslot>} {<Hbus|Lbus|FPGA>, <stream>,<timeslot>}

5. Set H.100 bus master
ict config t8110 <slot> [**Hclk**(string)] {master | slave}

6. show connection list
ict config t8110 <slot> [**show**(string)]

4 Radio-Digital Interface Transceiver (DIT)

The radio unit is capable of transmitting and receiving on many different frequency bands including: 5.8GHz. This document describes the 5.7/5.8 GHz Digital Interface Transceiver (DIT). The transmit input signal is 12 bit LVDS base-band signal. The maximum transmit RF output signal is 250 mW. The receive RF input level is from –20 to –90 dBm. The receive output is a 12 bit LVDS base-band signal. The telemetry control and monitor signals are through an I2C bus. A 58 pin digital interface connector connects to the modem at base-band. The RF diplexer output has an SMA connector to interface to an external antenna. DC voltage interface connector is the regular PC power supply connector. The DC voltage is 12 VDC at 0.83 amp. Max. and 5 VDC, 0.50 amp max.

4.1 Theory of operation

The DIT generates and terminates an OFDM signal on the 5.8GHz Band. It accomplishes this in a two stage process; for both the transmitter and receiver sections. The transmitter and receiver sections can be broken down as follows:

4.1.1 The Transmitter section

- The QPSK, QAM16 or QAM64 signal is generated in the OFDM modem card in a 12 bit Base Band stream.
- The first stage of the DIT terminates the 12-bit OFDM stream and converts it to an analog serial signal with a sampling rate at or around 12.8 MHz.
- The signal is then filtered to remove low frequency and base band components.
- The analog serial signal is then converted to IF by a modulator at or around 2.4GHz from a source clock of 10MHz.
- In the second stage the IF signal is up converted to RF and filtered by a composite signal mixer at or around 5.75GHz
- The RF signal is then amplified to the desired power level at or around 15 dBm and transmitted through the Diplexer to the antenna.

4.1.2 The Receiver section

- The RF signal is receiver from the antenna and passed through the Diplexer, then amplified to the desired level.
- The RF signal is then down converted to IF and filtered by a composite signal mixer at or around 2.42GHz with a source clock of 10MHz.
- The signal is then attenuated to the desired level.
- The IF signal is then down converted again by a mixer whose reference clock is 10MHz and filtered.
- The analog IF serial signal is then converted to 12-bit digital Base-Band by the Analog to Digital converter with sampling clock at or around 12.8MHz.
- The OFDM modem card then terminates the QPSK, QAM16 or QAM64 signal.

4.2 ELECTRICAL SPECIFICATIONS

4.2.1 RF Frequency Plan:

The Transmit channel (6MHz Bandwidth) can be set anywhere within the first band below. The Receive channel (6MHz Bandwidth) can be set anywhere within the second band below. TX frequencies are obtained by adding the values for TX IF and TX RF. RX frequencies are obtained by adding the values for RX IF and RX RF.

Table 3 RF Frequency Plan

TX RF Frequency	5.730GHz to 5.769GHz
RX RF Frequency	5.780GHz to 5.820GHz

4.2.2 RF SPECIFICATIONS

Table 4 TX-RF signal specification

TX Specifications	Comments	min	typ	max
TX Base band (BB) DAC input level (1.2Vp reference in unipolar DAC and 50Ω load)				1.2Vp
TX BB flatness over bandwidth	N/A (COFDM 12Bits//)			
TX BB dynamic range	N/A (COFDM 12Bits//)			
TX Phase noise	@ 10Khz Offset		-93dBc/Hz	-90dBc/Hz
TX output flatness over bandwidth	2dB slope @ 3dB cut-off with ALC compensation	-1dB	0dB	+1dB
TX output P1dB			+10dBm	
TX output VSWR	Measured at antenna port			1.4 (15.0dB)

Table 5 RX-RF signal Specification

RX Specifications	Comments	min	typ	max
RX BB ADC output level (1.2Vp reference in unipolar DAC and 50Ω load)	6MHz BB bandwidth			1.2Vp

RX BB flatness over bandwidth	N/A (COFDM 12Bits//)			
RX BBdynamic range	N/A (COFDM 12Bits//)			
RX Phase noise	@ 10Khz Offset		-93dBc/Hz	-90dBc/Hz
RX Dynamic range	6MHz		70dB	
RX input Level	6MHz	-90dBm		-20dBm
RX input VSWR	Measured at antenna port			1.4 (15.0dB)

Receiver is compensated with AGC Loop

4.2.3 DC POWER Specification

Table 6 DC Specifications

DC specifications	min	typ	max
DC voltage – Transceiver Analog Circuits	+9V	+12.0V	+15V
DC current (P1dB = 10mW)		0.50A	
DC Voltage – Micro-controller & Baseband circuits	3.2	3.3	3.4
DC Current for 3.3 VDC		0.450A	0.50A

4.2.4 DIGITAL I/O DESCRIPTIONS

Table 7 Digital IO

Digital signals	Descriptions	Level
DTX0- DTX11	TX Data input to DAC (12BITS Parallel)	3.3v (cmos)
DRX0- DRX11	RX Data output from ADC (12BITS Parallel)	3.3v (cmos)
DTXCLK	TX Sampling clock, variable with BB BW, **13Mhz to 15Mhz programmable (input)	3.3v (cmos)
DRXCLK	RX Sampling clock, variable with BB BW **13Mhz to 15Mhz programmable (input)	3.3v (cmos)
IIC_SCL	Serial communication Clock line (half-duplex)	3.3v (cmos)
IIC_SDA	Serial communication Data line (half-duplex)	3.3v (cmos)

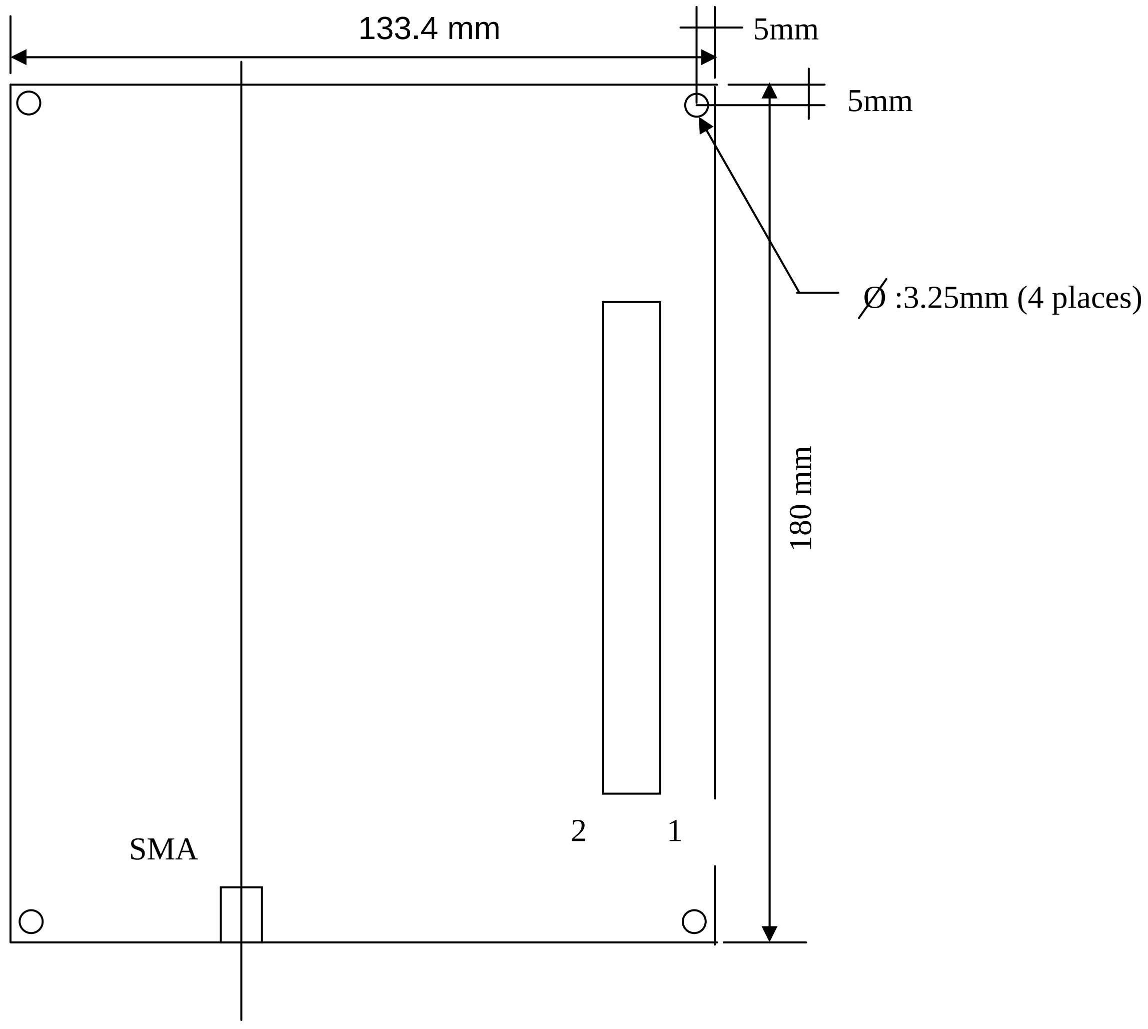
** Real time Bandwidth on demand

4.3 MECHANICAL SPECIFICATIONS

4.3.1 DIMENSIONS (See Figure)

Length	Approximately 7.08”
Width	Approximately 5.25”
Height	62mils (four Layers)
Weight	TBD (To be Defined)
Antenna port position	PCB - edge
Digital conn. position	PCB side as shown
Mounting hole positions	As shown or equivalent

Figure 4 Dimensions of Transceiver Card



4.3.2 Digital CONNECTORS

58 pins, SCCI Type III.

Table 8 Digital Connector Pin and Signal Assignment

Signal name	In / Out	Pin #	Pin #	In / Out	Signal name	Comments	
+12V	In	54	53	In	**+12V**	Vin 12VDC (830 mA max)	
+3.3V		52	51	In	**+3.3V**	+3.3VDC (500mA max),	
GND	In	50	49	In	**GND**		
DTX10	In	48	47	In	DTX11		
DTX8	In	46	45	In	DTX9	Transceiver BB Input data	
GND	In	44	43	In	**GND**		
DTX6	In	42	41	In	DTX7		
DTX4	In	40	39	In	DTX5		
GND	In	38	37	In	**GND**		
DTX2	In	36	35	In	DTX3		
DTX0	In	34	33	In	DTX1		
GND	In	32	31	In	**GND**		
PLLCLK	Out	30	29	In/Out	DTXCLK	Clock from RF Board	TX sample CLK
GND	In	28	27	In	**GND**		
CLK/UART0TX	In/Out	26	25	In/Out	DRXCLK	IIC CLK/UARTTX	RX sample CLK
D/UART0RX	In	24	23	In/Out	GND	IIC data/UARTRX	
GND	In	22	21	In	XRESET	RESET to Transceiver Micro-processor	
DRX0	Out	20	19	Out	DRX1		
DRX2	Out	18	17	Out	DRX3	Transceiver BB Output data	
GND	Out	16	15	Out	**GND**		
DRX4	Out	14	13	Out	DRX5		
DRX6	Out	12	11	Out	DRX7		
GND	Out	10	9	Out	**GND**		
DRX8	Out	8	7	Out	DRX9		
DRX10	Out	6	5	Out	DRX11		
TXMUTE	In	4	3	In	RXMUTE	To disable TX or RX channel	
GND	In	2	1	In	**GND**		

Connector to Antenna Cable: Right angle SMA female connector, 50Ω – Johnson Part No. 142-0701-551.

4.4 Radio Maintenance and Debugging

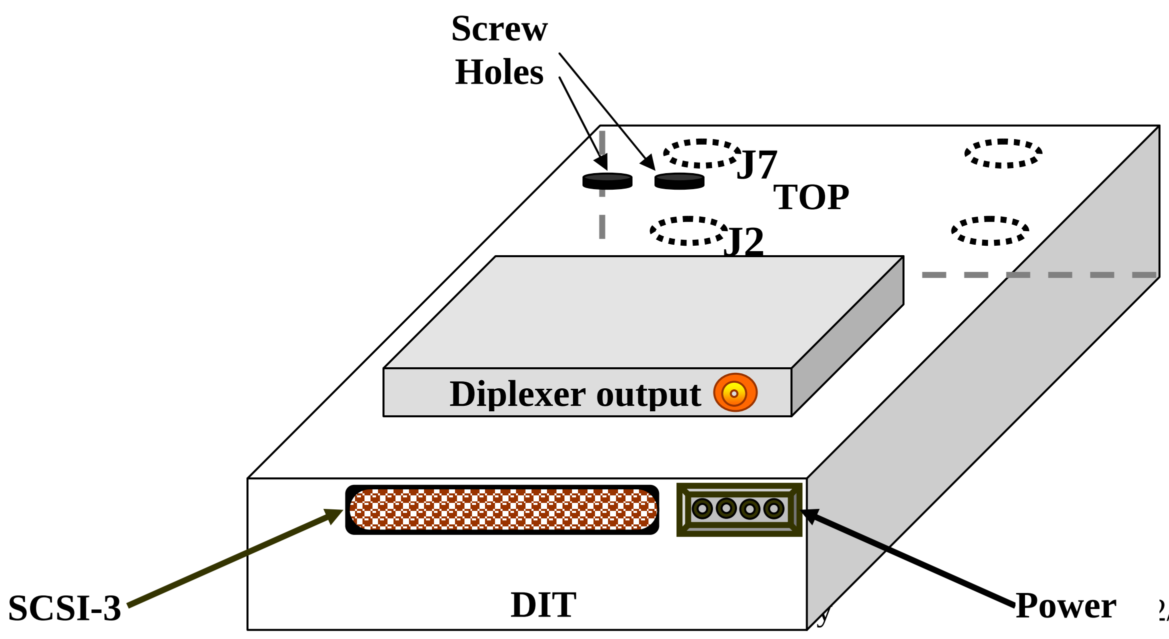


Figure 5 DIT Mechanical diagram

4.4.1 Carrier leakage adjustment procedure:

A step by step carrier leakage adjustment procedure can be described as follows:

- Power up the DIT and let it initialize for a minimum of 3 minutes.
- Verify the coax cable with MMCX connectors between the TX IF and TX RF sections of the unit (J7 or TX-IF-OUT_MIXER) are disconnected. Connect a Spectrum analyzer to the TX IF connector (J7).
- Remove the OFDM signal from the modem output by either setting the “tx_gain” to zero or disconnecting the SCSI-3 cable. Scan for the carrier signal between the frequencies 2300 MHz to 2500 MHz.
- With a small screwdriver adjust each of the potentiometers to their minimum possible value, through the two small holes on the BB to IF lid (DIT top). If the unit is operating correctly the spectrum analyzer should measure a -70 dBm output level or less.
- After adjustment, disconnect the coax cable to the spectrum analyzer and re-connect the coax cable with MMCX connectors between J7 and J2.

4.4.2 RF transmit power

The RF output level is fixed by the preset value of the TX_Gain at 65. The RF power can be measured at or after the diplexer (**NOTE: be sure to insert a 10 dB pad between the diplexer and spectrum analyzer to avoid damaging the transmitter**). Set the spectrum analyzer to VBW = 100Khz, RBW = 30KHz, and SWP = 5 ms. If the unit is operating correctly, the TX channel power should measure +10 dBm.

5 Compliances

5.1 FCC Compliance

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions; (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Changes or modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment.

NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:
-- Reorient or relocate the receiving antenna.
-- Increase the separation between the equipment and receiver.
-- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
-- Consult the dealer or an experienced radio/TV technician for help.

IMPORTANT!

To comply with FCC RF exposure requirements, this device must be located and operated to ensure a minimum separation distance of 20cm or more from a person's body. Other operating configurations should be avoided.

5.2 UL Compliance

Part 15

5.3 NEBs Compliance

GR-1093-core, G.867…