Download Protocol

The following is based on the booter.spin file released by Chip Gracey of Parallax as an attachment to his message posting to the Parallax Propeller forum titled: "Propeller ROM source code HERE".

Though there is no explicit copyright in the message or the attached source code, this Wiki editor believes U.S. copyright law interprets that as copyrighted with all rights reserved. However, this Wiki editor is not a lawyer.

In the Parallax Propeller forum thread titled "Standalone, cross-platform Propeller assembler", Chip Gracey of Parallax also posted a message with the attachments: source for a download program written in Delphi Pascal, and a text explanation of the download protocol.

Also posted by Chip Gracey of Parallax is a SPIN object for loading a Propeller from a Propeller, in the thread titled "Propeller Loader"

Chip Gracey's "Minimal Spin bootstrap code for assembly language launch" is also a handy reference.

Three Bit Protocol (3BP)

The Propeller uses an adaptive Three Bit Protocol (3BP) for serial signaling which works at most any bit rate, at the cost of reduced throughput because of signaling overhead. 3BP symbols begin high with an idle line, followed by either one or two bit times low to signal a one or zero bit respectively. These two bit times are referred to as T1 and T2.

As part of the handshake sequence, the Propeller calculates a threshold of 1.5 bit times in system clock cycles. If the RXD pin is low for longer than this threshold, a zero symbol is received, otherwise a one symbol is received.

Since the Propeller boots using its internal RC clock (RCFAST), which can vary in frequency from 8 MHz to 20 MHz, all timings are at best approximations. Any timeouts are calculated assuming a 20 MHz clock.

3BP Meets 8N1

Assume that 3BP(X) and 8N1(X) map data X into a unique symbol that can be signaled on a serial line, then:

- **3BP Symbols**
- 3BP(idle): 1

- 3BP(0): Two bit times low which is 001 transmitted least significant bit first.
- 3BP(1): One bit time low, which is 101 transmitted least significant bit first.

- **8N1 Symbols**
- 8N1(idle): 1
- 8N1(8 data bits): start bit (0) + 8 data bits, least significant first + stop bit (1)

Transmitting one 3BP symbol per 8N1 symbol is simplest, but very inefficient:

- 3BP(0) = 8N1(FE)
- 3BP(1) = 8N1(FF)

It is possible to pack one or more 3BP symbols into a traditional ten bit time 8N1 serial symbol!

Fixed length 3BP-to-8N1 symbol packing looks like this:

8N1 Symbol	Line State	3BP Symbol
Idle or Stop	1	Idle
Start	0	T1
Bit 0	Data Bit 0	T2 or Idle
Bit 1	1	Idle
Bit 2	0	T1
Bit 3	Data Bit 1	T2 or Idle
Bit 4	1	Idle
Bit 5	0	T1
Bit 6	Data Bit 2	T2 or Idle
Bit 7	1	Idle
Stop	1	Idle

It is also possible to do variable length packing to reduce the back-to-back idle bit times, but it is more complex to do as it depends on the symbols being sent.

Handshake Sequence

All communication with the Propeller is done using 3BP.

There is a timeout on completion of the entire handshake sequence of 375,000 cycles (150 ms at 20 MHz). There is a reset delay of 50 ms, and then COG 0 is loaded, which takes 512 longs * 16 cycles per hub access = 8,192 cycles.

Important: With a 150 ms limit for the handshake, which requires 251 8N1 symbols to be transmitted (2510 bits), the closest standard minimal bitrate required is 19200 BPS. With a more conservative 100 ms for the handshake, 38400 BPS is recommended as a minimum bitrate. For lower bit-rates, multiple 3BP symbols must be sent per 8N1 symbol.

1. Transmit on RXD: 3BP(idle).
2. Reset the Propeller (on Parallax development boards: set DTR low, wait at least 10 ms, set DTR high).
3. Wait 100 ms. This leaves roughly 100 ms to complete the handshake.
4. Transmit on RXD: 3BP(0) followed by 3BP(1), which is equivalent to 8N1(F9). This sets the bit timing threshold.
5. Transmit on RXD: 250 bits from a Linear Feedback Shift Register (LFSR):

```
Initialize the LFSR to ASCII 'P' (50 hex)
    Repeat 250 times:
        Transmit on RXD:  3BP(LFSR bit 0), which is equivalent to 8N1(
FE) if zero or 8N1(FF) if one.
        Set rotate-in bit to even parity of LFSR AND B2.
        Rotate LFSR left one bit.
```

which is equivalent to:

```
Initialize the LFSR to ASCII 'P' (50 hex)
    Repeat 250 times:
        Transmit on RXD:  3BP(LFSR bit 0), which is equivalent to 8N1(
FE) if zero or 8N1(FF) if one.
        Set LFSR to (shift LFSR left 1 bit) OR (
            (
                    (shift LFSR right 7 bits)
                XOR (shift LFSR right 5 bits)
                XOR (shift LFSR right 4 bits)
                XOR (shift LFSR right 1 bit)
```

```
            ) AND 1
        )
```

The first 250 bits of the LFSR encoded with 8N1 are:

```
FE FF FE FF FF FF FE FE FF FF FF FF FE FF FE FF
FF FF FF FF FE FE FE FF FF FF FE FE FF FE FF FE
FE FE FF FF FF FF FE FE FE FE FF FE FE FF FE FE
FF FE FF FF FF FF FE FE FF FE FE FE FF FE FE FE
FF FF FE FF FF FE FF FF FF FE FE FE FF FE FF FF
FE FE FF FF FE FE FF FE FF FF FE FF FF FE FE FF
FE FE FF FF FF FE FF FE FF FE FF FE FF FF FF FE
FF FF FE FF FE FF FF FE FF FE FE FF FF FF FF FF
FF FF FF FE FE FF FF FE FF FF FF FF FE FF FF FF
FE FF FE FE FE FE FE FE FE FF FE FF FE FF FF FE
FE FE FF FF FE FE FF FF FF FE FE FE FE FE FE FF
FF FF FF FF FE FF FE FE FF FE FF FE FF FE FE FF
FE FE FE FE FE FF FE FE FE FE FF FF FF FE FF FF
FF FF FF FF FE FF FF FE FE FE FE FF FF FE FE FE
FF FE FE FF FF FE FE FE FE FE FF FF FE FF FE FE
FE FF FE FF FE FE FF FF FE FF
```

Note that the LFSR only requires an 8 bit wide register to calculate, as all higher bits are ignored.

Identify Sequence

All communication with the Propeller is done using 3BP.

The Propeller relies on received pacing symbols to frame a 3BP response for the next 250 bits of the LFSR.

There is a timeout on transmitting each bit of 250,000 cycles (100 ms at 20 MHz), and it is reset with each bit the Propeller sends.

1. Receive on TXD: the next 250 bits of the LFSR, and verify.

```
Repeat 250 times:
    Transmit on RXD:  3BP(1) followed by 3BP(0) for pacing, which is e
quivalent to 8N1(F9).
    Receive on TXD:  3BP(0) or 3BP(1) for the next bit of the LFSR, an
```

```
d verify
```

The next 250 bits of the LFSR encoded with 8N1 are:

```
FE FF FE FE FE FE FF FE FF FF FF FE FE FF FF FF
FF FE FF FE FF FF FF FF FF FE FE FE FF FF FF FE
FE FF FE FF FE FE FE FF FF FF FF FE FE FE FE FF
FE FE FF FE FE FF FE FF FF FF FF FE FE FF FE FE
FE FF FE FE FE FF FF FE FF FF FE FF FF FF FE FE
FE FF FE FF FF FE FE FF FF FE FE FF FE FF FF FE
FF FF FE FE FF FE FE FF FF FF FE FF FE FF FE FF
FE FF FF FF FE FF FF FE FF FE FF FF FE FF FE FE
FF FF FF FF FF FF FF FF FE FE FF FF FE FF FF FF
FF FE FF FF FF FE FF FE FE FE FE FE FE FE FF FE
FF FE FF FF FE FE FE FF FF FE FE FF FF FF FE FE
FE FE FE FE FF FF FF FF FF FE FF FE FE FF FE FF
FE FF FE FE FF FE FE FE FE FE FF FE FE FE FE FF
FF FF FE FF FF FF FF FF FF FE FF FF FE FE FE FE
FF FF FE FE FE FF FE FE FF FF FE FE FE FE FE FF
FF FE FF FE FE FE FF FE FF FE
```

2. Receive on TXD: the 8 bit Propeller chip version.

```
Repeat 8 times:
    Transmit on RXD:  3BP(1) followed by 3BP(0) for pacing, which is e
quivalent to 8N1(F9).
    Receive on TXD:  3BP(0) or 3BP(1) for the next bit of the Propelle
r chip version.
```

Transfer Sequence

All communication with the Propeller is done using 3BP.

All data is sent as 32 bit longs, least significant bit first. There is roughly 100 ms timeout on receiving each long.

1. Transmit on RXD a 32 bit command:
   Command = 0: Shutdown.
   Command = 1: Load RAM and launch.
   Command = 2: Load RAM, write and verify EEPROM, and shutdown.

Command = 3: Load RAM, write and verify EEPROM, and launch.

2. Transmit on RXD; 3BP(N), a 32 bit count of the number of 32 bit longs to follow. There is a maximum of 8192 longs allowed (32 KB).
3. Transmit on RXD: 3BP(N x 32 bit longs). If less then 8192 longs, then the remaining 8192 - N are set to 0. The 8192 longs must have a 0 checksum.
4. Transmit on RXD every 10 ms: 3BP(0) followed by 3BP(1) for pacing, which is equivalent to 8N1(F9).
5. Receive on TXD: the checksum result 3BP(0) = passed or 3BP(1) = failed, which is equivalent to 8N1(FE) = passed and 8N1(FF) = failed.

DeSilva has prepared a quite readable article about self-clocking data streams and the "Three-Bit-Protocol" here (Propeller Forum) some time ago...