

# **SDR-14 Interface Specification**

**Ver. 1.00**

**April 20, 2004**

**MoeTronix  
[www.moetronix.com](http://www.moetronix.com)**

# Table of Contents

<b>1. Purpose</b>	<b>3</b>
<b>2. Basic Protocol Concepts</b>	<b>3</b>
<b>3. Message Block Format</b>	<b>5</b>
3.1. <i>Detailed Description of the Message Block Types and Their Purpose</i>	7
3.1.1. Set Control Item	7
3.1.2. Request Control Item	7
3.1.3. Request Control Item Range	7
3.1.4. Response to Set or Request Current Control Item	7
3.1.5. Unsolicited Control Item	7
3.1.6. Response to Request Control Item Range	7
3.1.7. Data Item Messages	7
3.2. <i>The ACK and NAK Messages and Their Purpose</i>	8
<b>4. SDR-14 Architecture</b>	<b>9</b>
4.1. <i>Block Diagram</i>	9
4.2. <i>Functionality</i>	9
4.3. <i>FTDI USB interface</i>	9
<b>5. SDR-14 Control Item Definitions</b>	<b>9</b>
5.1. <i>General Control Items</i>	10
5.1.1. Target Name	10
5.1.2. Target Serial Number	10
5.1.3. Interface Version	10
5.1.4. Hardware/Firmware Versions	11
5.1.5. Status/Error Code	11
5.1.6. Status String	11
5.2. <i>SDR-14 Receiver Control Items</i>	12
5.2.1. Receiver State	12
5.2.2. Receiver Frequency	17
5.2.3. SDR-14 A/D Input Sample Rate	17
5.2.4. RF Gain	17
5.3. <i>SDR-14 Data Item Definitions</i>	18
5.3.1. SDR-14 Output Data Item 0	18
5.3.2. AD6620 Setup Data Item 1	18

# 1. Purpose

This specification describes the protocol used to communicate with the SDR-14 digital receiver. The protocol is based on a more generic protocol called ASCP(Amateur Station Control Protocol).

# 2. Basic Protocol Concepts

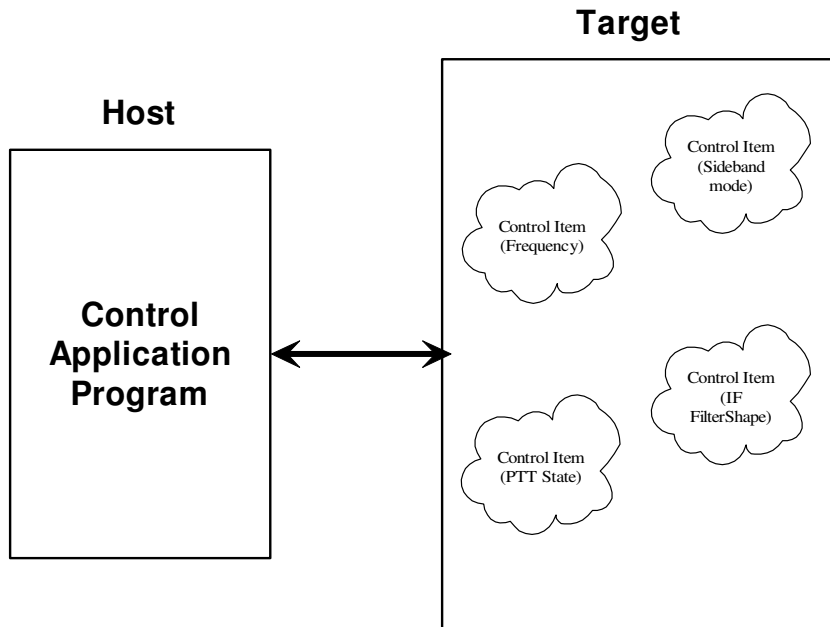
Definitions used in this specification:

In the case of the SDR-14, the host is the PC and the Target device is the SDR-14 hardware.

**Host** == The main initiator of communications. Typically would be a PC or other computer system such as a custom user interface controller.

**Target** == The device that is to be controlled or monitored by the Host.

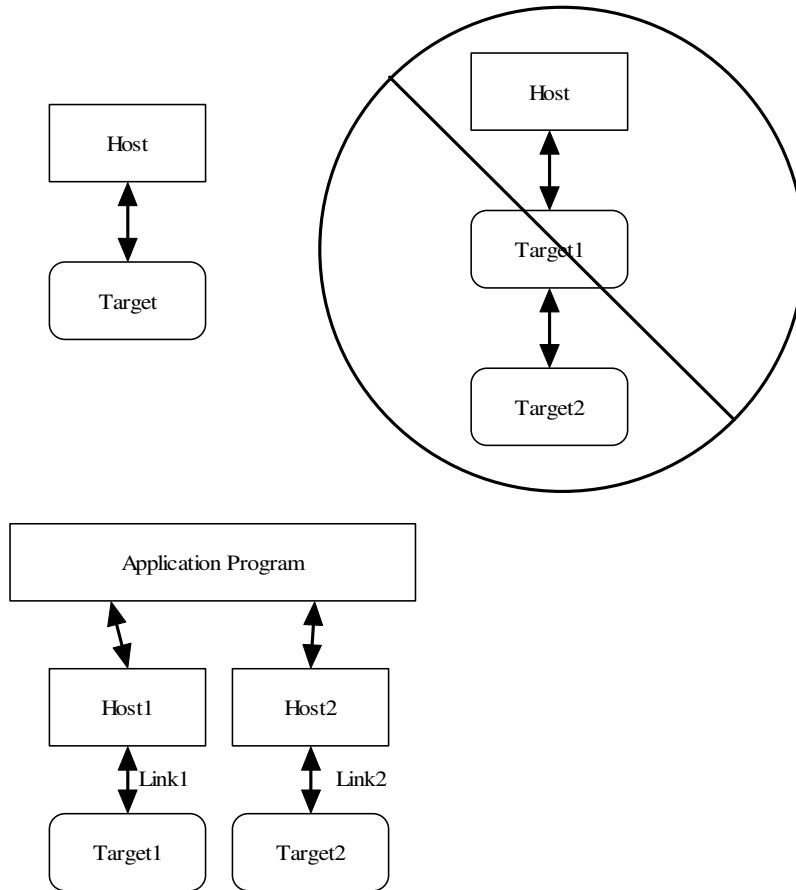
**Control Item** == The value, setting, or state of the target that is to be controlled or monitored by the Host. For example Frequency, Antenna direction, modulation mode, transmitter state, etc.



**Data Item** == Digital data associated with the received or transmitted signal. This could be digitized audio, IF, frequency domain data(FFT), information data such as text or AX25 data, etc.

**Message Block** == A contiguous block of bytes comprising a single Control Item or Data Item transfer from target to host or host to target.

To simplify the protocol, the link can only comprise one host and one target. The Host is the only one that can set or request Control items. This means a Target device cannot connect to another target device or daisy chain to other targets. The Host can control multiple Targets by utilizing multiple links such as USB endpoints or multiple TCP/IP Sockets.



The protocol allows a Target to send unsolicited Control Item messages to the Host. This is desirable for updating the Host when a something changes in the Target without the need for polling by the Host. An example would be when a user changes the frequency of the radio using the radio's frequency knob. The target can send the updated frequency as it occurs without requiring the Host to ask for it.

Message blocks contain the block length in the message header. This is useful to aid in decoding messages as well as being able to support variable length Control Items. For example, a Control Item containing the text string for the Target's manufacturer and model number can be different lengths.

The protocol contains a mechanism for obtaining information about the range and resolution of a Control Item. This allows the Host to obtain the limits of Control Items for the particular target device. For example, the frequency ranges of the target as well as the frequency step size can be obtained without any knowledge of the model or manufacturer of the radio. This capability is key in being able to write a generic application interface program without having to maintain a list of all possible radios and devices that may be connected.

Target devices are not required to implement all the functionality of the protocol. A simple Target device that turns on and off a light need only implement the single Control Item message associated with it. This allows micro controllers with limited capability to be used.

The Data Item message blocks allow various raw data blocks to be sent and received along with the Control Items over the same physical link. The Header type allows up to 4 logical channels of data to be specified in each direction. This permits sending digitized audio, digitized I/Q IF data, etc. to and from a target over the same physical connection.

Note that there is no synchronization or error handling mechanism in this protocol. This layer of protocol assumes that the block synchronization and error handling is done at a lower level. This is a reasonable assumption since

Ethernet, USB, IEEE 1394, and most other modern physical links provide error recovery. The exception is RS232 serial links. This protocol will work over it as-is but if any data corruption is anticipated, then some simple method such as "SLIP" should be used to frame the data bytes and perhaps add error detection as well. A little "sanity" checking of message block parameters can be used as a simple error detection scheme and allow the RS232 system to eventually sync back up. Adding a timeout will handle the case where a link is disconnected in the middle of a transfer. Since RS232 serial links are being phased out, it doesn't make sense to burden a protocol to handle such a legacy physical link.

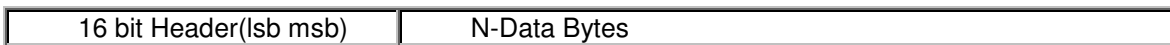
### 3. Message Block Format

The basic message structure starts with a 16 bit header that contains the length of the block in bytes and also a 3 bit type field. If the message is a Control Item, then a 16 bit Control Item code follows the header and contains the code describing the object of the message block. This is followed by an optional number of parameter or data bytes associated with this message. The byte order for all fields greater than 8 bits is "Little Endian" or least significant byte first.

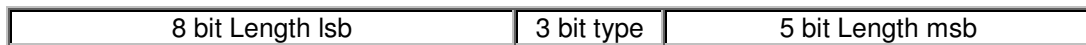
Control Item Message block format:



Data Item Message block format:



The 16 bit header is defined as follows:



The 13 bit Length parameter value is the total number of bytes in the message including this header. The range of the message Length is 0 to 8191 bytes.

A special case for Data Items is that a message length of Zero is used to specify an actual message length of 8194 bytes(8192 data bytes + 2 header bytes). This allows data blocks of a power of 2 to be used which is useful in dealing with FFT data.

The message type field is used by the receiving side to determine how to process this message block. It has a different meaning depending upon whether the message is from the Host or Target.

<b>3 bit Msg Type field</b>	<b>Message Source</b>	<b>Message Type</b>
000	Host	Set Control Item
001	Host	Request Current Control Item
010	Host	Request Control Item Range
011	Host	Data Item ACK from Host to Target
100	Host	Host Data Item 0
101	Host	Host Data Item 1
110	Host	Host Data Item 2
111	Host	Host Data Item 3
000	Target	Response to Set or Request Current Control Item
001	Target	Unsolicited Control Item
010	Target	Response to Request Control Item Range
011	Target	Data Item ACK from Target to Host
100	Target	Target Data Item 0
101	Target	Target Data Item 1
110	Target	Target Data Item 2
111	Target	Target Data Item 3

### **3.1. Detailed Description of the Message Block Types and Their Purpose**

#### **3.1.1. Set Control Item**

This Message type is sent from the Host to the Target requesting that the Target change the specified Control Item to the new value supplied in this message. A request to change to a new frequency would be an example of this type of message. The Target must respond to this message.

#### **3.1.2. Request Control Item**

This Message type is sent from the Host to the Target requesting that the Target respond with its current state or value of the specified Control Item of this message. A request to get the current S-meter reading would be an example of this message type. The Target must respond to this message.

#### **3.1.3. Request Control Item Range**

This Message type is sent from the Host to the Target requesting that the Target respond with the acceptable range of values of the Control Item supplied in this message. A request for the targets frequency range(s) and step sizes would be an example of this message type. The Target must respond to this message.

#### **3.1.4. Response to Set or Request Current Control Item**

This Message type is sent from the Target to the Host in response to a request from the Host to either set or just return the current value of the Control Item supplied in this message. This message contains the current value of the Control Item. It is sent in response to either the "Set Control Item" or "Request Control Item" message.

#### **3.1.5. Unsolicited Control Item**

This Message type can be sent from the Target to the Host without any request from the Host. It contains the current value of the Control Item supplied in this message. This message can be sent at any time to the Host. It can be used to update the Host to any changes that have occurred in the Target Control Items. An example would be if the user changed frequency using the Targets frequency knob, then the Target could send the new Control Item value to the Host without having to wait for the Host to ask for it. There is no response back from the Host when this message is received.

#### **3.1.6. Response to Request Control Item Range**

This Message is sent from the Target to the Host in response to a "Request Control Item Range" message from the Host. It contains the allowable range and step size of the Control Item supplied in this message.

#### **3.1.7. Data Item Messages**

Data Item message allow data messages to be allocated to different logical "channels". Different types of data blocks may be interleaved together and this mechanism allows each end to keep the data separated. For example, Data Item 2 blocks may be digitized audio from a Target receiver that needs to be processed and sent to a soundcard speaker. Data Item 3 Blocks may be spectral data from an FFT inside the Target receiver that needs to be sent to the Host applications display screen.

The current scheme allows up to four different logical channels for each data direction.

### **3.2. The ACK and NAK Messages and Their Purpose**

A "NAK" message is a 16 bit header without a Control Item or parameters (Message length of 2) . When the NAK message block is returned by the Target, it indicates that the specified Control Item is not supported. This allows a target to implement only the Control Items it actually needs. Any Host message requesting an unimplemented Control Item will be returned the NAK message. The Host can then exclude this Control Item from its list of Items to control or monitor.

As an example, suppose a Host requests the elevation setting from a rotor Target controller that only supports azimuth readings. The Target controller would just return the NAK header.

Implementation on the Target side is easily done by simply decoding only the Control Item messages that it supports and returning the NAK for all others.

On the Host side, one could initially poll the Target for all the Control Items it may use and then tag the ones that return NAK for exclusion.

This dynamically allocated feature set allows application software to determine on the fly what capabilities are available without any prior knowledge of the Target device.

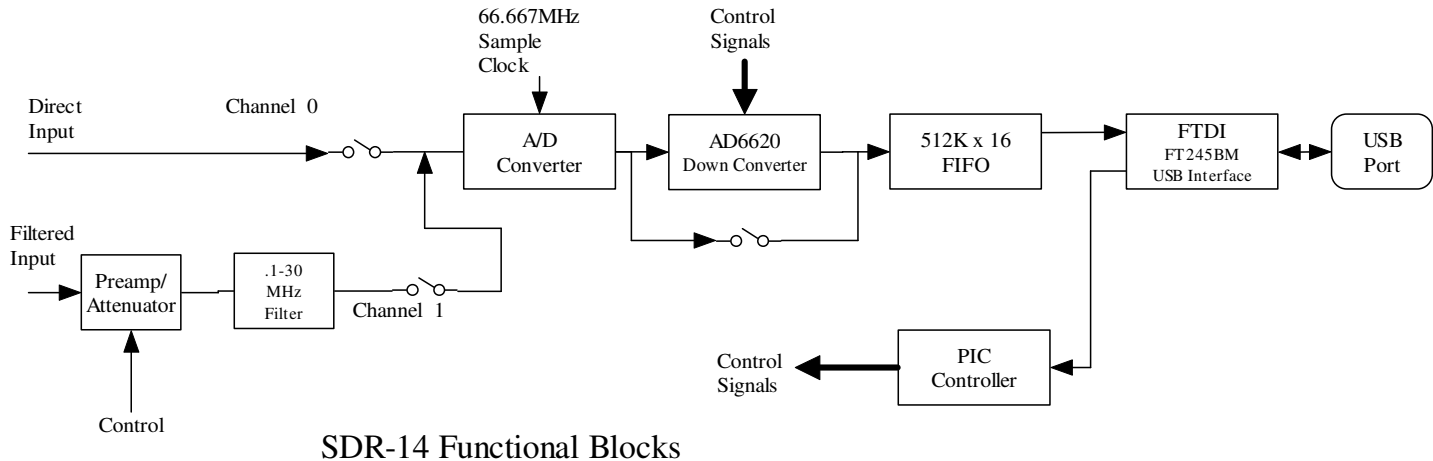
A Data Item "ACK" message is a 16 bit header with a Message Type = 011b with a single parameter byte specifying the Data Item (0 to 3). The 16 bit header is a fixed value (0000 0011 0110 0000 = 0x0360 ). The parameter byte following the header specifies which Data Item block that is being ACK'd.

For example if the Target received a block of Data Item 2 data correctly it could send the following back to the Host:  
[03][60] [02]

The ACK response messages is to provide handshaking to data item transfers. If a data item message is received correctly then an ACK response message could be sent back to the sender. This implementation is optional as one may want to stream data without error checking or only ACK periodically the data stream.

## 4. SDR-14 Architecture

### 4.1. Block Diagram



### 4.2. Functionality

The SDR-14 consists of a 14 bit A/D converter sampling at 66.66667MHz, a digital down converter, a FIFO, and a USB interface. There are two selectable input paths going to the A/D converter.

1. Direct input with no filtering or gain.
2. A .1 to 30MHz filtered path with a preamplifier.

The 14 bit data from the A/D converter is then either passed straight on as real data to the FIFO or is used as the input to an Analog Devices (<http://www.analog.com/>) AD6620 digital down converter chip. This chip contains a digital mixer which converts the real input data into two separate data streams of I and Q data. An NCO (Numerically Controlled Oscillator) can be programmed with the desired down converter center frequency from 0 to 30MHz with a 1Hz step size. The AD6620 also contains three programmable decimation stages to reduce the sample rate as well as a FIR filter to provide band pass filtering on the decimated data.

The data from either the A/D or down converter chip is placed into a 262144 sample FIFO buffer. Data can then be sent out over the USB interface as continuously streaming data or captured blocks depending on the data rates.

### 4.3. FTDI USB interface

The USB interface of the SDR-14 is provided by an interface chip manufactured by FTDI (<http://www.ftdichip.com/>). The chip implements the low level USB protocol stack in hardware and provides the basic data connection to the SDR-14.

FTDI provides a Windows driver for the FT245BM chip and so the low level USB interface is hidden from the application software writer. The driver provides basic read and write functions that are then used to implement the actual SDR-14 interface protocol. Please refer to the FTDI driver documentation for the function specifications and also example code.

The generic driver provided by FTDI is used but one must modify the ftd2xx.inf file to use a PID number of 0xF728. The driver files included with the SpectraVue (<http://www.moetronix.com/>) program already contain a modified ftd2xx.inf file.

## 5. SDR-14 Control Item Definitions

All examples use hexadecimal notation within brackets [] for the individual byte values.  
The "Target" referred to is the SDR-14 unit.

## 5.1. General Control Items

### 5.1.1. Target Name

Purpose: Returns an ASCII string describing the Target device.

Control Item Code : 0x0001

Control Item Parameter Format: The data is a NULL(zero) terminated character byte string.

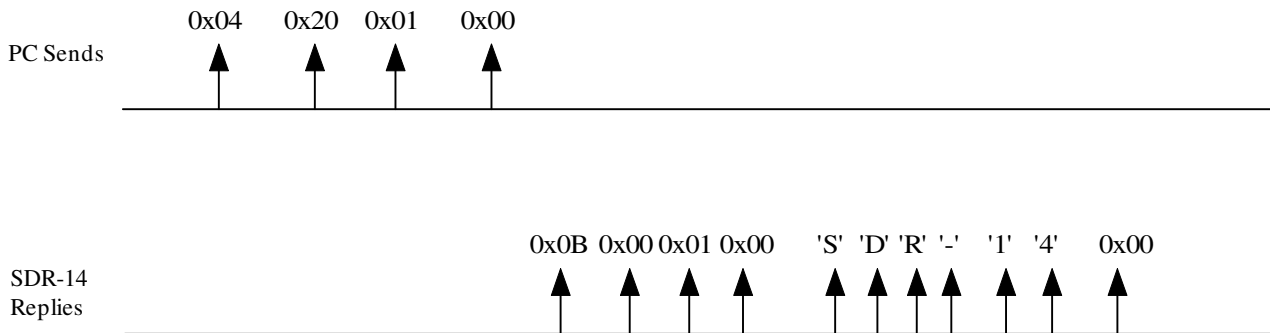
Example, to request the target name, the host sends:

[04][20] [01][00]

The Target responds with "SDR-14" :

[0B][00] [01][00] [53][44][52][2D][31][34][00]

### Example of Obtaining Target Name



### 5.1.2. Target Serial Number

Purpose: Contains an ASCII string containing the Target device serial number.

Control Item Code : 0x0002

Control Item Parameter Format: The data is a NULL(zero) terminated character byte string.

Example, to request the target serial number the host sends:

[04][20] [02][00]

The Target responds with "MT123456" or the serial number the particular device:

[0C][00] [02][00] [4D][54][31][32][33][34][35][36][00]

### 5.1.3. Interface Version

Purpose: Contains the version number of the Host or Targets implemented Interface. This allows the Host or Target to display or adapt to different versions of the interface.

Control Item Code : 0x0003

Control Item Parameter Format: The data is a 2 byte 16 bit unsigned variable equal to the version times 100. For example the value 123 would be version 1.23.

Example, to request the target interface version the host sends:

[04][20] [03][00]

The Target with an interface version of 5.29 responds with:

[06][00] [03][00] [11][02]

#### 5.1.4. Hardware/Firmware Versions

Purpose: Contains the Firmware or Hardware version information of the Target.

Control Item Code: 0x0004

Control Item Parameter Format:

The first parameter is a 1 byte Firmware ID specifying which firmware or hardware version to retrieve.

ID=0 returns the PIC boot code version.

ID=1 returns the PIC firmware version.

The version data is a 2 byte 16 bit unsigned variable equal to the version times 100. For example the value 123 would be version 1.23.

Example, to request the PIC firmware version host sends:

[05][20] [04][00] [01]

The Target with a PIC firmware version of 5.29 responds with:

[07][00] [04][00] [01] [11][02]

Example, to request the PIC boot code version host sends:

[05][20] [04][00] [00]

The Target with a PIC firmware version of 5.29 responds with:

[07][00] [04][00] [00] [11][02]

#### 5.1.5. Status/Error Code

Purpose: Contains the Error/Status code(s) of the Target. This item is used to notify the Host of any error or problem using a list of code values. Once the error code(s) are obtained, the host can interrogate the Target "Error String" Control Item to obtain a description string of the error(s) or status.

Control Item Code: 0x0005

Control Item Parameter Format: The data is a list of 1 byte unsigned variable equal to the error number associated with a particular error. There can be multiple error codes returned by the Target.

0x0B = SDR-14 Idle

0x0C = SDR-14 Busy(capturing data)

0x0D = SDR-14 Loading AD6620 parameters

0x0E = SDR-14 Boot mode Idle

0x0F = SDR-14 Boot mode busy programming

0x80 = SDR-14 Boot mode programming error

Example, host request status:

[04][20] [05][00]

The idle Target responds with

[05][00] [05][00] [0B]

#### 5.1.6. Status String

Purpose: Contains the Status/Error Status text strings describing the current state of the target device.

Control Item Code: 0x0006

Control Item Request Parameter Format: The data is a 1 byte unsigned variable that specifies the error number of the desired error string. This allows the Host to obtain an information text string from the Target device about any of the targets error codes.

Control Item Response Parameter Format: The data is a NULL(zero) terminated character byte string.

Example, a Host wants to obtain the text string associated with the status code (0x0C) returned by the target.

The Host would send:

[05][20] [06][00] [0C]

The Target would respond with a text string "Running"

[0C][00] [06][00] [52][75][6E][6E][69][6E][67][00]

## 5.2. SDR-14 Receiver Control Items

### 5.2.1. Receiver State

Purpose: Controls the operational state of the SDR-14 and specifies the input channel.

Control Item Code: 0x0018

Control Item Parameter Format:

The first parameter is a 1 byte receiver channel ID ( 0x00, 0x01, 0x80, or 0x81).

ID=0 (non-AD6620 real data mode)Direct input to the A/D

ID=1 (non-AD6620 real data mode)Preamp followed by a.1 to 30MHz Filter then the A/D

ID=0x80 (AD6620 complex data mode)Direct input to the A/D

ID=0x81 (AD6620 complex data mode)Preamp followed by a.1 to 30MHz Filter then the A/D

The second parameter is a 1 byte value defined as:

0x01 = Idle(Stop)

0x02 = Run.

The third parameter is a 1 byte parameter specifying the capture mode.

0 = Contiguous mode where the data is contiguously sent back to the Host.

(Only is possible for sample rates < 160 KSPS)

1 = Continuous mode where N blocks of 8192 bytes of data is continually sent back to the Host then the SDR-14 FIFO is reset and the next N blocks are sent.

(Used for sample rates > 160 KSPS)

2 = One Shot mode where N blocks of 8192 bytes of data is sent back to the Host

The fourth parameter is N, the number of blocks requested to be sent before resetting the FIFO.

Range is 1 to 128. Value is ignored for Contiguous mode. Each block is a Data Item of 8192 bytes.

### **!!! NOTE !!!**

A Data Item "ACK" message must be sent by the host every 2-3seconds while the SDR-14 is sending data to prevent a data watchdog timeout in the SDR-14.

A Data Item "ACK" message is a 16 bit header with a Message Type = 011b with a single parameter byte specifying the Data Item (0 in this case). The 16 bit header is a fixed value (0000 0011 0110 0000 = 0x0360 ). The parameter byte following the header specifies which Data Item block that is being ACK'd.

The host would send the following every 2-3seconds:

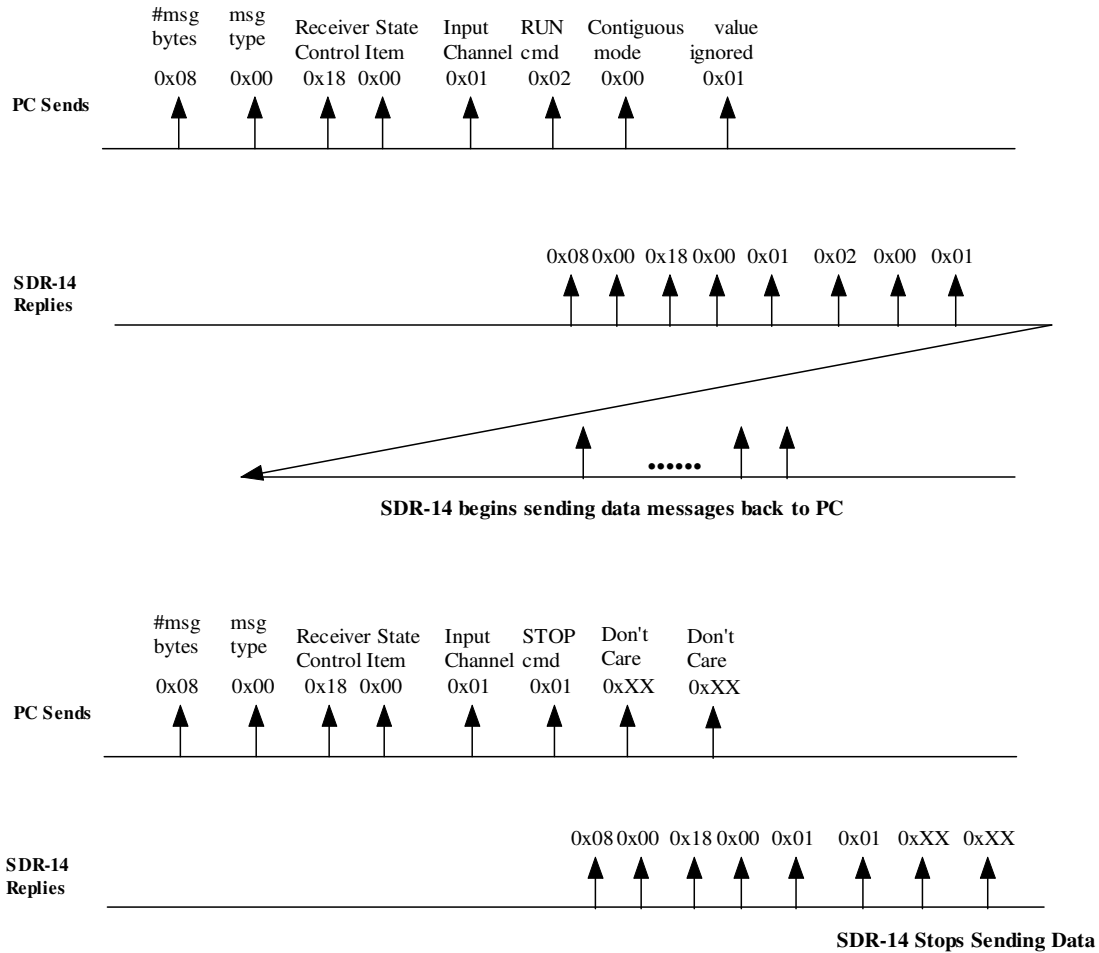
[03][60] [00]

Actually any message sent to the SDR-14 will reset the watchdog but a Data Item Ack is the shortest.

### **!!! NOTE !!!**

Example: Request to start the SDR-14 capturing data in the continuous mode on channel 1.  
 The host sends:  
 [08][00] [18][00] [01] [02] [00] [01]  
 The Target responds with:  
 [08][00] [18][00] [01] [02] [00] [01]

**Example of Starting/Stopping the SDR-14 in Contiguous Mode**



If in the continuous data mode, the SDR-14 sends an unsolicited Receiver State message back to the host after the specified number of data blocks has been sent and the SDR-14 has reset its FIFO. This message can be used to synchronize the incoming data blocks with the FIFO reset occurring. Note that the SDR-14 does not stop sending data after sending the requested number of data blocks but continues with the next set. By not having to wait for the host to restart the data transfer, a much faster capture process is realized.

Example: Request to start the SDR-14 capturing data in the continuous mode and send 9 data blocks on channel 0 before resetting the FIFO.

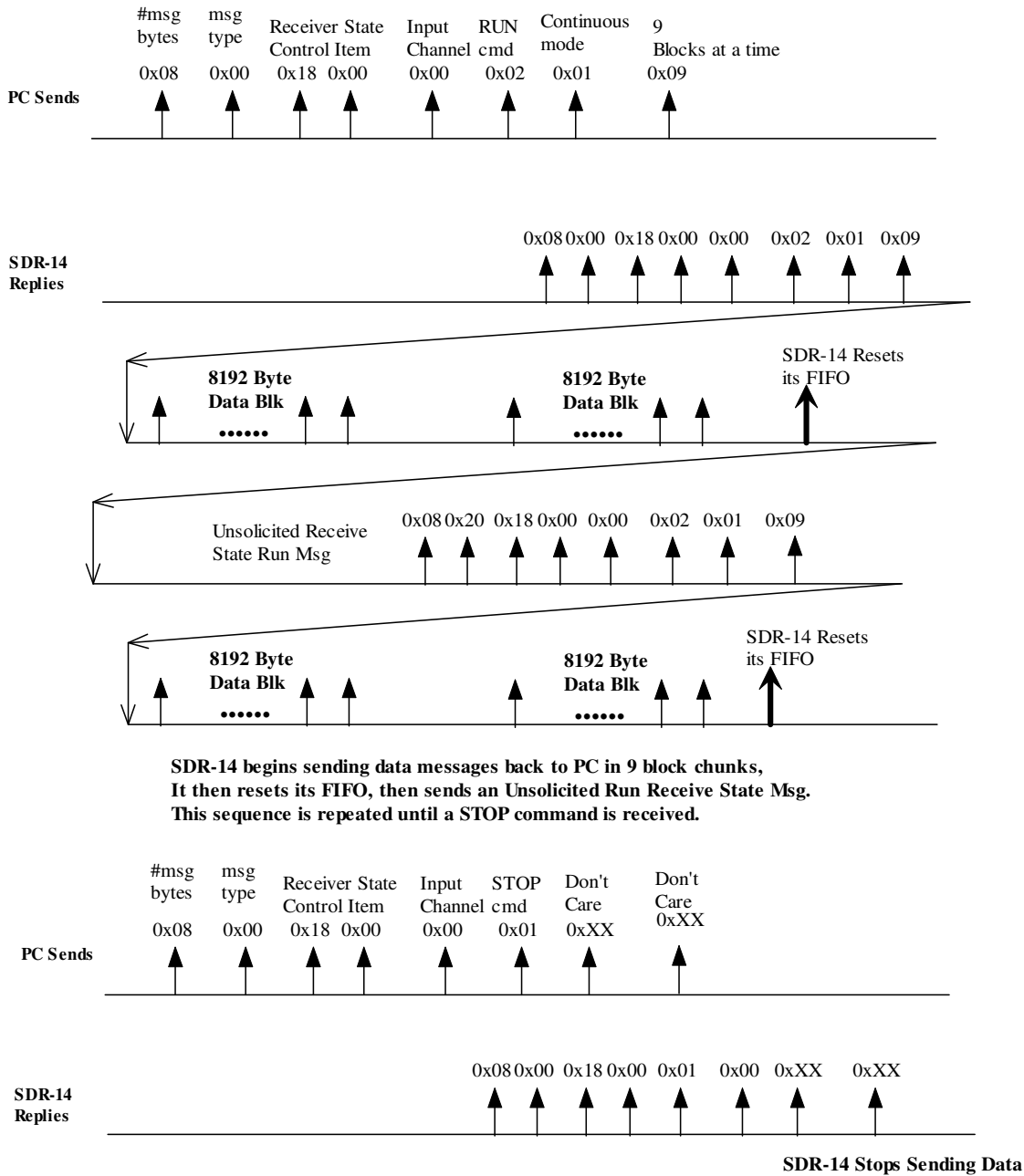
The host sends:

[08][00] [18][00] [00] [02] [01] [09]

The Target responds with:

[08][00] [18][00] [00] [02] [01] [09]

### Example of Starting/Stopping the SDR-14 in Continuous Mode



Example: Request to stop the SDR-14 capturing in continuous mode.

The host sends:

[08][00] [18][00] [00] [01] [00] [00]

The Target responds with:

[08][00] [18][00] [00] [01] [00] [00]

In the One Shot data mode after the specified number of data blocks has been sent, the SDR-14 sends an unsolicited Receiver State ON message and then an unsolicited Receiver State IDLE message back to the host. The SDR-14 then stops and goes back to idle mode. This mode can be used to get a number of blocks once without having to tell the SDR-14 to stop.

Example: Request to start the SDR-14 capturing 4 blocks of data in the one shot mode from channel 0 before resetting the FIFO and going back to idle.

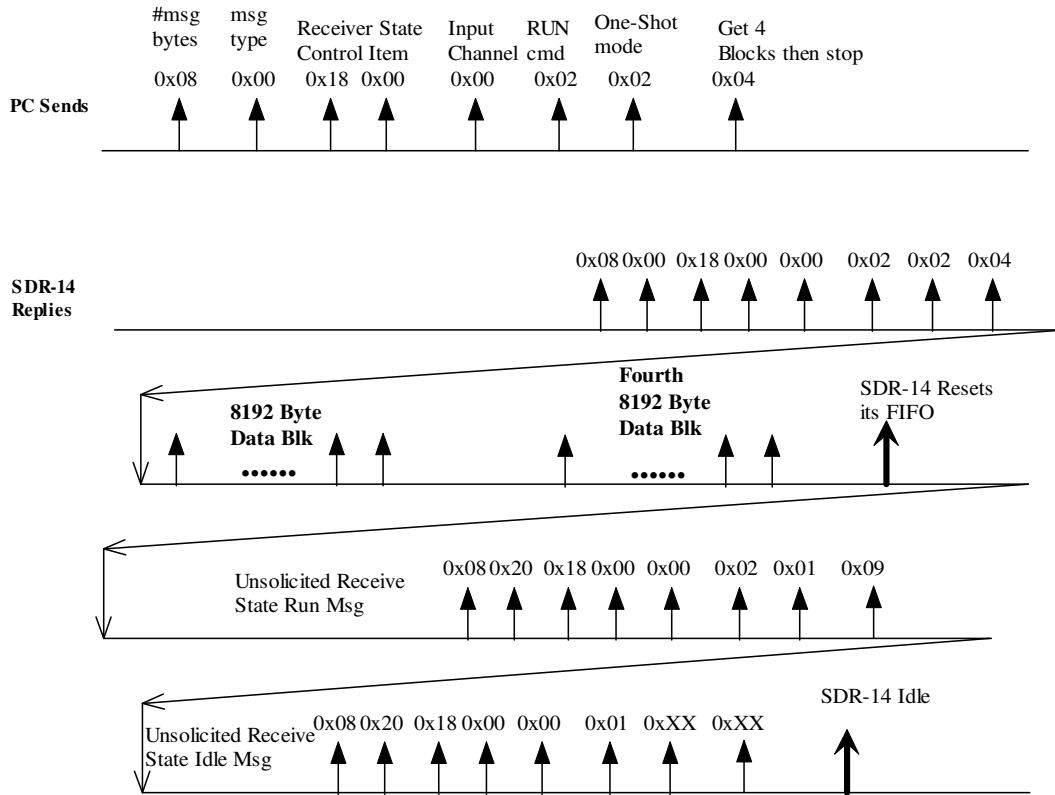
The host sends:

[08][00] [18][00] [00] [02] [02] [04]

The Target responds with:

[08][00] [18][00] [00] [02] [02] [04]

### Example of Starting the SDR-14 in One-Shot Mode



**SDR-14 begins sending data messages back to PC until the 4th block is sent, It then resets its FIFO, then sends an Unsolicited Run Receive State Msg and an Unsolicited Byte Run Receive State Msg then stops and goes idle.**

### 5.2.2.Receiver Frequency

Purpose: Controls the SDR-14 NCO center frequency.

Control Item Code: 0x0020

Control Item Parameter Format:

The first parameter is a 1 byte receiver channel ID ( 0 to 3). This parameter is ignored.

Followed by a 4 byte frequency value in Hz(32 bit unsigned integer LSB first)

The range of this number must be 0 to 33,333,333Hz

Followed by a 1 byte(8 bit unsigned integer ) multiplier value. The multiplier should be 1.

Example, To set The SDR-14 NCO frequency to 14.010 MHz.

The host sends this:

[0A][00] [20][00] [00] [90][C6][D5][00] [01]

The Target responds with:

[0A][00] [20][00] [00] [90][C6][D5][00] [01]

Example, To get the current NCO frequency:

The host sends this:

[05][20] [20][00] [00]

The Target responds with:

[0A][00] [20][00] [00] [90][C6][D5][00] [01]

### 5.2.3. SDR-14 A/D Input Sample Rate

Purpose: Specifies the SDR-14 input sample rate.

Control Item Code: B0

Control Item Parameter Format:

The first parameter is a 1 byte receiver channel ID ( 0 to 3). This parameter is ignored.

The following 4 byte parameter specifies the input source sample rate in Hz.

The SDR-14 nominally has a sample rate of 66,666,667Hz. This command can be used to specify the actual sample rate so that the frequency can be set more accurately.

Note, this does not change the sample rate but is a way to tell the SDR-14 what its actual sample rate is so it can accurately set the AD6620 NCO frequency. This value is saved in the SDR-14's nonvolatile memory so it does not have to be set by the Host on power up.

Example, suppose the actual A/D sample rate is 66,666,123Hz.

To set the SDR-14 sample rate to 66,666,123Hz.

The host sends:

[09][00] [B0][00] [02] [8B][3E][F9][03]

The Target would reply with the following:

[09][00] [B0][00] [02] [8B][3E][F9][03]

### 5.2.4.RF Gain

Purpose: Controls the Level of RF gain( or attenuation) of the receiver.

Control Item Code: 0x0038

Control Item Parameter Format:

The first parameter is a 1 byte receiver channel ID ( not used).

Followed by a 1 byte signed variable( values can be one of the following: 0, -10, -20 or -30 dB).

Example, to set the receiver RF Gain to -20 dB (-20 is 0xEC).

The host sends this:

[06][00] [38][00] [00] [EC]

The Target responds with:

[06][00] [38][00] [00] [EC]

The host sends this to request the current RF Gain setting:

[05][20] [38][00] [00]

The Target responds with:

[06][00] [38][00] [00] [EC]

### 5.3. SDR-14 Data Item Definitions

#### 5.3.1.SDR-14 Output Data Item 0

Purpose: This is the main data item message that is sent back to the host when the SDR-14 is running.

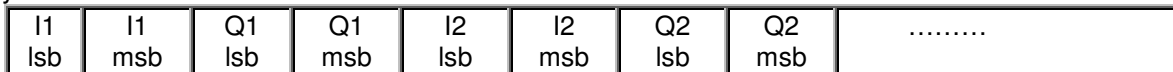
Data 0 Item Parameter Format:

All data blocks sent from the SDR-14 to the host are fixed size of 8194 bytes containing 8192 bytes of data with the 2 byte data item 0 header.



The 8192 data bytes represent either two 16 bit sample values representing the I and Q data or 1 real 16 bit sample value.

The byte breakdown for the I/Q data mode:



The byte breakdown for the real data mode:

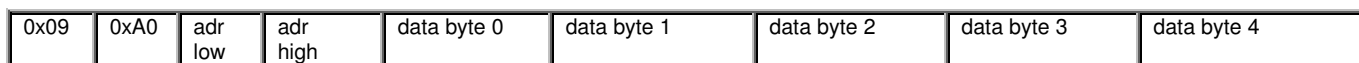


#### 5.3.2.AD6620 Setup Data Item 1

Purpose: This data message is used to download AD6620 setup data to the SDR-14. (See the Analog Devices AD6620 data sheet for the details of the parameters.)

Data 1 Item Parameter Format:

All data blocks sent from the SDR-14 to the host are fixed size of 9 bytes containing a 2 byte AD6620 register address followed by 5 data bytes to load at the specified AD6620 address.



The SDR-14 acks this data message with the 3 byte data ack message.

Example, to set the AD6620 register data at address 0x302 to 0x123456789A.

The host sends this:

[09][A0] [02][03] [9A] [78] [56] [34] [12]

The SDR-14 responds with:

[03][60] [01]

