**Controlling the Wireless EEG Recorder / DBS Device**

**1 – Overview of System Changes**

Transmission of commands from the computer to the wireless EEG recorder is simple, and works by 1-byte UART commands. This occurs at 115.2Kbaud, which is the same rate at which EEG is coming in. Until now I’ve been using terminal viewing freeware RealTerm to transmit commands to the EEG recorder and to check the content of the packets.

This time around there are 14 bytes coming in per packet rather than 12. This consists of the existing EEG samples from each channel (making up 12 bytes), and the frequency/pulse-width that DBS is currently set to (2 bytes). The three previously unused status bits immediately after each 16-bit sample now have meaning – they correspond to the DBS active state (ON/OFF), and the system identifier (I can program the systems with different transmit frequencies for simultaneous operation).

The DBS code has undergone a complete redesign to accommodate a large range of pulse-widths and frequencies. The only way I managed this is through two instructions per frequency and two instructions per pulse width – although the user doesn’t necessarily have to transmit both instructions to get the DBS parameters they’re after. Check out the section I’ve written on DBS control for details.

**2 – DBS Control**

**2.1 – How the System Works**

The new DBS parameters are versatile and allow the user to select a pulse-width ranging from 10uS – 500mS, and a frequency ranging from 0.1Hz – 5KHz. There are two commands for the frequency and two for pulse-width, which changes the frequency/pulse-width value Fv, Pv, and the frequency/pulse-width multiplier Fm, Pm. For the frequency/pulse-width the value Fv, Pv is a number which ranges from 1-50, and the multipliers Fm, Pm are x0.1, x1, x10 and x100. This gives a total of 200 different frequency values, and 200 pulse-width values as summarised in table 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Multiplier, Pm / Fm** | | | |
|  | X 0.1 | X 1 | X 10 | X 100 |
| Frequency  (Base = 1Hz) | 0.1Hz – 5Hz  (0.1 Hz steps) | 1Hz – 50Hz  (1Hz steps) | 10Hz – 500Hz  (10Hz steps) | 100Hz – 5000Hz (100Hz steps) |
| Pulse-Width  (Base = 100uS) | 10uS – 500uS  (10uS steps) | 100uS – 5mS  (100uS steps) | 1mS – 50mS  (1mS steps) | 10mS – 500mS  (10mS steps) |

Using the multiplier and the value, the following calculation gives the desired frequency and pulse-width:

Frequency = Fv \* Fm \* Fb

Pulse Width = Pv \* Pm \* Pb

Where Fv = Frequency value (set by a command and ranges from 1 to 50)

Fm = Frequency multiplier (set by a command, 4 choices as seen in table 1)

Fb = Frequency base, which is 1Hz

Pv = Pulse-width value (set by a command and ranges from 1 to 50)

Pm = Pulse-width multiplier (set by a command, 4 choices as seen in table 1)

Pb = Pulse-width base, which is 100uS

Fv, Fm, Pm and Pb are the one-byte commands to change the values and multipliers. Changing the multiplier effectively changes the range of the pulse-width / frequency. One does not have to input both commands to change a particular value. If for example the frequency multiplier is already set to x10 and the user wishes to input a frequency of 200Hz, then they would simply set Fv = 20. Alternatively they could change the range Fm to x100, and set Fv to 2, which would have the same effect with f=200Hz. If the frequency value is already set to 200Hz (with Fv = 2, Fm = x100) and they wish to change it to 2Hz, then they would only have to change the range Fm to x1.

Some more examples:

1. Fm = x0.1 and Fv = 12 gives 1.2Hz
2. Pm = x10 and Pv = 25 gives 25mS
3. Pm = x0.1 and Pv = 40 gives 400uS
4. Fm = x1 and Fv = 45 gives 45Hz
5. Fm = x10 and Fv = 5 gives 50Hz
6. Fm = x1 and Fv = 50 gives 50Hz

The system is pre-loaded with 130Hz, 200uS DBS settings as default (default value can be changed to whatever’s convenient).

**2.2 – What happens if they input an impossible value?**

If the user decided to input a pulse-width which is higher than the oscillation period, e.g. 500mS for a 100Hz frequency, they’d end up with a 500mS pulse train separated by a brief delay. This brief delay is in the order of microseconds to tens of microseconds depending on exactly which values were entered – and is due to the way the DBS timing algorithm works. Needless to say if they decide to input an impossible value like this they’d end up with a periodic pulse (with frequency 1/(PW + delay) )with over 99% duty cycle. Perhaps this scenario would be ideal for creating a lesion at the electrode sites by command, but it’s probably safer if WinEDR restricts values above a certain duty cycle. I’ll try implementing these checks into the system if I manage to free up memory.

In any case the EEG recorder transmits back the frequency and pulse-width that DBS is currently configured to. Resetting the system would revert back to the default pre-loaded frequency/pulse-width values, which again can be reprogrammed to whatever’s convenient.

**2.3 – Tolerance**

The values and multipliers that are input from the computer into the receiver undergo a series of calculations, and these determine the appropriate settings for the timing counters on which the DBS algorithm works. The calculations are performed with 16-bit variables and the accuracy is pretty good. Attempting to set the frequency to 130Hz results in a theoretical frequency of 129.99Hz (based on the timing calculations and crystal oscillator frequency), and at this frequency I can’t notice a deviation from 130Hz on the oscilloscope. Pulse-widths are also spot on, though the effects of pulse rise/fall time are more readily seen in the uS range. Due to a lower limit at which the timing loop can operate the lowest value I can use for pulse-width is 10uS. Here it can be seen that the fall-time is significantly higher than the rise-time. Just something to keep in mind.

There is one discrepancy which I’ve noticed… As soon as the duty cycle of pulses exceed ~20%, the frequency jumps up by ~5% e.g. 100Hz becomes 105Hz. This isn’t a gradual effect, since the frequency and pulse-width is spot on for ~19.5% duty cycle. I roughly know what causes this, and will correct that in due course. Although duty cycles of 20% are quite large with regards to common DBS parameters anyway, especially for a current of 100uA.

**2.4 – Discussion of new DBS Control**

This system is designed to be flexible and allow for a large range of frequency and pulse-widths to be used, without having to reprogram the system. There are a number of restrictions faced when trying to devise the ideal DBS control scheme, and by far the biggest one is program memory on the microcontroller (both at the EEG stage and receiver). As such both 16-bit UART transmission and the use of 32-bit variables are not possible. The current 16-bit variables places restrictions on some of the calculations that determine the accurate timing, and so in favour of precision the range of frequency values has been cut short. If 16-bit UART transmission was used, it’d alleviate many of the problems and allow one 16-bit frequency or pulse-width value to be transmitted. However we can only do with one-byte UART instructions due to memory, and one byte alone has not enough information to allow for the large range of frequency/pulse-width values. That’s where the separate multiplier byte comes in.

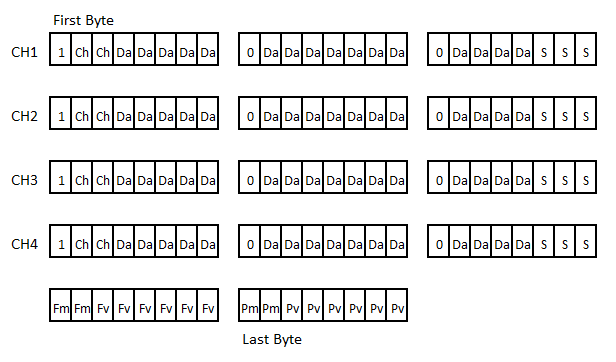
**3.0 – Information Sent to the Computer**

**3.1 – Overview**

Information transmitted from the receiver to the computer via serial port is exactly the same as before except:

* EEG Data is now sampled at 500Hz instead of 600Hz
* Two new bytes appended to the end of the packet that shows the frequency and pulse-width the wireless system is currently programmed to
* The three (previously unused) status bits at the end of each 3-byte sequence now correspond to a) DBS active state and b) the devices frequency channel – since two or more devices can be active simultaneously.

The new structure is seen in figure 1.



**Figure 1 –** Format of packets transmitted through the serial port consists of 12 EEG bytes and two DBS status bytes.

Looking at figure 1, the following bits are:

Ch Channel identifier. For each channel this will be either 00, 01, 10 or 11 indicating which EEG channel the sample is recorded from.

Da Recorded 16-bit EEG sample, with the most significant bit first.

S The remaining 3 bits for each channel correspond to a) the frequency channel of the device (useful for when 2 devices are in use simultaneously, and b) whether DBS is switched on or off (see table. 2)

Fm Frequency multiplier bits

Fv Frequency value

Pm Pulse-width multiplier bits

Pv Pulse-width value

As before each channel contains a single 16-bit sample of EEG. Each channel consists of a 3-byte sequence with the first bit of each byte given 1 or 0 to show whether or not it’s the first byte in the sequence. The two new bytes contain the frequency and pulse-width multipliers, as well as its 1-50 value.

**3.2 – The S-Bits**

The remaining three ‘S’ bits for each 3-byte sequence are summarised in table 2:

|  |  |
| --- | --- |
| **S Value** | **Meaning** |
| 000 | Frequency Ch. 1 – DBS OFF |
| 001 | Frequency Ch. 1 – DBS ON |
| 010 | Frequency Ch. 2 – DBS OFF |
| 011 | Frequency Ch. 2 – DBS ON |
| 100 | Unused |
| 101 | Unused |
| 110 | Unused |
| 111 | Unused |

**Table 2 –** Meaning of the ‘S’ bytes

In table 2 I’ve got 2 variables (frequency channel, DBS active state) multiplexed onto 3 bits, giving 8 possible combinations. Of course this could also be achieved by setting two bits either 1 or 0. But doing it this way allows me to add further frequency channels should one wish to have 4 devices operating simultaneously.

**3.3 – Frequency and Pulse-Width Status Bytes**

The end two bytes correspond to the frequency/PW value and frequency/PW multipliers the wireless system is currently programmed to. This is summarised in tables 3, 4 and 5.

|  |  |
| --- | --- |
| **Fm Value** | **Description** |
| 00 | Frequency multiplier is x100 |
| 01 | Frequency multiplier is x10 |
| 10 | Frequency multiplier is x1 |
| 11 | Frequency multiplier is x0.1 |

**Table 3 –** Frequency Multiplier Values

|  |  |
| --- | --- |
| **Pm Value** | **Description** |
| 00 | Pulse-Width multiplier is x0.1 |
| 01 | Pulse-Width multiplier is x1 |
| 10 | Pulse-Width multiplier is x10 |
| 11 | Pulse-Width multiplier is x100 |

**Table 4 –** Pulse-Width Multiplier Values

Note that these multipliers go in reverse order to each other. Fm = 11 gives x0.1, whereas Pm = 11 gives x100. This is due to the way in which these bits are used for calculation of frequency and pulse-width range.

As for the frequency and pulse-width values, these are both the same and are summarised in table 5:

|  |  |
| --- | --- |
| **Fv or Pv Value** | **Description** |
| 000001 | Fv / Pv is set to 1 |
| 000010 | Fv / Pv is set to 2 |
| 000011 | Fv / Pv is set to 3 |
| …… | …… |
| 110001 | Fv / Pv is set to 49 |
| 110010 | Fv / Pv is set to 50 |

**Table 5 –** Fv/Pv Values

It can be noted that with 6-bits the Fv/Pv can reach a maximum of 63 instead of 50, but roughly anything higher than 55 causes some of the results of the internal calculations to exceed their 16-bit limit, and thus spurious results. 32-bit integers are also out due to ahem, memory.

To summarise, the new 14-byte sequence gives the following information

* 16-bit samples from each EEG channel
* Device frequency channel (CH.1, CH.2 etc)
* DBS active mode (DBS ON/OFF)
* Currently programmed DBS pulse-width
* Currently programmed DBS frequency.

**4.0 System Commands**

The following is a list of single-byte commands that the receiver accepts:

**4.1 – Basic Commands**

Switch DBS ON – 0x0A

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

Switch DBS OFF – 0x0B

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

Put System into SLEEP mode – 0x0C

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

WAKE/RESET System – 0x0D

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |

If DBS is currently active when the system is put into sleep mode, DBS will resume with most of the peripherals (and transceiver) powered down – necessary for overnight DBS runs. Otherwise everything will power down.

The WAKE/RESET command is the equivalent to taking the system battery out and putting it back in. DBS values will revert to their defaults, and the system will begin actively recording and transmitting EEG. Note that I can set the system to immediately enter SLEEP mode when the battery’s inserted – that’s a matter of preference.

The WAKE/RESET command results in a special 1-second burst transmission from the receiver to the EEG recorder. This is because in SLEEP mode, the EEG recorder’s transceiver is only waking to ‘sniff out’ packets for 1.95mS every second. Thus it’s advised that during this 1-second delay when the WAKE/RESET command’s been issued, no other commands are sent.

**4.2 – DBS Commands**

As explained there are four command types. The first two are:

Change Frequency Value – 0x41 to 0x72

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | Fv | Fv | Fv | Fv | Fv | Fv |

Change Pulse-Width Value – 0x81 to 0xB2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | Pv | Pv | Pv | Pv | Pv | Pv |

Where in both cases Fv/Pv ranges from 1 to 50. Values outside these limits are ignored. For reference, the following table highlights the Fv and Pv values along with their associated commands:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fv / Pv Value** | **Change Fv Command** | **Change Pv Command** | **Fv / Pv Value** | **Change Fv Command** | **Change Pv Command** |
| 1 | 0x41 | 0x81 | 26 | 0x5A | 0x9A |
| 2 | 0x42 | 0x82 | 27 | 0x5B | 0x9B |
| 3 | 0x43 | 0x83 | 28 | 0x5C | 0x9C |
| 4 | 0x44 | 0x84 | 29 | 0x5D | 0x9D |
| 5 | 0x45 | 0x85 | 30 | 0x5E | 0x9E |
| 6 | 0x46 | 0x86 | 31 | 0x5F | 0x9F |
| 7 | 0x47 | 0x87 | 32 | 0x60 | 0xA0 |
| 8 | 0x48 | 0x88 | 33 | 0x61 | 0xA1 |
| 9 | 0x49 | 0x89 | 34 | 0x62 | 0xA2 |
| 10 | 0x4A | 0x8A | 35 | 0x63 | 0xA3 |
| 11 | 0x4B | 0x8B | 36 | 0x64 | 0xA4 |
| 12 | 0x4C | 0x8C | 37 | 0x65 | 0xA5 |
| 13 | 0x4D | 0x8D | 38 | 0x66 | 0xA6 |
| 14 | 0x4E | 0x8E | 39 | 0x67 | 0xA7 |
| 15 | 0x4F | 0x8F | 40 | 0x68 | 0xA8 |
| 16 | 0x50 | 0x90 | 41 | 0x69 | 0xA9 |
| 17 | 0x51 | 0x91 | 42 | 0x6A | 0xAA |
| 18 | 0x52 | 0x92 | 43 | 0x6B | 0xAB |
| 19 | 0x53 | 0x93 | 44 | 0x6C | 0xAC |
| 20 | 0x54 | 0x94 | 45 | 0x6D | 0xAD |
| 21 | 0x55 | 0x95 | 46 | 0x6E | 0xAE |
| 22 | 0x56 | 0x96 | 47 | 0x6F | 0xAF |
| 23 | 0x57 | 0x97 | 48 | 0x70 | 0xB0 |
| 24 | 0x58 | 0x98 | 49 | 0x71 | 0xB1 |
| 25 | 0x59 | 0x99 | 50 | 0x72 | 0xB2 |

The remaining two command types change the frequency/pulse-width multipliers:

Change Frequency Multiplier to x0.1 – 0xCB

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |

Change Frequency Multiplier to x1 – 0xCA

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |

Change Frequency Multiplier to x10 – 0xC9

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

Change Frequency Multiplier to x100 – 0xC8

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

Change Pulse-Width Multiplier to x0.1 – 0xE4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

Change Pulse-Width Multiplier to x1 – 0xE5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |

Change Pulse-Width Multiplier to x10 – 0xE6

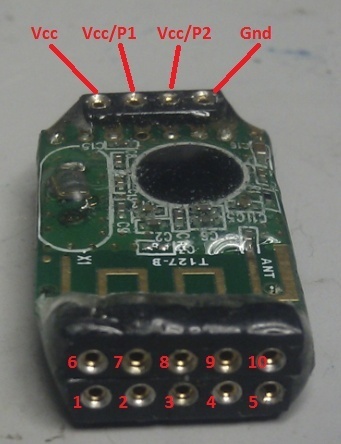
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |

Change Pulse-Width Multiplier to x100 – 0xE7

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |

**5.0 – System Pin Diagram**

This section shows the inputs and outputs of the system in diagram form. Note that there are 4 pins on the battery terminal, two of which are programming/debug pins for quick reprogramming of the device. When the battery is inserted the two programming pins are set to V+, hence the need for a custom battery connector. This is shown in figure 2:



**Figure 2 –** Device Pin Diagram

In figure 2 these are the pin assignments:

1 – EEG Channel 3 (10)

2 – EEG Channel 4 (11)

3 – EEG Ground

4 – DBS 1 Ground

5 – DBS 2 Ground

6 – EEG Channel 1 (00)

7 – EEG Channel 2 (01)

8 – EEG Reference

9 – DBS Channel 1

10 – DBS Channel 2

P1/P2 – Programming/Debug connection (shorted to Vcc when not programming / debugging)

Note that all EEG channels share the same reference.

**6.0 – Final Note**

I’ve currently not implemented the dual-channel operation onto the receiver, so the switch won’t do anything, besides interrupting the power whilst it’s flicked. Besides if two systems are used simultaneously, it’s better to have 2 computers and two receivers.