Radio Sync Receiver

Application Guide

:

Winter 2008

Copyright(c) 2004-2008 Beagle Software. All rights reserved.
Beagle Software provides this publication "as is" without warranty of any kind, either expressed or implied. The use of the software accompanying this documentation is subject to the Software License Agreement contained herein.

MS-DOS, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP and Windows 2003 are trademarks of Microsoft, Inc.

beagle software
www.beaglesoft.com

1. Introduction

The Radio Sync BS2 is an integrated receiver with high sensitivity and a pre-decoding of the time signal transmitted from WWVB, DCF, JJY, MSF and HBG. The receiver is prepared for multi-frequency and country reception by using integrated reception and decoder logic. The Radio Sync BS2 is powered by a micro-controller which provides for a smart and simple instruction interface.

Features

o Automatic reception of long wave time signals worldwide including WWVB
o Manual or automatic selection of radio control signals. Automatic switch between dual band signals. Built in decoding for different bands. Built-in multi-band ferrite core antenna
o Adjustable reception settings (including time and duration of reception). Forced reception mode
o Real time clock (local time and UTC time), 24-hour system. DST and time zone support
o Real-time signal quality value during reception
O Standard RS-232 Interface

2. Overview

2.1 Interface to application

The Radio Sync BS2 uses two wires (RXD, TXD) in the RS-232 interface to communicate.

2.2 Time piece functions

The receiver will respond to commands from the host and return required time information.
- Time information available includes hours, minutes, second, day of week, month, year (ie yy), special information (e.g. DST status)
- The Radio Sync BS2 provides local time (factoring in time zone) and UTC time
- The receiver's real time clock can be set manually, e.g. if no reception is possible

2.3 Time signal reception functions

- Selection of WWVB, DCF, JJY or MSF broadcast signal
- Adjustable reception options:
  - o max. duration of reception
  - o number of receptions
  - o validation of receptions – minimum requirements for consecutive or cumulative time signal receptions
  - o Decoding options, e.g. time only, time + date only
- Three selectable reception modes:
  - o manual

- o continuous
- o pre-set reception sequence

(The duration of reception, the number of auto receptions, interval between auto reception trials (if more than one is defined), start time of 1st auto reception of the day can all be defined by the command interface)

2.4 Available reception signal settings

Countries with long wave time broadcast supported by the Radio Sync receiver include the USA (**WWVB**), UK (**MSF**), JAPAN (**JJY**) and Europe /Germany (**DCF**) bands. A number of standards exist for the transmission of accurate time data using RF signals. Collectively, the standards are known as atomic clock or Radio Controlled Clock (RCC) standards. The RCC standards all share certain signal characteristics, but differ in other ways. One common characteristic is that each standard transmits one data bit each second using pulse width modulation, and the data is transmitted in frames of 60 bits (or a period of 60 seconds). However, the transmission frequency, the width of the pulses and the order in which data is transmitted varies according to the particular standard which is set by each country.

Country	Band Name	Frequency (kHz)
Germany	DCF	77.5
Japan	JJY40	40
Japan	JJY60	60
UK	MSF	60
USA	WWVB	60

In the US the NIST sponsored WWVB is a 60 kHz broadcast from Colorado and Hawaii. Signal coverage is throughout the continental US, and is generally better for locations in proximity to the transmitters.

Japan broadcasts its time signal data at two frequencies: 40 kHz and 60 kHz. However the data carried at each frequency is identical, so in practice it does not matter which frequency is chosen if the chosen frequency is decoded correctly.

Each format differs depending on the data carried by the 60 bits during a time span of 1 minute is allocated and used. The width, or duration, of high and low pulses used to signify data bits, markers within the data, and the start and end of the data frame also varies according to the individual country standards.

In the Radio Sync receiver clock reception can be set by software command from an external host. Various single and multi reception-modes can be used:

Single band :	Dual band:	Multi band
DCF	JJY40 / JJY60	Scan all signals
JJY40	MSF / DCF	
JJY60		
MSF		
WWVB		

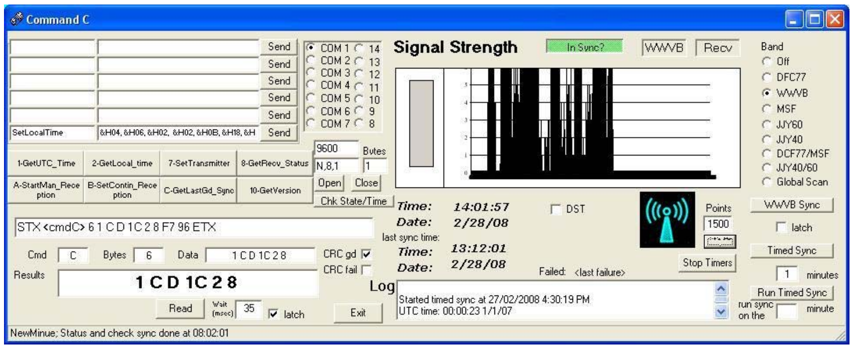
3. Application Development

3.1 Minimal Implementation

The minimal time keeping application includes an initialization, reception and time reading. The initialization sequence is used for setting reception parameters including the transmitter type. Choice of reception mode (manual, continuous, pre-set) instructs the receiver to begin receiving broadcast signals. The typical application includes a time read step and a verification step to check when the receiver has last been synchronization. Further refinements include setting of the local clock and local settings and fine tuning the reception sequence and reception parameters.

3.2 Demo Application

The demo application provides an hands on introduction to functions of the Radio Sync BS2 receiver. The demo application software (including a Windows based installer) is available for download from our homepage. The Demo application includes the most commands and provides a way of sending data to the receiver.



SetLocalTime &H04, &H06, &H02, &H02, &H0B, &H18, &H

[✓ COM 1] COM 2 COM 3 COM 4 COM 5 COM 6 COM 7 14 13 12 11 10 9 8

9600 N,8,1 Bytes 1

STX <cmdC> 6 1 C D 1C 2 8 F7 96 ETX

Cmd C Bytes 6 Data 1 C D 1C 2 8 CRC gd ✓ CRC fail ☐

Results 1 C D 1C 2 8

Wait (msec) 35 ✓ latch

WWVB

Time: 14:01:57
Date: 2/28/08
last sync time:
Time: 13:12:01
Date: 2/28/08

☐ DST

Failed: <last failure>

Points 1500

[✓ WWVB]

☐ latch

1 minutes

run sync on the minute

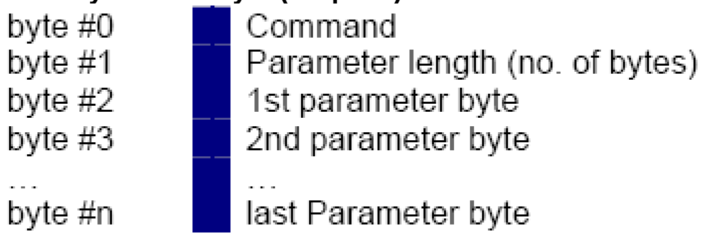
Log
Started timed sync at 27/02/2008 4:30:19 PM
UTC time: 00:00:23 1/1/07

NewMinue; Status and check sync done at 08:02:01

4. Command Protocol

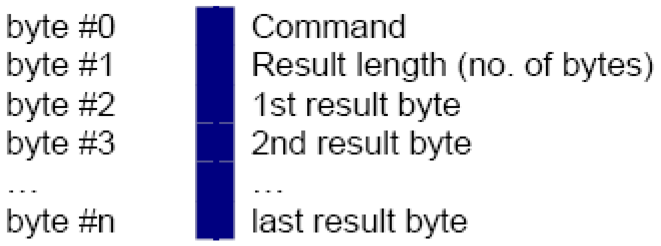
The Radio Sync receiver replies to commands made from an external device. The commands must be formatted specifically for the receiver. The protocol used involves a series of signal, command and data and verification bytes of information.

Radio Sync BS2 Layer (Request):



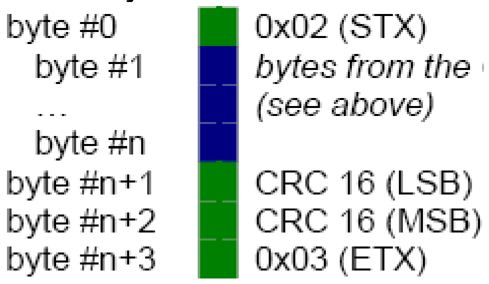
byte #0	🟦	Command
byte #1	🟦	Parameter length (no. of bytes)
byte #2	🟦	1st parameter byte
byte #3	🟦	2nd parameter byte
…	🟦	…
byte #n	🟦	last Parameter byte

Radio Sync BS2 Layer (Response):



byte #0	🟦	Command
byte #1	🟦	Result length (no. of bytes)
byte #2	🟦	1st result byte
byte #3	🟦	2nd result byte
…	🟦	…
byte #n	🟦	last result byte

RS-232 Layer:



byte #0	🟩	0x02 (STX)
byte #1	🟦	*bytes from the*
…	🟦	*(see above)*
byte #n	🟦	
byte #n+1	🟩	CRC 16 (LSB)
byte #n+2	🟩	CRC 16 (MSB)
byte #n+3	🟩	0x03 (ETX)

4.1 RS-232 Interface

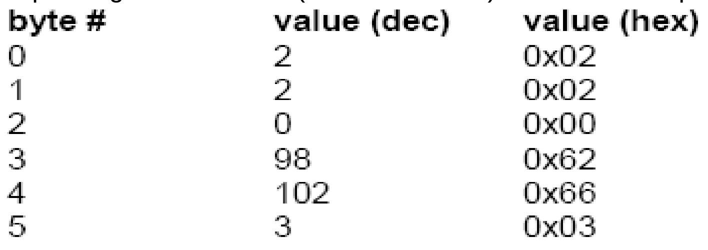
The Radio Sync receiver features a simple 3-wire interface (Transmit, Receive and Ground). It requires no hardware or software handshaking be set to communicate.

Use the following serial settings when communicating to the receiver:

- **9600 baud**
- **No parity bit**
- **Eight data bits**
- **One stop bit**

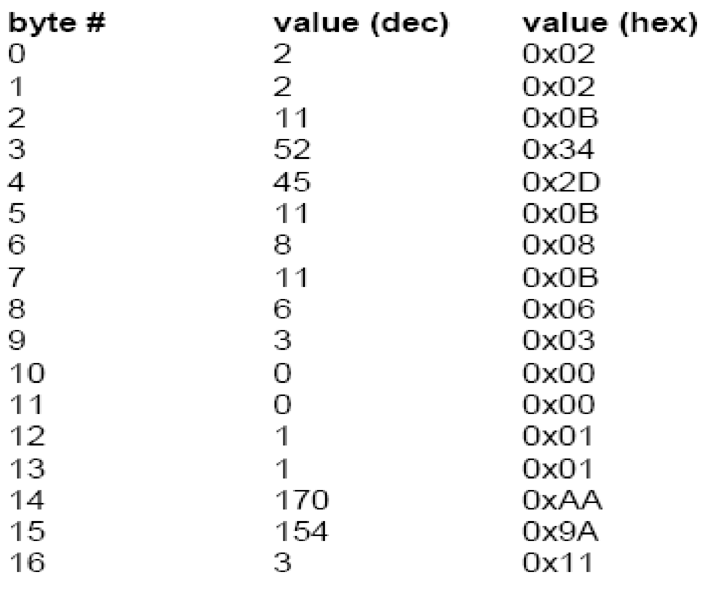
Protocol Example

Requesting the local time (command 0x02) via the RS232 protocol:



byte #	value (dec)	value (hex)
0	2	0x02
1	2	0x02
2	0	0x00
3	98	0x62
4	102	0x66
5	3	0x03

The Radio Sync BS2 response:



byte #	value (dec)	value (hex)
0	2	0x02
1	2	0x02
2	11	0x0B
3	52	0x34
4	45	0x2D
5	11	0x0B
6	8	0x08
7	11	0x0B
8	6	0x06
9	3	0x03
10	0	0x00
11	0	0x00
12	1	0x01
13	1	0x01
14	170	0xAA
15	154	0x9A
16	3	0x11

Interpretation: It is 11:47:49 AM on November 8th 06 (bytes 3->8) (a Wednesday (byte 9)). The clock of the Radio Sync BS2 is not in sync with a time signal transmitter (byte 10) and the DST is not active (byte 11). The time zone is set to UTC+1 (byte 12) and while calculating local time, the DST information has to be considered (byte 13).

4.2 CRC-Calculation

A cyclic redundancy check (**CRC**) is a type of hash function used to produce a checksum to verify incoming data integrity. CRC is used for all strings sent to and from the receiver (exceptions are noted). The CRC used in the Radio Sync BS2 is based on a CRC16 (see http://en.wikipedia.org/wiki/CRC16 for details). It is calculated for all bytes of the Radio Sync BS2 Layer (see *General command structure*).
Note: For testing and evaluating reasons, the Radio Sync BS2 will also accept two 0x00 bytes as CRC. For production environments however, we recommend using valid CRC bytes.

4.3 CRC Implementation

In Windows, the commonly available CRC 16 dynamic link library CRC16.dll, can be used to calculate the CRC directly.
A standard implementation in C:

```
unsigned int CalcCrc16(unsigned char *Buffer,
unsigned char Length)
{
volatile unsigned char i,j;
volatile unsigned int Crc16 = 0;
volatile int c;
char *pBuf;
pBuf = (char*)(Buffer);
for( i=0;i<Length;i++)
{
c = *pBuf++;
c = c << 8;
for( j=0;j<8;j++ )
{
if( (Crc16^c) & 0x8000 )
Crc16 = ( Crc16<<1 ) ^ 0x1021;
else
Crc16 = Crc16 << 1;
c = c << 1;
}
}
return (Crc16);
}
```

Implementation of CRC Calculation using a lookup table in C# (.NET Framework) and C:

```
public class CRCCalc
{
static ushort[] Crc16Table = new ushort[] {
0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50A5, 0x60C6, 0x70E7,
0x8108, 0x9129, 0xA14A, 0xB16B, 0xC18C, 0xD1AD, 0xE1CE, 0xF1EF,
0x1231, 0x0210, 0x3273, 0x2252, 0x52B5, 0x4294, 0x72F7, 0x62D6,
0x9339, 0x8318, 0xB37B, 0xA35A, 0xD3BD, 0xC39C, 0xF3FF, 0xE3DE,
0x2462, 0x3443, 0x0420, 0x1401, 0x64E6, 0x74C7, 0x44A4, 0x5485,
0xA56A, 0xB54B, 0x8528, 0x9509, 0xE5EE, 0xF5CF, 0xC5AC, 0xD58D,
0x3653, 0x2672, 0x1611, 0x0630, 0x76D7, 0x66F6, 0x5695, 0x46B4,
0xB75B, 0xA77A, 0x9719, 0x8738, 0xF7DF, 0xE7FE, 0xD79D, 0xC7BC,
0x48C4, 0x58E5, 0x6886, 0x78A7, 0x0840, 0x1861, 0x2802, 0x3823,
0xC9CC, 0xD9ED, 0xE98E, 0xF9AF, 0x8948, 0x9969, 0xA90A, 0xB92B,
0x5AF5, 0x4AD4, 0x7AB7, 0x6A96, 0x1A71, 0x0A50, 0x3A33, 0x2A12,
0xDBFD, 0xCBDC, 0xFBBF, 0xEB9E, 0x9B79, 0x8B58, 0xBB3B, 0xAB1A,
0x6CA6, 0x7C87, 0x4CE4, 0x5CC5, 0x2C22, 0x3C03, 0x0C60, 0x1C41,
0xEDAE, 0xFD8F, 0xCDEC, 0xDDCD, 0xAD2A, 0xBD0B,
0x8D68, 0x9D49, 0x7E97, 0x6EB6, 0x5ED5, 0x4EF4, 0x3E13, 0x2E32,
0x1E51, 0x0E70, 0xFF9F, 0xEFBE, 0xDFDD, 0xCFFC, 0xBF1B, 0xAF3A,
0x9F59, 0x8F78, 0x9188, 0x81A9, 0xB1CA, 0xA1EB, 0xD10C, 0xC12D,
0xF14E, 0xE16F, 0x1080, 0x00A1, 0x30C2, 0x20E3, 0x5004, 0x4025,
0x7046, 0x6067, 0x83B9, 0x9398, 0xA3FB, 0xB3DA, 0xC33D, 0xD31C,
0xE37F, 0xF35E, 0x02B1, 0x1290, 0x22F3, 0x32D2, 0x4235, 0x5214,
0x6277, 0x7256, 0xB5EA, 0xA5CB, 0x95A8, 0x8589, 0xF56E, 0xE54F,
0xD52C, 0xC50D, 0x34E2, 0x24C3, 0x14A0, 0x0481, 0x7466, 0x6447,
0x5424, 0x4405, 0xA7DB, 0xB7FA, 0x8799, 0x97B8, 0xE75F, 0xF77E,
0xC71D, 0xD73C, 0x26D3, 0x36F2, 0x0691, 0x16B0, 0x6657, 0x7676,
0x4615, 0x5634, 0xD94C, 0xC96D, 0xF90E, 0xE92F, 0x99C8, 0x89E9,
0xB98A, 0xA9AB, 0x5844, 0x4865, 0x7806, 0x6827, 0x18C0, 0x08E1,
0x3882, 0x28A3, 0xCB7D, 0xDB5C, 0xEB3F, 0xFB1E, 0x8BF9, 0x9BD8,
0xABBB, 0xBB9A, 0x4A75, 0x5A54, 0x6A37, 0x7A16, 0x0AF1, 0x1AD0,
0x2AB3, 0x3A92, 0xFD2E, 0xED0F, 0xDD6C, 0xCD4D, 0xBDAA, 0xAD8B,
0x9DE8, 0x8DC9, 0x7C26, 0x6C07, 0x5C64, 0x4C45, 0x3CA2, 0x2C83,
0x1CE0, 0x0CC1, 0xEF1F, 0xFF3E, 0xCF5D, 0xDF7C, 0xAF9B, 0xBFBA,
0x8FD9, 0x9FF8, 0x6E17, 0x7E36, 0x4E55, 0x5E74, 0x2E93, 0x3EB2,
0x0ED1, 0x1EF0,
};
public static ushort CalcCRC(byte[] buffer)
{
ushort crc = 0;
ushort bufferPointer = 0;
for (int i = 0; i < buffer.Length; i++)
{
ushort crcRightShift = (ushort)(crc >> 8);
ushort crcLeftShift = (ushort)(crc << 8);
ushort byteFromBuffer =
buffer[bufferPointer++];
crc =
(ushort)(Crc16Table[(ushort)(((ushort)(crcRig
htShift ^ byteFromBuffer)) & 0xFF)] ^
crcLeftShift);
}
return crc;
}
}
const unsigned int code Crc16Table[256] =
{
0X0000, 0X1021, 0X2042, 0X3063, 0X4084, 0X50A5, 0X60C6, 0X70E7,
0X8108, 0X9129, 0XA14A, 0XB16B, 0XC18C, 0XD1AD, 0XE1CE, 0XF1EF,
0X1231, 0X0210, 0X3273, 0X2252, 0X52B5, 0X4294, 0X72F7, 0X62D6,
0X9339, 0X8318, 0XB37B, 0XA35A, 0XD3BD, 0XC39C, 0XF3FF, 0XE3DE,
0X2462, 0X3443, 0X0420, 0X1401, 0X64E6, 0X74C7, 0X44A4, 0X5485,
0XA56A, 0XB54B, 0X8528, 0X9509, 0XE5EE, 0XF5CF, 0XC5AC, 0XD58D,
0X3653, 0X2672, 0X1611, 0X0630, 0X76D7, 0X66F6, 0X5695, 0X46B4,
0XB75B, 0XA77A, 0X9719, 0X8738, 0XF7DF, 0XE7FE, 0XD79D, 0XC7BC,
0X48C4, 0X58E5, 0X6886, 0X78A7, 0X0840, 0X1861, 0X2802, 0X3823,
0XC9CC, 0XD9ED, 0XE98E, 0XF9AF, 0X8948, 0X9969, 0XA90A, 0XB92B,
0X5AF5, 0X4AD4, 0X7AB7, 0X6A96, 0X1A71, 0X0A50, 0X3A33, 0X2A12,
0XDBFD, 0XCBDC, 0XFBBF, 0XEB9E, 0X9B79, 0X8B58, 0XBB3B, 0XAB1A,
0X6CA6, 0X7C87, 0X4CE4, 0X5CC5, 0X2C22, 0X3C03, 0X0C60, 0X1C41,
0XEDAE, 0XFD8F, 0XCDEC, 0XDDCD, 0XAD2A, 0XBD0B, 0X8D68, 0X9D49,
0X7E97, 0X6EB6, 0X5ED5, 0X4EF4, 0X3E13, 0X2E32, 0X1E51, 0X0E70,
0XFF9F, 0XEFBE, 0XDFDD, 0XCFFC, 0XBF1B, 0XAF3A, 0X9F59, 0X8F78,
0X9188, 0X81A9, 0XB1CA, 0XA1EB, 0XD10C, 0XC12D, 0XF14E, 0XE16F,
0X1080, 0X00A1, 0X30C2, 0X20E3, 0X5004, 0X4025, 0X7046, 0X6067,
0X83B9, 0X9398, 0XA3FB, 0XB3DA, 0XC33D, 0XD31C, 0XE37F, 0XF35E,
0X02B1, 0X1290, 0X22F3, 0X32D2, 0X4235, 0X5214, 0X6277, 0X7256,
0XB5EA, 0XA5CB, 0X95A8, 0X8589, 0XF56E, 0XE54F, 0XD52C, 0XC50D,
0X34E2, 0X24C3, 0X14A0, 0X0481, 0X7466, 0X6447, 0X5424, 0X4405,
0XA7DB, 0XB7FA, 0X8799, 0X97B8, 0XE75F, 0XF77E, 0XC71D, 0XD73C,
0X26D3, 0X36F2, 0X0691, 0X16B0, 0X6657, 0X7676, 0X4615, 0X5634,
0XD94C, 0XC96D, 0XF90E, 0XE92F, 0X99C8, 0X89E9, 0XB98A, 0XA9AB,
0X5844, 0X4865, 0X7806, 0X6827, 0X18C0, 0X08E1, 0X3882, 0X28A3,
0XCB7D, 0XDB5C, 0XEB3F, 0XFB1E, 0X8BF9, 0X9BD8, 0XABBB, 0XBB9A,
0X4A75, 0X5A54, 0X6A37, 0X7A16, 0X0AF1, 0X1AD0, 0X2AB3, 0X3A92,
0XFD2E, 0XED0F, 0XDD6C, 0XCD4D, 0XBDAA, 0XAD8B, 0X9DE8, 0X8DC9,
0X7C26, 0X6C07, 0X5C64, 0X4C45, 0X3CA2, 0X2C83, 0X1CE0, 0X0CC1,
0XEF1F, 0XFF3E, 0XCF5D, 0XDF7C, 0XAF9B, 0XBFBA, 0X8FD9, 0X9FF8,
0X6E17, 0X7E36, 0X4E55, 0X5E74, 0X2E93, 0X3EB2, 0X0ED1, 0X1EF0,
};
unsigned int CalcCrc16(char *Buffer, unsigned char Length)


{
unsigned char i;
unsigned int Crc16 = 0;
for( i=0;i<Length;i++)
{
Crc16 =
Crc16Table[((Crc16>>8)^*Buffer++)&0xFF]^(Crc16<<8);
}
return (Crc16);
}
```

5.0 Radio Sync BS2 – List of commands

Num-ber	Command	Description
0x01	GetUTCTime	Reads the UTC time from the re-ceiver
0x02	GetLocalTime	Reads the local time from the re-ceiver.
0x03	SetUTCTime	Sets the UTC internal clock of the receiver
0X04	SetLocalTime	Sets the local time of the internal clock of the receiver
0x05	SetLocalTimeOptions	Sets the options to enable the Re-ceiver to calculate the local time from the UTC time.
0x06	SetDST	Enables / disables the one hour DST offset
0x07	SetTransmitter	Sets the transmitter / mode of the receiver
0x08	GetReceiverStatus	Gets the current receive status of the receiver.
0x09	SetReceptionOptions	Sets the reception options of the receiver
0x0A	StartReception	Starts a manual reception of re-ceiver.
0X0B	StartContinuousRecep-tion	Starts a continuous reception
0x0C	GetLastSuccessfulSync	Returns the time (UTC) and date of the last successful reception.
0X0D	SetSyncTimespan	Defines the time span between re-ceptions.
0x0E	SetSecondPolling	Toggles second polling.
0x0F	GetRAWData	Raw data dump
0X10	GetVersion	Returns the firmware version of the receiver
0X11	UseReferenceTime	End reception when a time-code is received that matches the internal time of the receiver

5.1 **Command Detail**

GetUTCTime (0x01)

Parameter(s) –

Result

```
byte second 00-59
byte minute 00-59
byte hour 00-23
byte day 01-31
byte month 01-12
byte year 00-99
byte day of week
     1 … Monday
     2 … Tuesday
     […]
     6 … Saturday
     7 … Sunday
byte Sync State:
     0 not synchronized
     1 synchronized
```

Description Reads the UTC time from the Radio Sync BS2.

GetLocalTime (0x02)

Parameter(s) –

Result

```
byte second 00-59
byte minute 00-59
byte hour 00-23
byte day 01-31
byte month 01-12
byte year 00-99
byte day of week
     1 … Monday
     2 … Tuesday
     […]
     6 … Saturday
     7 … Sunday
byte sync state 0 … not synchronized  1 …
synchronized
byte DST-Flag 0 … Standard time 1 … DST
byte timeZone See SetLocalTimeOptions
byte DSTSupport See SetLocalTimeOptions
```

Description Reads the UTC time from the Radio Sync BS2.

`SetUTCTime` `(0x03)`

Parameter(s) `byte second 00-59`
`byte minute 00-59`
`byte hour 00-23`
`byte day 01-31`
`byte month 01-12`
`byte year 00-99`

Result `byte commandResult see below`
Description Sets the internal clock of the Radio Sync BS2.
Side Effects:

- After the time is set manually, the `GetLastSuccessfulSync` command will always return `000000` and the sync state of the `GetUTCTime / GetLocalTime` commands will always be `0`.

`SetLocalTime` `(0x04)`

Parameter(s) `byte second 00-59`
`byte minute 00-59`
`byte hour 00-23`
`byte day 01-31`
`byte month 01-12`
`byte year 00-99`

Result `byte commandResult, see below`
Description Sets the internal clock of the Radio Sync BS2. If the time/date information was set successfully, the Radio Sync BS2 returns `1`. If the Radio Sync BS2 is not able to set the time-information (e.g. `hour >23`), it returns `0`.
Side Effects:

- After the time is set manually, the `GetLastSuccessfulSync` command will always return `000000` and the sync state of the `GetUTCTime / GetLocalTime` commands will always be `0`.

`SetLocalTimeOptions` `(0x05)`

Parameter(s)

```
byte timeZone
      0 … +- 0 hr
      1 … +1 hr
      […]
      14 … +14hr
      255 … -1 hr
      […]
      244 … -12
byte DSTSupport 0 … false  1 … true
```

Result `byte commandResult, see below`

Description Sets the options to enable the Radio Sync BS2 to calculate the local time from the UTC time.

• `timeZone`: defines the deviation of your local timezone from the UTC time.

• `hasDST`: if `false`, the one hour DST offset is not taken into account when calculating the local time, even if the transmitter sends the DST flag.

`SetDST` `(0x06)`

Parameter(s) `byte dst 0 … DST off   1 … DST on`

Result

`byte commandResult, see below`

Description Enables / disables the one hour DST offset. At the next successful reception, the manual set DST flag will be overwritten by the transmitted value.

`SetTransmitter` `(0x07)`

Parameter(s) `byte transmitterCode`

```
1 … DCF77
2 … WWVB
3 … MSF
4 … JJY60
5 … JJY40
6 … DCF77 / MSF
7 … JJY 40/60
8 … Global Scan
```

Result `byte commandResult, see below`

Description Sets the transmitter / mode of the Radio Sync BS2

Side Effects:

• The timezone information is set according to the location of the chosen transmitter. If you are using one of the scan modes, you should set the time zone manually by using `SetLocalTimeOptions` to avoid getting wrong local time calculations.

• A manual reception is started

`GetReceiverStatus` (0x08)

Parameter(s) –

Result

`byte transmitterCode` See `SetTransmitter`
`byte isReceiving 1 … true 0 … false`
`byte bsi 0-3`

Description • Gets the current receive status of the Radio Sync BS2.

• `transmitterCode`: code of the transmitter, the Radio Sync BS2 is using (has used the last time)
• `isReceiving`: indicates, whether the Radio Sync BS2 is receiving or not
• `bsi`: value of the BS-Indicator of the Radio Sync BS2. Always 0 if BS2 is not in receiving mode.

`SetReceptionOptions` (0x09)

Parameter(s) `byte maxDuration`
`byte numberOfValidReceptions`
`byte consecutiveReceptions 0 … false 1 … true`
`byte autoRecEnabled 0 … false 1 … true`
`byte autoRecHour 0-23`
`byte autoRecMinute 0-59`
`byte autoRecDelta 0-255`
`byte autoRecCount 0-10`
`byte checkParity 0 … true 1 … false`
`byte protocolMask`
`0 … FullDecode`
`2 … TimeOnly`
`1 … TimeDateOnly`

Result

`byte commandResult, see below`

Description Sets the reception options of the Radio Sync BS2.

• `maxDuration`: duration of a reception try
• `numberOfValidReceptions`: defines how many decodings must fit logical together, to allow an update of the internal clock.
• `consecutiveReceptions`: if `true`, the decodings defined by `numberOfValidReceptions` must be received in a consecutive order.
• autoRecEnabled: definies wether the automatic reception is enabled or not.
• `autoRecHour` / `autoRecMinute`: defines the starting point of an automatic reception
• `autoRecDelta`: timespan (in minutes) between two autoreception tries, if the prior reception has failed.
• `autoRecCount`: number of autoreception tries. If the last try was also unsuccessful, the next try will start at the next the starting point defined by `autoRecHour` / `autoRecMinute`
• `checkParity`: if false, the parity-bits sent by the transmitters are not taken into account.
• `protocolMask`: defines which part of the transmission is to be decoded.

`StartReception` `(0x0A)`

Parameter(s) –

Result
`byte commandResult see below`

Description Starts a manual reception.

`StartContinuousReception` `(0x0B)`

Parameter(s) –

Result
`byte commandResult, see below`

Description Starts a continuous reception. Immediately after an invalid or valid reception, the next reception is started again.

`GetLastSuccessfulSync` `(0x0C)`

Parameter(s) –

Result

```
byte second 00-59
byte minute 00-59
byte hour 00-23
byte day 01-31
byte month 01-12
byte year 00-99
```

Description Returns the time (UTC) and date of the last successful reception. This information will be lost, if the receiver looses power. In this case it will return six bytes valued 0.

`SetSyncTimespan` `(0x0D)`

Parameter(s) `byte hour 0-255`

Result
`byte commandResult, see below`

Description Defines the time span between the last successful reception and the moment after that the `sync state` of the `GetLocalTime` / `GetUTCTime` result will switch back to `0`

`SetSecondPolling` `(0x0E)`

Parameter(s) `byte polling 0 … off 1 … on`

Result `byte commandResult, see below`

Description Enables / disables the second polling. If enabled, the Radio Sync BS2 will send a single byte exactly at the change between two seconds. The sent bytes represents the new second. If the Radio Sync BS2 is in polling mode, no other commands than `SetSecondPolling` will be accepted.

`GetRAWData` `(0x0F)`

Parameter(s) –

Result

`10 x byte[8] of rawData`

Description This commands returns 10 packages of an eight-byte-array (64 bits).

The Protocol Identifier Bits added by the BS2 are stored in the first 4 bits. The remaining 60 bits represent the signal received by the BS2. Altogether the raw data of the 10 last receptions are transmitted. The latest received raw data is sent first. There are no CRC bytes sent at the end of the packet.

All raw data information will be erased, if

• the Radio Sync BS2 is disconnected from the power supply

• a new reception is started

`GetVersion` `(0x10)`

Parameter(s) –

Result

`char[] version`

Description Returns the firmware version of the Radio Sync BS2.

`UseReferenceTime` `(0x11)`

Parameter(s)

`byte useReferenceTime 0 … false 1 … true`

Result `byte commandResult, see below`

Description If the reference time is enabled, a reception will end, if a time-code is received, that fits to the internal time of the Radio Sync BS2.

Command Result codes:

Each command that sets an option of the Radio Sync BS2 returns a single data byte. If the Radio Sync BS2 has accepted the command, it will return 255 (or 0xFF). In case of an error, the result byte gives you some information about the cause of the error:

RD_START 0x00
RD_BOARDADDRESS 0x01
RD_REQUESTOR_ADDRESS 0x02
RD_RX_CMD 0x03
RD_DATA_LENGTH 0x04
RD_DATA 0x05
RD_CRC 0x06
RD_ETX 0x07
CRC_CHECK 0x08
RD_END 0x09
RD_ERROR 0x0A
SET_SECOND_ERROR 0x20
SET_MINUTE_ERROR 0x21
SET_HOUR_ERROR 0x22
SET_DATE_ERROR 0x23
SET_MONTH_ERROR 0x24
SET_YEAR_ERROR 0x25
SET_LOCAL_TIME_OPTION_ERROR 0x30
SET_HAS_DST_OPTION_ERROR 0x31
SET_DST_ERROR 0x40
TRANSMITTER_CODE_ERROR 0x50
MAX_DURATION_ERROR 0x60
NMB_VALID_RECEPTIONS_ERROR 0x61
CONSECUTIVE_RECEPTIONS_ERROR 0x62
AUTO_REC_ON_OFF_ERROR 0x63
AUTO_REC_HOUR_ERROR 0x64
AUTO_REC_MINUTE_ERROR 0x65
AUTO_REC_DELTA_ERROR 0x66
AUTO_REC_COUNT_ERROR 0x67
CHECK_PARITY__ERROR 0x68
PROTOCOLL_MASK_ERROR 0x69
START_RECEPTION_ERROR 0x70
SET_SECOND_POLLING_ERROR 0x80
REFERENCE_TIME_ERROR 0x90
NO_ERROR 0xFF

6. Technical Support:

If you are having problems with the Adapter that this document does not address, please feel free to contact us at:

E-mail:	support@beaglesoft.com
Phone:	612-370-1091
Fax:	612-605-7138
Mail:	Beagle Software
	126 N 3rd St, Ste 407
	Minneapolis, MN 55401 USA