Propeller® chip bootloader protocol

Andrey Demenev

January 25, 2010

1 Introduction

Parallax® Propeller® chip has a built-in bootloader used to upload program code into on-chip RAM and external EEPROM, and verify the code uploaded. The bootloader uses a serial protocol with data rate auto-detection.

This document describes the serial protocol used by bootloader. This description is based on analysis of source code available on Parallax forums and Propeller Object Exchange. The author is not related with Parallax, Inc. This document should not be considered as official information. All information is provoded on "AS-IS" basis, without warranty of any kind.

2 Definitions

Propeller Parallax® Propeller® chip

Host Device performing the task of loading program code into Propeller

RXD Propeller pin 31. Host sends data on this line.

TXD Propeller pin 30. Propeller sends data on this line.

RESET Propeller pin RESn.

3 The protocol

Timeouts mentioned in protocol description are specified for 20 MHz Propeller system clock, although after startup Propeller runs at about 12 MHz. This allows some reasonable threshold. Host should use timeouts specified in this document without recalculating them for 12 MHz clock.

3.1 How the host sends data

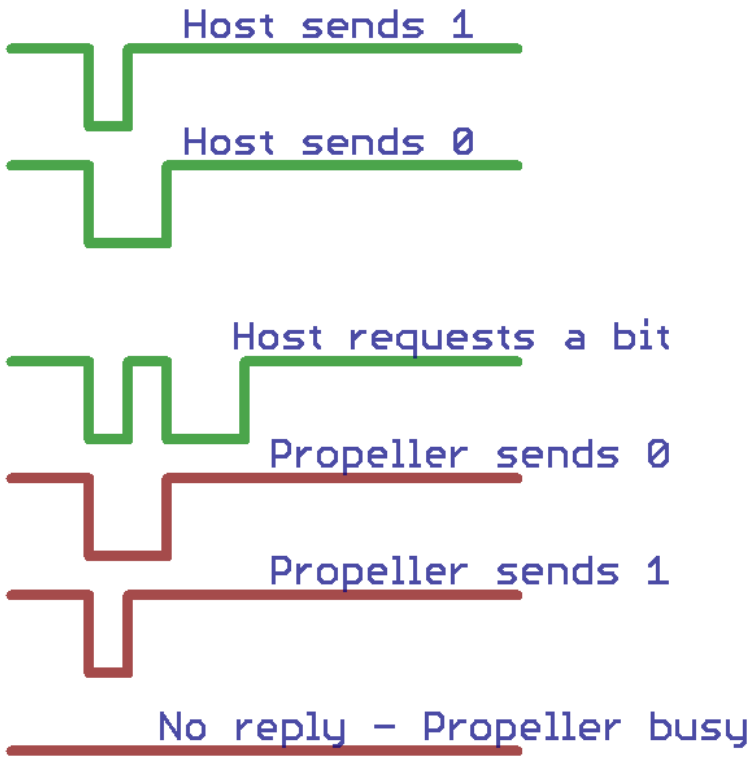
Data is sent serially, one bit at a time. In idle state, when no transmission is in progress, both RXD and TXD are high. The host sends a bit by issuing a low pulse on RXD. One is sent with a short pulse (1t), zero with 2 times longer pulse (2t). Bits should be separated by keeping RXD high at least 1t. The exact time of 1t depends on host hardware capabilities. 5 to 100 microseconds should work. Sending more than 1 bit is done in LSB manner.

3.2 How Propeller sends data

Propeller sends data on request from host only. Host sends 2 bits - one followed by zero. During first pulse, Propeller holds TXD low. When RXD is set high by host, Propeller sets TXD to the value of the bit it wishes to transmit. On the falling edge of second pulse on RXD, Propeller sets TXD high.

If Propeller does not hold TXD low during first pulse sent by host, the host must retry the request in 10 ms. Number of retries depends on timeout value. If host does not receive a reply from Propeller before timeout expires, this is considered a communication error.

Figure 1: Communication patterns



3.3 Bootloader procedure

During the bootloader procedure, there are moments when Propeller waits for data from the host, or the host waits for data from Propeller. Depending on protocol stage, the wait period (timeout) can change. If timeout expires when Propeller is waiting for data, Propeller terminates the bootloader process, and launches program from external EEPROM.

If timeout expires when the host is waiting for data, host terminates the bootloader procedure with error. In the foolowing sections timeout is assumed to be 100 ms, if not specified explicitly.

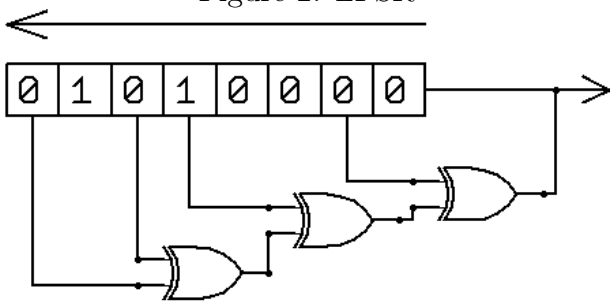
3.3.1 Bootloader init

RXD and RESET are host outputs, TXD is host input. Code loading is initiated by host by pulling RESET low. While keeping RESET low, host sets RXD high, then releases RESET, by either setting it high, or leaving it floating. After releasing RESET, host should wait at least 50 ms to allow Propeller startup.

3.3.2 Data rate calibration

Timeout is set to 150 ms. Now host sequentally sends 2 bits - one followed by zero. This allows Propeller to measure 2 pulses, one short, and another 2 times longer, in order to determine data rate.

Figure 2: LFSR



3.3.3 Link reliability check

After calibration, a pseudorandom sequence of bits is sent in both directions in order to check reliability of data link. Both Propeller and host use identical pseudorandom generators, so both know the bit they expect to receive next.

Pseudorandom generator is a 8-bit Fibonacci LFSR with $B2 taps (see Figure 2). LFSR initial value is 80 decimal ($50 hexadecimal, %01010000 binary, 'P' ASCII).

Host sends 250 bits obtained from LFSR. After receiving each bit, Propeller compares it with bit obtained from its own LFSR. After 250 bits, host and Propeller exchange roles, and 250 bits are sent in opposite direction. If during this stage either host or Propeller detect bit mismatch, they perform the same termination procedure as in case of timeout.

3.3.4 Version check

After link check, host reads Propeller chip version. Chip version is sent by propeller as a byte, LSB first. Host compares version sent by Propeller with the version it expects, and terminates in case of mismatch.

3.3.5 Command

If version check was successfull, host sends a 32-bit command. The following 4 commands are supported:

0 No action, Propeller will shut down. This can be used to check if Propeller is connected, without loading program code.

1 Load program into Propeller RAM and launch it.

2 Load program into external EEPROM and shut down.

3 Load program into external EEPROM and launch it.

If command sent was 0, the Propeller shuts down, and the host terminates with success.

3.3.6 Program code download

If command was 1, 2 or 3, host sends number of LONGs in the program code as a 32-bit number. The host can obtain this number from binary file. Compiled binaries for Propeller

contain program length at byte offsets 8 and 9. Byte 8 contains lower 8 bits, and byte 9 - higher 8 bits. Program length is specified in BYTEs. Before sending to propeller, host divides the number by 4 to convert from BYTEs to LONGs.

The host continues by sending the contents of bynary program file to Propeller.

3.3.7 Checksum

After successfull code download, Propeller calculates checksum. If checksum does not match, Propeller sends bit 1 and shuts down, host on reception of 1 terminates with error. Host should allow 250 ms timeout for checksum reply from propeller.

If checksum matches, Propeller sends 0 response bit. If command was 1, Propeller shuts down, and host terminates with success. Otherwice, the procedure continues.

3.3.8 EEPROM programming

Propeller writes the program code into external EEPROM. When finished, it sends a 0 bit. If an error occured during programming, Propeller sends a 1 bit, and shuts down, and host on reception of 1 terminates with error. Host should allow 5 seconds timeout for programming result response from Propeller.

3.3.9 EEPROM verification

Propeller verifies the program code written into external EEPROM. When finished, it sendsa 0 bit. If an error occured during verification, Propeller sends a 1 bit, and shuts down, and host on reception of 1 terminates with error. Host should allow 2 seconds timeout for verification result response from Propeller.

If command sent was 2, Propeller shuts down. Otherwise, it launches the program downloaded from host.

Host terminates with success.

3.4 Using UART to load Propeller

If we look at Figure 1, we can see that communication between host and Propeller can be performed using a standard UART hardware, such as a PC serial port or UART included with many microcontrollers. Indeed, each pattern starts with a zero bit corresponding to start bit in serial protocol, followed by a maximum of 3 data bits. With long enough pause between sending bits, each pattern can be generated or received using a UART.

Bit 1 can be sent by host as $FF byte, and bit 0 - as $FE. When host requests data from Propeller, it can send $F9 byte. Reply $FF corresponds to bit 1 sent by Propeller, and $FE corresponds to 0.

It can be easily seen that host can encode 3 to 5 bits in each byte it sends over serial port when transmitting, and receive 2 bits from Propeller in a byte. This can reduce bandwidth loss.