Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

WT21 BLUETOOTH HCI MODULE

DATA SHEET

Wednesday, 18 September 2013

Version 1.84

VERSION HISTORY

Version	Comment
1.0	Electrical characteristic added. Some minor updates.
1.1	Function of the regulator enable pin corrected. Some minor updates.
1.2	New template
1.3	Pinout fixed (GND pins 1 – 3 removed, pin 23 grounded). Layout guide updated.
1.4	Improved dimensions chapter
1.41	Table 5 fixed (pad types)
1.5	Footprint added
1.6	Footprint fixed. Pin number 3 (NC) added.
1.7	Recommended PCB land pattern for WT21-N added, power control and regulation updated. Certification information added
1.71	Information about qualified antenna types added
1.72	Minor updates to chapter 14.5
1.73	Physical dimensions corrected
1.74	UART directions clarified to table 5
1.8	Physical dimensions updated and recommended PCB land patterns corrected
1.81	Table 5 corrected.
1.82	Layout guide for WT21-N
1.83	Temperature range
1.84	MSL updated to MSL1 (was MSL3)

TABLE OF CONTENTS

WT21 *Bluetooth*® HCI Module

DESCRIPTION

WT21 is intended for *Bluetooth* applications where a host processor is capable of running the *Bluetooth* software stack. WT21 only implements the low level *Bluetooth* Host Controller Interface (HCI) but still offers advantages of a module - easy implementation and certifications.

APPLICATIONS:

- PCs and laptops
- PDAs
- Embedded systems

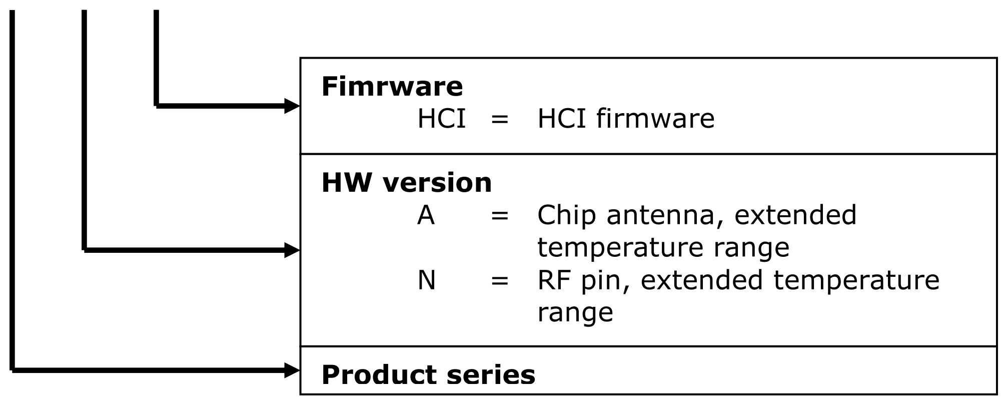
FEATURES:

- Fully Qualified Bluetooth v2.1 + EDR System
- Piconet and Scatternet Support
- Low Power Consumption
- 1,8V to 3,6V I/O Voltage
- Integrated 1,8V Regulator
- UART to 4 Mbaud
- SDIO (Bluetooth Type A) and CSPI Host Interfaces
- Deep-Sleep SDIO Operation
- Support for 802.11 Coexistence
- RoHS Compliant
- AuriStream Baseband Codec

Bluegiga Technologies Oy

1 Ordering Information

WT21-A-HCI



Fimrware

HCI	=	HCI firmware

HW version

A	=	Chip antenna, extended temperature range
N	=	RF pin, extended temperature range

Product series

2 Pinout and Terminal Description

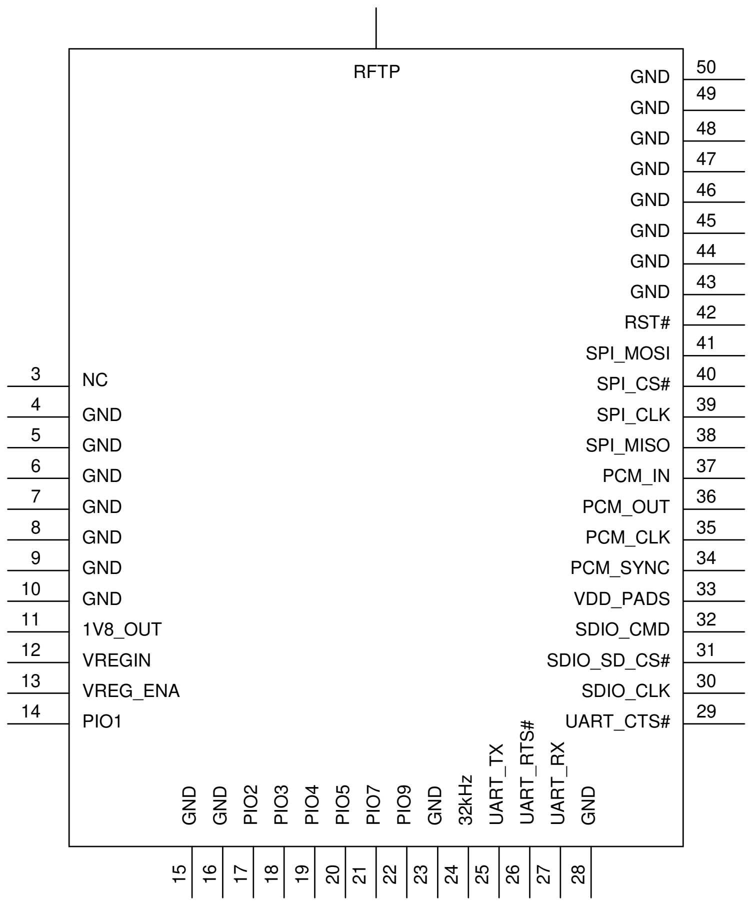


Figure 1: WT21 pin out

	PIN NUMBER	PAD TYPE	DESCRIPTION
NC	3	Not in use	Leave floating or connect to GND
RST#	42	Input, weak internal pull-up	Active low reset. Keep low for >5 ms to cause a reset
GND	23	GND	GND

Table 1: Terminal Descriptions

POWER SUPPLIES	**PIN NUMBER**	**DESCRIPTION**
VREGIN	12	Input for the internal 1,8V regulator
1v8_OUT	11	1,8V regulator output
VREG_ENA	13	Take high to enable internal voltage regulators
GND	4-10, 15-16, 28, 43-50	Ground
VDD_PADS	33	Positive supply for the digital interfaces

Table 2: Terminal Descriptions

PIO PORT	**PIN NUMBER**	**PAD TYPE**	**DESCRIPTION**
PIO[1]	14	Bi-directional, programmamble strength internal pull-down/pull-up	Programmamble input/output line
PIO[2]	17	Bi-directional, programmamble strength internal pull-down/pull-up	Programmamble input/output line
PIO[3]	18	Bi-directional, programmamble strength internal pull-down/pull-up	Programmamble input/output line
PIO[4]	19	Bi-directional, programmamble strength internal pull-down/pull-up	Programmamble input/output line
PIO[5]	20	Bi-directional, programmamble strength internal pull-down/pull-up	Programmamble input/output line
PIO[7]	21	Bi-directional, programmamble strength internal pull-down/pull-up	Programmamble input/output line
PIO[9]	22	Bi-directional, programmamble strength internal pull-down/pull-up	Programmamble input/output line

Table 3: Terminal Descriptions

SPI INTERFACE	PIN NUMBER	PAD TYPE	DESCRIPTION
PCM_OUT	36	Output, tri-state, weak internal pull-down	Synchronous data output
PCM_IN	37	Input, weak internal pull-down	Synchronous data input
PCM_SYNC	34	Bi-directional, weak internal pull-down	Synchronous data sync
PCM_CLK	35	Bi-directional, weak internal pull-down	Synchronous data clock

Table 4: Terminal Descriptions

SDIO/CSPI/UART Interfaces	PIN NUMBER	PAD TYPE	DESCRIPTION
SDIO_DATA[0]	25	Bi-directional, tri-state, weak internal pull-up	Synchronous data input/output
CSPI_MISO			CSPI data output
UART_TX			UART data output, active high
SDIO_DATA[1]	26	Bi-directional, weak internal pull-up	Synchronous data input/output
CSPI_INT			CSPI data input
UART_RTS#			UART request to send output, active low
SDIO_DATA[2]	27	Bi-directional, weak internal pull-up	Synchronous data input/output
UART_RX			UART data input, active high
SDIO_DATA[3]	29	Bi-directional, weak internal pull-up	Synchronous data input/output
CSPI_CS#			Chip select for CSR Serial Peripheral Interface, active low
UART_CTS#			UART clear to send input, active low
SDIO_CLK	30	Bi-directional, weak internal pull-up	SDIO clock
CSPI_CLK			CSPI clock
SDIO_SD_CS#	31	Bi-directional, weak internal pull-up	SDIO chip select to allow SDIO accessess
SDIO_CMD	32	Bi-directional, weak internal pull-up	SDIO data input
CSPI_MOSI			CSPI data input

Table 5: Terminal Descriptions

SPI INTERFACE	**PIN NUMBER**	**PAD TYPE**	**DESCRIPTION**
SPI_MOSI	41	Weak internal pull-down	SPI data input
SPI_CS#	40	Bi-directional, weak internal pull-down	Chip select for Serial Peripheral Interface, active low
SPI_CLK	39	Bi-directional, weak internal pull-down	SPI clock
SPI_MISO	38	Output, tri-state, weak internal pull-down	SPI data output

Table 6: Terminal Descriptions

3 Microcontroller, Memory and Baseband Logic

3.1 AuriStream CODEC

The AuriStream CODEC works on the principle of transmitting the delta between the actual value of the signal and a prediction rather than the signal itself. Hence, the information transmitted is reduced along with the power requirement. The quality of the output depends on the number of bits used to represent the sample.

The inclusion of AuriStream results in reduced power consumption compared to a CVSD implementation when used at both ends of the system.

3.1.1 AuriStream CODEC Requirements

AuriStream supports the following modes of operation:

	fs	Bit Rate (kbps)							
		16	20	24	32	40	48	64	80
G726	8 kHz	(✓)		✓	✓	✓			
	10 kHz				(✓)		(✓)	(✓)	(✓)
G722	8 kHz		(✓)	(✓)	(✓)				
	16 kHz					(✓)	✓	✓	

Table 7: AuriStream Supported Bitrates

Table Key:

= Standard Mode

(✓) = Optional Mode

Where possible, AuriStream shares hardware between the encoder and decoder as well as the G726 and G722 implementations of the standard. The 40kbs and 20kbs modes of the G722 codec are specific to CSR.

The AuriStream module will be required to support the 3Mbps stream transmitted by the BT radio. The worst-case scenario arises when the AuriStream block is configured as 16kbps at 8 kHz, which equates to 2 bits per sample, giving a worst-case symbol rate at the input to the AuriStream block of 1.5Msps to sustain the transmitted bit stream.

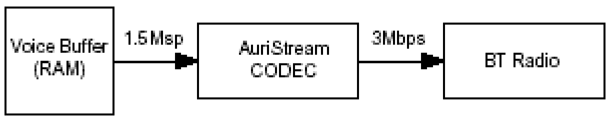


Figure 2: AuriStream CODEC and the BT Radio

3.1.2 AuriStream Hierarchy

The AuriStream CODEC is positioned in parallel with the CVSD CODEC as shown in Figure 4.

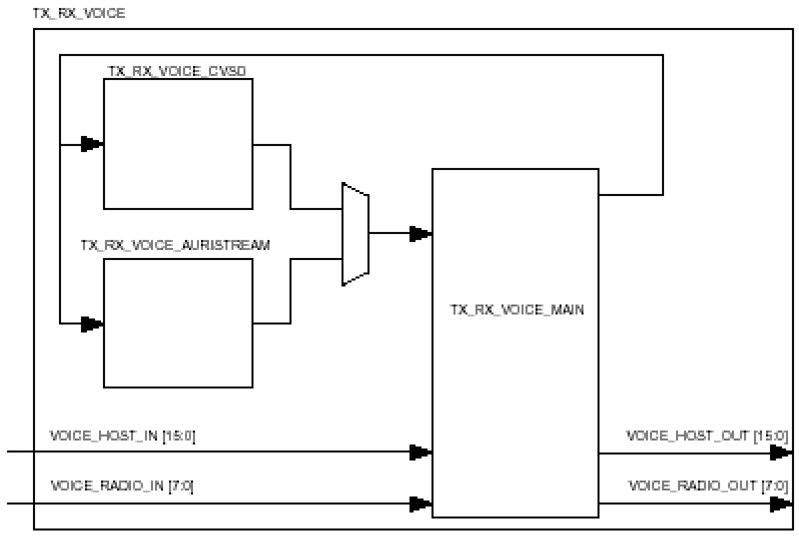


Figure 3: AuriStream CODEC and the CVSD CODEC

The AuriStream CODEC is controlled by the TX_RX_VOICEmain block and the processor. Raw data from the host is read from the MMU by the transmit block. This data is fed via the TX_RX_VOICE_MAIN module to the required CODEC, the encoded data is then fed back to the transmit block for broadcast over the Bluetooth interface. During reception, the data is sourced from the radio and applied to the required CODEC. The decoded data is then stored back to RAM by the bluetooth receiver.

3.2 Memory Managements Unit

The Memory Management Unit (MMU) provides a number of dynamically allocated ring buffers that hold the data that is in transit between the host and the air. The dynamic allocation of memory ensures efficient use of the available Random Access Memory(RAM) and is performed by a hardware MMU to minimise the overheads on the processor during data/voice transfers.

3.3 Burst Mode Controller

During transmission the Burst Mode Controller(BMC) constructs a packet from header information previously loaded into memory-mapped registers by the software and payload data/voice taken from the appropriate ring buffer in the RAM. During reception, the BMC stores the packet header in memory-mapped registers and the payload data in the appropriate ring buffer in RAM. This architecture minimises the intervention required by the processor during transmission and reception.

3.4 Physical Layer Hardware Engine DSP

Dedicated logic is used to perform the following:

- Forward error correction
- Header error correction
- Cyclic redundancy check
- Encryption
- Data whitening
- Access code correlation
- Audio transcoding

The following voice data translations and operations are performed by firmware:

- A-law/µ-law/linear voice data (from host)
- A-law/µ-law/Continuously variable Slope Delta (CVSD) (over the air)
- Voice interpolation for lost packets
- Rate mismatches

The hardware supports all optional and mandatory features of Bluetooth v2.1 + EDR including AFH and eSCO.

3.5 WLAN Coexistence

Dedicated hardware is provided to implement a variety of coexistence schemes. Channel skipping AFH, priority signalling, channel signalling and host passing of channel instructions are all supported. The features are configured in firmware.

For more information contact Buegiga technical support.

3.6 Configurable I/O Parallel Ports

lines of programmable bi-directional input/outputs (I/O) are provided. PIO[1: 5, 7, 9] are powered from VDD_PADS.

PIO lines can be configured through software to have either weak or strong pull-ups or pull-downs. All PIO lines are configured as inputs with weak pull-downs at reset.

Any of the PIO lines can be configured as interrupt request lines or as wake-up lines from sleep modes.

Bluegiga cannot guarantee that the PIO assignments remain as described. Refer to the relevant software release note for the implementation of these PIO lines, as they are firmware build-specific.

4 Clock Generation

WT21 uses an internal 26 MHz crystal as a Bluetooth reference clock. All WT21 internal digital clocks are generated using a phase locked loop, which is locked to the 26 MHz reference clock.

Also supplied to the digits is a watchdog clock, for use in low power modes. This uses a frequency of 32.768kHz from CLK_32K, or an internally generated reference clock frequency of 1kHz, determined by PSKEY_DEEP_SLEEP_EXTERNAL_CLOCK_SOURCE.

The use of the watchdog clock is determined with respect to Bluetooth operation in low power modes.

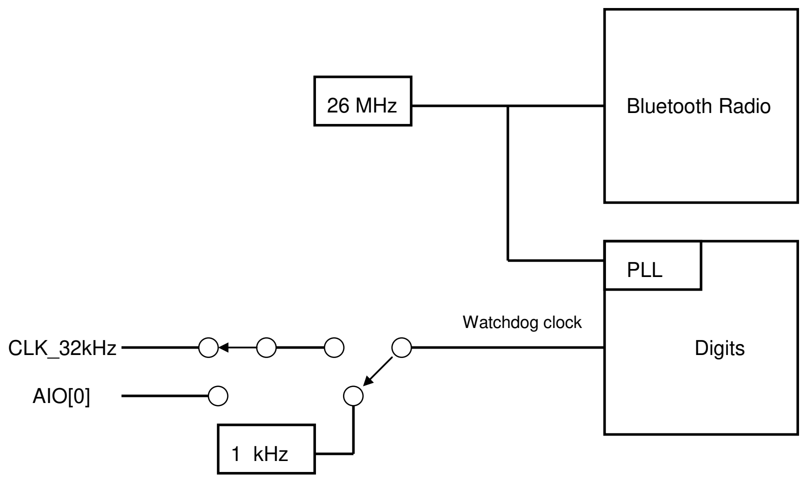


Figure 4: Clock Architecture

4.1 32kHz External Reference Clock

A 32kHz clock can be applied to CLK_32K, using PSKEY_DEEP_SLEEP_EXTERNAL_CLOCK_SOURCE.

The CLK_32K pad is in the VDD_PADS domain with all the other digital I/O pads and is driven between levels specified in Section 11.3.4.

5 Serial Peripheral Interface (SPI)

5.1 WT21 Serial Peripheral Interface (SPI)

SPI is used for debuging primarily. This section details the considerations required when interfacing to WT21 via the SPI.

Data may be written or read one word at a time or the auto increment feature may be used to access blocks.

5.2 Instruction Cycle

WT21 is the slave and receives commands on SPI_MOSI and outputs data on SPI_MISO. Table 8 shows the instruction cycle for an SPI transaction.

1	Reset the SPI interface	Hold SPI_CS# high for two SPI_CLK cycles
2	Write the command word	Take SPI_CS# low and clock in the 8 bit command
3	Write the address	Clock in the 16-bit address word
4	Write or read data words	Clock in or out 16-bit data word(s)
5	Termination	Take SPI_CS# high

Table 8: Instruction Cycle for an SPI Transaction

With the exception of reset, SPI_CS# must be held low during the transaction. Data on SPI_MOSI is clocked into the WT21 on the rising edge of the clock line SPI_CLK. When reading, WT21 replies to the master on SPI_MISO with the data changing on the falling edge of the SPI_CLK. The master provides the clock on SPI_CLK. The transaction is terminated by taking SPI_CS# high.

Sending a command word and the address of a register for every time it is to be read or written is a significant overhead, especially when large amounts of data are to be transferred. To overcome this WT21 offers increased data transfer efficiency via an auto increment operation. To invoke auto increment, SPI_CS# is kept low, which auto increments the address, while providing an extra 16 clock cycles for each extra word to be written or read.

5.2.1 Writing to the Device

To write to WT21, the 8-bit write command (00000010) is sent first (C[7:0]) followed by a 16-bit address (A[15:0]). The next 16-bits (D[15:0]) clocked in on SPI_MOSI are written to the location set by the address (A). Thereafter for each subsequent 16-bits clocked in, the address (A) is incremented and the data written to consecutive locations until the transaction terminates when SPI_CS# is taken high.

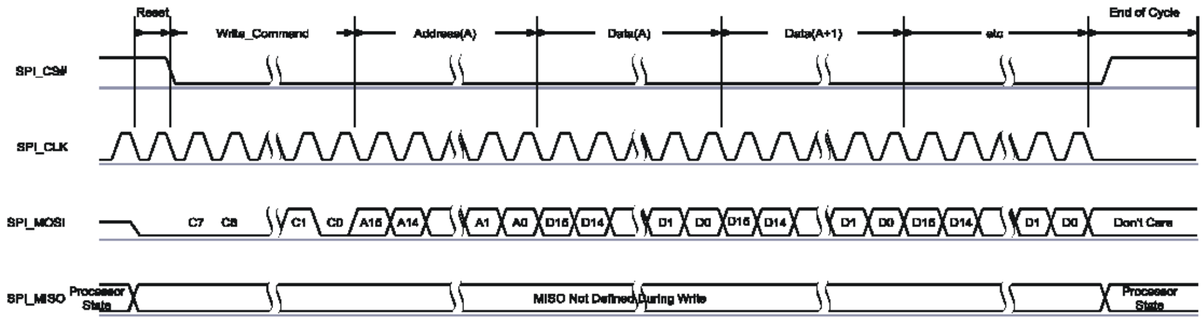


Figure 5: SPI Write Operation

5.2.2 Reading from the Device

Reading from WT21 is similar to writing to it. An 8-bit read command (00000011) is sent first (C [7:0]), followed by the address of the location to be read (A[15:0]). WT21 then outputs on SPI_MISO a check word during T[15:0] followed by the 16-bit contents of the addressed location during bits D[15:0].

The check word is composed of {command, address [15:8]}. The check word may be used to confirm a read operation to a memory location. This overcomes the problems encountered with typical serial peripheral interface slaves, whereby it is impossible to determine whether the data returned by a read operation is valid data or the result of the slave device not responding.

If SPI_CS# is kept low, data from consecutive locations is read out on SPI_MISO for each subsequent 16 clocks, until the transaction terminates when SPI_CS# is taken high.

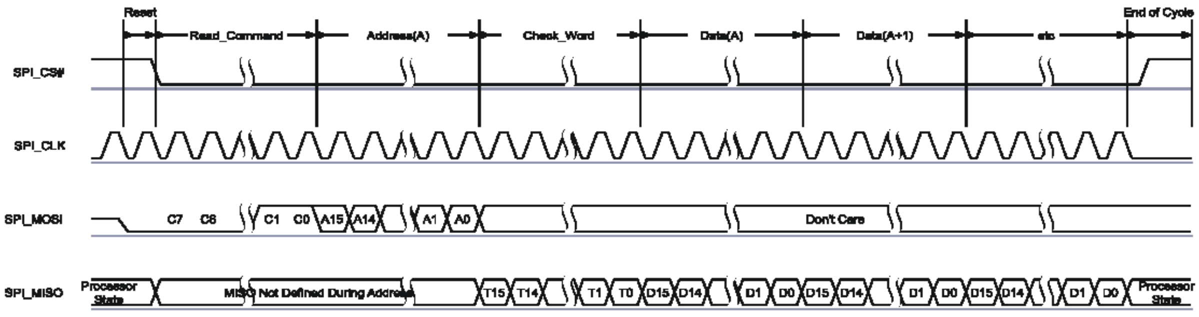


Figure 6: SPI Read Operation

5.2.3 Multi-Slave Operation

WT21 should not be connected in a multi-slave arrangement by simple parallel connection of slave MISO lines. When WT21 is deselected (SPI_CS# = 1), the SPI_MISO line does not float. Instead, WT21 outputs 0 if the processor is running or 1 if it is stopped.

6 Host Interfaces

6.1 Host Selection

The MCU selects the UART/SDIO interfaces by reading PIO[4] at boot-time. When PIO[4] is high, the SDIO interface is enabled; when PIO[4] is low, the UART is enabled.

If in UART mode, the MCU selects the UART transfer protocol automatically using the unused SDIO pins shown in Table 9

SDIO_CLK	SDIO_CMD	Protocol
0	0	bcsp
0	1	h4
1	0	h4ds
1	1	h5

Table 9: SDIO_CLK and SDIO_CMD transfer Protocols

6.2 UART Interface

This is a standard UART interface for communicating with other serial devices.

WT21 UART interface provides a simple mechanism for communicating with other serial devices using the RS232 protocol.

Note:

WT21 uses RS232 protocol, but voltage levels are 0V to VDD_PADS (requires external RS232 transceiver chip.

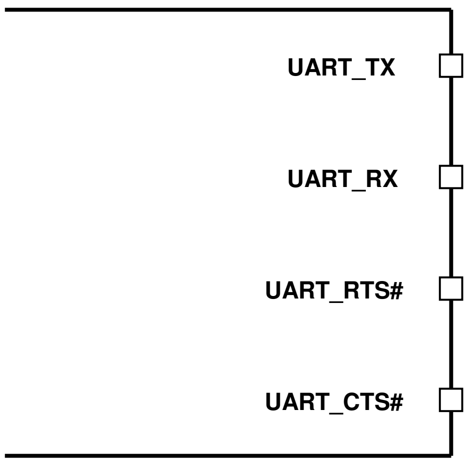


Figure 7: Universal Asynchronous Receiver

Four signals implement the UART function, as shown in Figure 8. When WT21 is connected to another digital device, UART_RX and UART_TX transfer data between the two devices. The remaining two signals, UART_CTS and UART_RTS, can be used to implement RS232 hardware flow control where both are active low indicators.

UART configuration parameters, such as baud rate and packet format, are set using WT21 firmware.

Note:

An accelerated serial port adapter is required to communicate with the UART at maximum baud rate using a standard PC.

Parameter		Possible Values
Baud Rate	Minimum	1200 baud (≤2%Error)
		9600 baud (≤1%Error)
	Maximum	4Mbaud (≤1%Error)
Flow Control		RTS/CTS or None
Parity		None, Odd or Even
Number of Stop Bits		1 or 2
Bits per Byte		8

Table 10: Possible UART Settings

Note:

Baud rate is the measure of symbol rate i.e. , the number of distinct symbol changes (signaling events) made to transmission medium per second in a digitally modulated signal.

The UART interface is capable of resetting WT21 on reception of a break signal. A break is identified by a continuous logic low (0V) on the UART_RX terminal, as shown in Figure 9. If tBRKis longer than the value, defined by the PSKEY_HOSTIO_UART_RESET_TIMEOUT, (0x1a4), a reset occurs. This feature allows a host to initialise the system to a known state. Also, WT21 can emit a break character that may be used to wake the host. By default this feature is disabled and it is recommended to enable it by setting PSKEY_HOSTIO_UART_RESET_TIMEOUT.

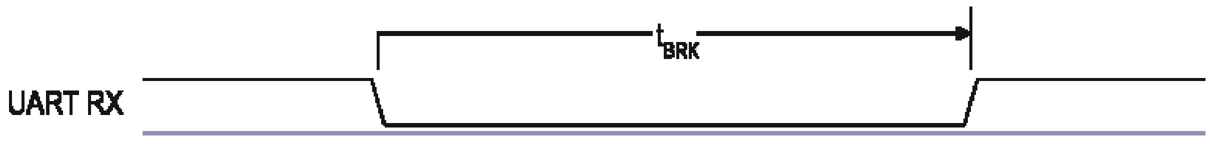


Figure 8: Break Signal

Table 11 shows a list of commonly used baud rates and their associated values for the PSKEY_UART_BAUDRATE (0x1be). There is no requirement to use these standard values. Any baud rate within the supported range can be set in the PS Key according to the formula in Equation XXX.

$$\text{Baud Rate} = \frac{\text{PSKEY\_UART\_BAUDRATE}}{0.004096}$$

Equation 1: Baud Rate

Baud Rate	Persistent Store Value		Error
	Hex	Dec	
1200	0x0005	5	1.73%
2400	0x000a	10	1.73%
4800	0x0014	20	1.73%
9600	0x0027	39	-0.82%
19200	0x004f	79	0.45%
38400	0x009d	157	-0.18%
57600	0x00ec	236	0.03%
76800	0x013b	315	0.14%
115200	0x01d8	472	0.03%
230400	0x03b0	944	0.03%
460800	0x075f	1887	-0.02%
921600	0x0ebf	3775	0.00%
1382400	0x161e	5662	-0.01%
1843200	0x1d7e	7550	0.00%
2764800	0x2c3d	11325	0.00%

Table 11: Standard Baud Rates

6.2.1 UART Configuration While Reset is Active

The UART interface for WT21 is tri-state while the chip is being held in reset. This allows the user to daisy chain devices onto the physical UART bus. The constraint on this method is that any devices connected to this bus must tri-state when WT21 reset is de-asserted and the firmware begins to run.

7 CSR Serial Peripheral Interface (CSPI)

The CSPI is a host interface which shares pins with the SDIO. It has been defined by CSR with the intention of producing a very simple interface. This has two advantages:

- It allows maximum compatibility with the possible host drivers
- It minimizes the host software effort needed to form that data to be sent (e.g., by removing the need to calculate CRCs)

This host interface allows an external host to control the Bluecore, using a CSR defined protocol built upon a 4-wire SPI bus.

Note:

The CSPI is entirely separated from the debug Serial Peripheral Interface

The CSPI allows access to the following:

- Function 0 registers
- Bluetooth Acceleration Registers
- MCU IO Registers
- Bluetooth MMU port

The CSPI is a third protocol available for the host to transfer data into the Bluecore and shares pins with the other SDIO protocols.

MMU buffers are accessed using burst read/writes. The command and address fields are used to select the correct buffer. The CSPI is able to generate an interrupt to the host when a memory access fails. This interrupt line is shared with the SDIO functions.

Table 12 shows the mapping of SDIO pins onto the CSPI functions when CSPI is enabled.

Pin	CSPI Function	Direction	Description
SDIO_DATA3	CSB	I	Chip Select
SDIO_CMD	MOSI	I	Master Out Slave In
SDIO_DATA0	MISO	O	Master In Slave Out
SDIO_CLK	CLK	I	Clock
SDIO_DATA1	INT	O	Interrupt

Table 12: SDIO Mapping to CSPI Functions

The CSPI Interface is an extension of the basic SPI Interface, with the access type determined by the following fields:

- 8-bit command
- 24-bit address
- 16-bit burst length (optional). Only applicable for burst transfers into or out of the MMU

7.1.1 CSPI Read/Write Cycles

Register read/write cycles are used to access Function 0, Bluetooth acceleration and MCU registers.

Burst read/write cycles are used to access the MMU.

7.1.2 CSPI Register Write Cycle

The command and address are locked into the slave, followed by 16bits of write data. An Error Byte is returned on the MISO signal indicating whether or not the transfer has been successful.

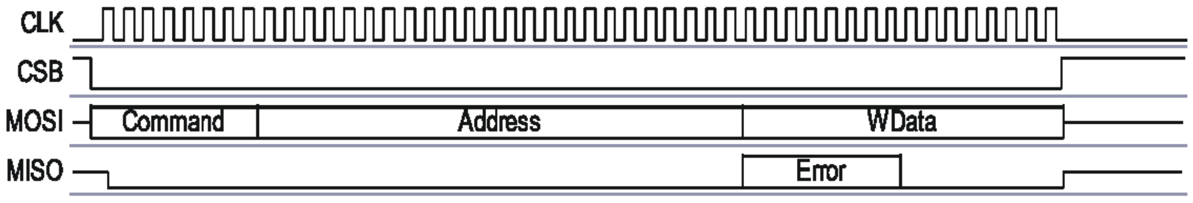


Figure 9: CSPI Register Write Cycle

7.1.3 CSPI Register Read Cycle

The command and address field are clocked into the slave, the slave then returns the following:

- Bytes of badding data (MISO held low)
- Error byte
- 16-bits of read data

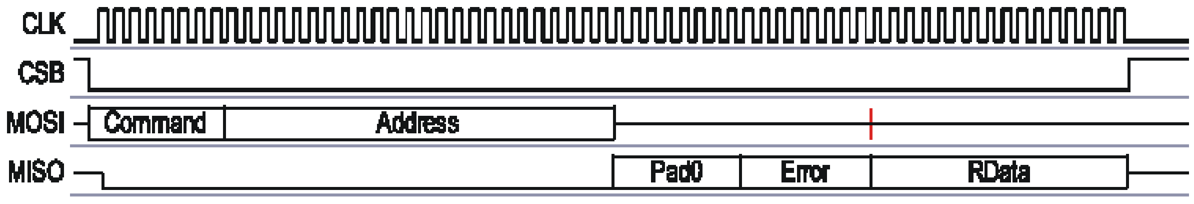


Figure 10: CSPI Register Read Cycle

7.1.4 CSPI Register Burst Write Cycle

Burst transfers are used to access the MMU buffers. They cannot be used to access registers. Burst read/write cycles are selected by setting the nRegister/Burst bit in the command field to 1.

Burst transfers are byte orientated, have a minimum length of 0 bytes and a maximum length of 64kbytes. Setting the length field to 0 results in no data being transferred to or from the MMU.

As with a register access, the command and address fields are transferred first. There is an optional length field transferred after the address. The use of the length field is controlled by the LengthFieldPresent bit in the Function 0 registers, which is cleared on reset.

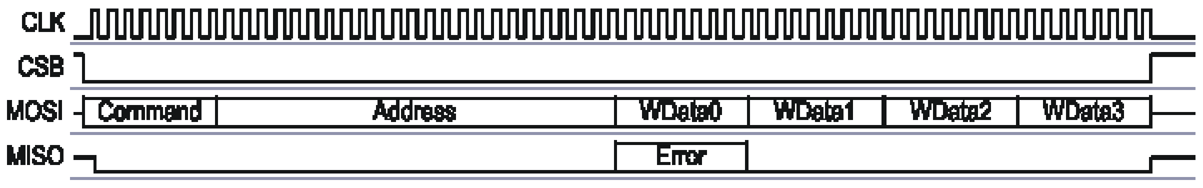


Figure 11: CSPI Burst Write Cycle

7.1.5 CSPI Register Read Cycle

Burst reads have a programmable amount of padding data that is returned by the slave. 0-15 bytes are returned as defined in the BurstPadding register. Following this the Error byte is returned followed by the data. Once the transfer has started, no further padding is needed.

A FIFO within SDIO_TOP will pre-fetch the data. The address is not retransmitted, and is auto-updated within the slave.

The length field is transmitted if LengthFieldPresentin the Function 0 registers is set. In the absence of a length field the CSB signal is used to indicate the end of the burst.

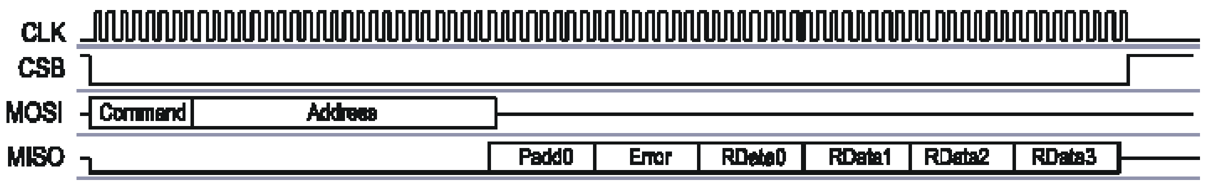


Figure 12: CSPI Burst Read Cycle