## 1.11 Program Maintenance Information

### 1.11.1 Q: Gosh, this must have been difficult to debug. How was it done?

**A:** Very carefully with built in debug code. To enable debug mode:

- Set bit 8 on the switch.
- Connect an RS-232 ASCII terminal to pin 7 and ground of U1 in the TXB-B as follows:
  - For a 25 pin RS-232 connector:
    - U1-7 (J4-square) to pin 3 on a DB-25 type connector.
    - Ground (J4-circular) to pin 7 on a DB-25 type connector.
  - For a 9 pin RS-232 connector:
    - U1-7 (J4-square) to pin 2 on a DB-9 type connector.
    - Ground (J4-circular) to pin 5 on a DB-9 type connector.

  Then on the left side of the screen there will be a command count and the command data as it was received from the Burle equipment. On the right side there is the command data that is being sent to the Spectra.

  Sometimes in debug mode Burle commands will be lost, but so what? You are running in debug mode, not real mode.

  The RS-232 ASCII terminal must be set up for:

  - 19200 baud
  - One start bit
  - One stop bit
  - Seven or eight data bits
  - No parity

### 1.11.2 Q: Does the debug display show up on the video monitor?

**A:** No it does not, it only shows up on an external serial ASCII terminal. It is a debugging aid and showing this type of information on a video monitor would make debugging the interface to the Spectra very difficult.

### 1.11.3 Q: What does this display look like? I saw the writeup in the footnote but it still isn't clear to me what it contains.

**A:** It is unlikely that anyone other than a software maintainer of the TXB-B program will ever need to know the answer to that question. However here is a small sample of what it looks like, with some notes as the contents of each field for a "pan left" command sequence from a variable speed keyboard.

```
      A B<-------------------->B   C<---------------->C   D<---------------->D
 1 " 8F 87 00 00 08  00 08 02 19   FF 01 00 04 0B 00 10 "
 2 " 90 87 00 00 08  00 18 02 29   FF 01 00 04 13 00 18 "
 3 " 91 87 00 00 08  00 30 02 41   FF 01 00 04 1F 00 24 "
 4 " 92 87 00 00 08  00 38 02 49   FF 01 00 04 23 00 28 "
 5 " 93 87 00 00 08  00 40 02 51   FF 01 00 04 27 00 2C "
 6 " 94 87 00 00 08  00 48 02 59   FF 01 00 04 2B 00 30 "
 7 " 95 87 00 00 08  00 50 02 61   FF 01 00 04 2F 00 34 "
 8 " 96 87 00 00 08  00 58 02 69   FF 01 00 04 33 00 38 "
 9 " 97 87 00 00 08  00 58 02 69 "
10  <Followed by 28 identical entries>
11 " B3 87 00 00 08  00 58 02 69 "
12 " B4 87 00 00 08  00 58 02 69   FF 01 00 04 33 00 38 "
13 " B5 87 00 00 08  00 58 02 69 "
14   <Followed by 28 identical entries>
15 " D1 87 00 00 08  00 58 02 69 "
16 " D2 87 00 00 08  00 58 02 69   FF 01 00 04 33 00 38 "
17 " D3 87 00 00 08  00 58 02 69 "
18   <Followed by 20 identical entries>
19 " E7 87 00 00 08  00 58 02 69 "
20 " E8 87 00 00 08  00 50 02 61   FF 01 00 04 2F 00 34 "
21 " E9 87 00 00 08  00 40 02 51   FF 01 00 04 27 00 2C "
22 " EA 87 00 00 08  00 20 02 31   FF 01 00 04 17 00 1C "
23 " EB 87 00 00 08  00 00 00 0F   FF 01 00 00 00 00 01   FF 01 00 00 00 00 01 "
24 " EC 86 00 00 07  05 6F 01      FF 01 00 07 00 21 29 "
25 " ED 86 00 00 07  15 34 56      FF 01 00 07 00 21 29 "
26 " EE 86 00 00 07  05 02 14      FF 01 00 07 00 02 0A "
      A B<-------------------->B   C<---------------->C   D<---------------->D
```

Note that the numbers on the left ($1 \rightarrow 26$), the letters on the top ($A \rightarrow D$) and the quote marks " are only used for documentation and do not appear on the screen. Also please note that the statements about repeating lines are not in the original.

Notes to the above sample listing.

**A** This is a hex field that increments by one for each command received from a Burle head end. It is an 8 bit number that wraps at `0xFF` around to `0x00`. It is rare that the total number of commands sent from a head end is important, but it is quite often important to know if two or three identical commands were sent. So by carefully watching the Ⓐ column, it is easy to identify multiple command sendings. (As long as there are less than 256 of them!)

**B** This is the command as output by the Burle equipment.

1. The first byte is a Burle format command. Note that it has its most significant bit set and that the rest of the byte is the number of following bytes. There are only two valid values for this byte: it may be either `0x87` or `0x86`, any other value and its corresponding command is thrown away. If the most significant bit on any byte after the first is set then the whole message is thrown away.
2. Then come two bytes as address fields of the command. The first one is used by the TXB-B to determine if the message might be for it. The second one is used by the

TXB-B and the Spectra to actually receive the message. If the second one does not match the address switches on the Spectra the message is thrown away. If first address byte does not match the address switches on the TXB-B the message is thrown away.

3. The fourth byte is the Burle opcode. The TXB-B processes opcodes `2`, `4`, `5`, `6`, `7` and `8`. Others are thrown away (I hope).
4. Next come either two or three bytes of Burle formatted command data. These are the data bytes of the command, with some op codes there are only two data bytes. To get the exact meaning of the various bits in each byte for each opcode see the protocol document.
5. Then comes a checksum for the command.

**C** This is the D protocol command generated by the TXB-B. The various bytes are:

1. The first byte is the "sync" byte and always consists of all ones.
2. The next byte is the D protocol address byte which gets sent to the Spectra. Note that it is always one greater than the address in the Burle protocol command. The Burle protocol is "zero based" and the D protocol is "one based". In all of this example the commands were sent to unit #`1`, i.e. the Spectra and the Burle keyboard were both set to `1`.
3. Then next two bytes are the "Command 1" and "Command 2" bytes of the D protocol.
4. The command bytes are followed by the "Data 1" and " Data 2" bytes of the D protocol.
5. Last there is the D protocol checksum. Note that it does not include the "sync", or first byte in it.

**D** Some commands have a stop motion command following them. These are indicated here. The fields are identical to those shown for block ©. This "stop all" command is automatically generated about 100 ms, following any "motion command"

It should be noted that:

1. The pan bit is set on most of the D protocol commands.
2. The pan speed changes the Burle protocol. This might not be as obvious as it might be as this field crosses two nibbles. I moved the joy stick rapidly and it did not actually hit all 16 steps of the pan speed.
3. The pan bit is set on second command byte of D protocol.
4. The pan speed is indicated in data byte 1 and starts out at other than "zero speed" and goes to a high speed (not "turbo speed", I didn't push the joy stick enough) and then back down.

5. In lines 9 through 11, and again twice more, many commands came into the TXB-B and nothing was sent out. But that after every 30 commands a new D protocol pan command was generated. (In future versions of the TXB-B software this interval may change. Also each keyboard generates repeats at different rates.) This is done to reduce the load on the Spectra as none of the Pelco keyboards continuously send commands. While Burle does send them at least 20 times a second. Every few seconds a command must be sent in D protocol, or the Spectra will time out and assume that it was being told to "run away". (Run away protection consists of stopping motion after about 15 seconds following the last motion causing command.

6. Most commands are eight bytes long, however the last three are only seven bytes long. For some reason or other, Burle does not include the first byte in its length field. Thus the length fields are off by one.

7. At lines 24 and 25 two different commands, SHOT 111 and SHOT 180, are input and generate the same output command. Both of these commands are used by Burle to "flip" a camera.

8. On line 26 is a normal SHOT 2 command.

### 1.11.4 Q: In addition to the serial debug data that you output, are there any other debug aids that will be left in the final product?

**A:** Yes, there are two types of built in debugging features that have been left in the final product.

1. The following pins are pulsed as debug outputs when various events occur.

| Port C Pin/Marking | Name | Use |
|---|---|---|
| 18/E8 | `EDGE` | Found an edge in the Manchester data |
| 19/E7 | `FRAMESYNC` | Found a sync pulse |
| 20/E6 | `BYTEFLAG` | True from the start of a byte until 4 bit times through the byte |
| 21/E5 | `BYTESYNC` | Pulsed when other than the first byte is detected |
| 22/E4 | `INTIMEOUT` | The no turbo mode timer is running |
| 23/E3 | `ADDRESSERROR` | Pulsed on detecting an address error |
| 24/E2 | `CHECKSUMERROR` | Pulsed on detecting a checksum error |
| 25/E1 | `PARITYERROR` | Pulsed on detecting a parity error |

2. The following jumper positions have various signals available on them. None of these jumpers are installed, but the two holes are there where it is easy to make a connection to them. In all cases the "circular" pin-hole is for ground and the "square" pin-hole is the signal.

| Jumper | Use |
|---|---|
| J2 | Monitor point for data received from the Spectra |
| J3 | Monitor point for data sent to the Spectra |
| J4 | Serial debug data output |
| J5 | "Cleaned up" input data |

### 1.11.5 Q: What are the new part numbers for the TXB-B?

**A:** An effort was made to not introduce any new part numbers into the Pelco supply system. As a result there are very few additional part numbers required for the TXB-B. The most significant of these are:

1. `PA05-0029-00x0` Is the fully assembled board with many other things on it.
2. `IC01-0928-0057` PIC 16C57C chip that gets stuff written into.
3. `IC51-0036-0111` The programmed IC that gets stuck on the board.
4. `PG51-0036-0111` All combined binary hex files. There is only one `.HEX` file on this project.
5. `BH51-0018-0111` Single binary hex file in `.HEX` format
6. `FW00-0116-0111` Source code for this project. Consists of four files:
   A. The main source file, `TXB-B-U1.ASM`.
   B. MicroChip's chip include file, `P16C57.INC`.
   C. A file of macros, `MACRO.INC`.
   D. A file of protocol definitions, `PROTOCOL.INC`.

Each time a software revision is made, the revision number on the `IC51`, `PG51`, `BH51` and `FW00` parts, must be updated.