# BASIC Stamp Programming Protocol

## INTRODUCTION

This document describes the protocol (media, timing, etc.) used to program BASIC Stamp 2, 2e, 2sx, 2p and 2pe modules with tokenized data (PBASIC tokens). Though the actual steps for implementing the BASIC Stamp programming protocol can be listed in less than one page (see the example), this document attempts to discuss all aspects in detail.

Note that, in the context of this document, the term "programming" means: delivering the appropriate signals to convey tokenized data that describes a PBASIC program. The tokenization process (converting source code to numbers) is not described by this document.

This document is organized into the following sections:

| Section Title | Description |
|---|---|
| • Protocol Overview | Summarizes the entire protocol. |
| • Programming Interface | Discusses the programming interface connections. |
| • Protocol Details | Presents in-depth detail about each protocol component. |
| • Example: Programming a BS2 | Demonstrates the steps necessary to program a BASIC Stamp 2 module. |

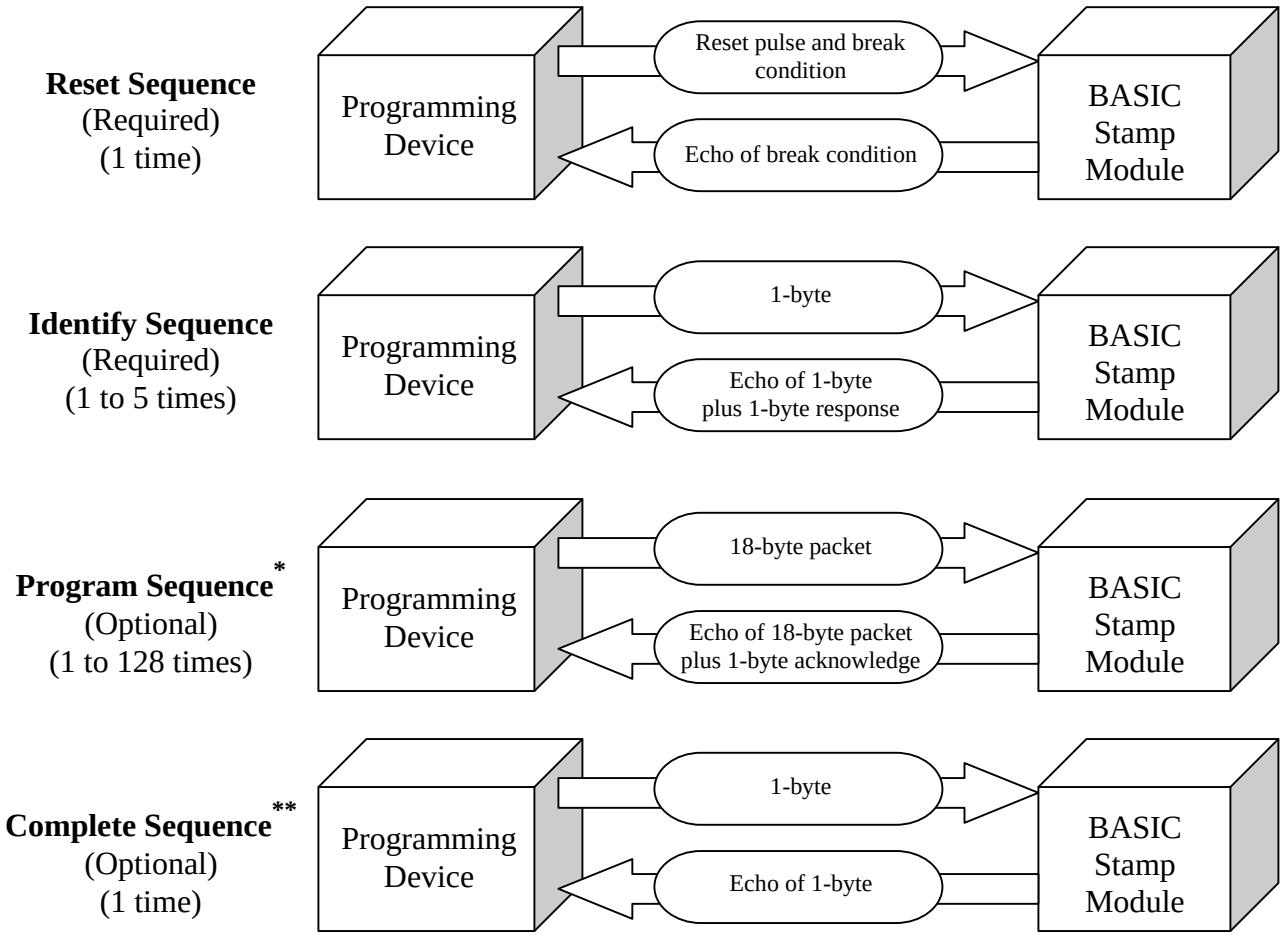
## PROTOCOL OVERVIEW

The BASIC Stamp programming protocol consists of four components:

| Component | Purpose | Function |
|---|---|---|
| 1) Reset | Start of communication. | Causes a hardware-reset of the BASIC Stamp module and indicates to the BASIC Stamp that it is about to be identified and possibly reprogrammed. |
| 2) Identify | Find module and version. | Determines the module-type (BS2, BS2e, etc) and firmware revision (1.0, 1.1, etc.) of the BASIC Stamp module. |
| 3) Program | Transmit tokens. | (Optional) Transmits the tokenized program information. |
| 4) Complete | End of communication. | (Optional) Signals the end of identify or program session. |

For the purposes of this document, the term "programming device" refers to any device capable of generating the proper signals on the BASIC Stamp's programming port. Most commonly, the programming device is a computer such as a PC or Mac, but could be a palmtop device or even another microcontroller. Figure 1 shows a summary of how the programming device interacts with the BASIC Stamp.

**Figure 1: Overview of BASIC Stamp Programming Protocol**



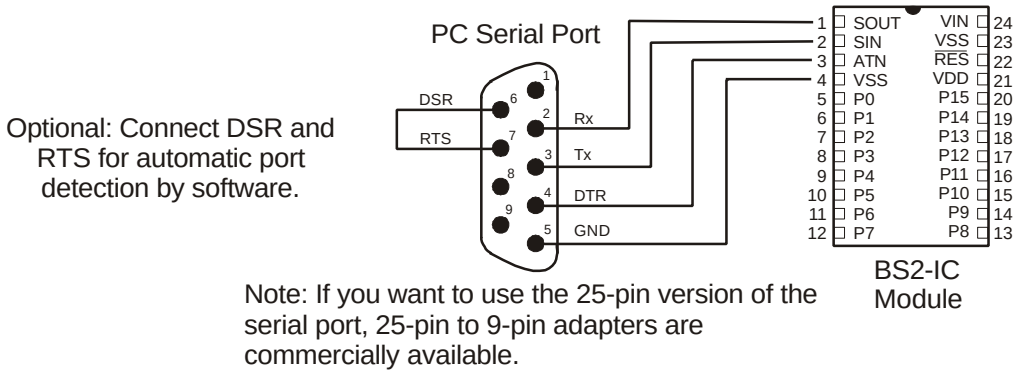

***** For a BS2e, BS2sx, BS2p or BS2pe, the program sequence (if used) is preceded by the transmission of a 1-byte value (0 to 7) representing the program slot to download to. A 1-byte echo must then be received and then the normal Program Sequence is executed.

** The complete sequence is simply the transmission of a 1-byte value (0) to tell the BASIC Stamp that the session is finished. This is optional. The BASIC Stamp will reset if no serial communication is detected for 2.3 seconds.

## PROGRAMMING INTERFACE

The BASIC Stamp modules (BS2, BS2e, BS2sx, BS2p and BS2pe) are programmed via an RS-232 serial interface. Any device that follows the RS-232c specifications (signal voltage, timing, etc.) should be capable of programming these BASIC Stamp modules. The serial port settings must be 9600 baud, N, 8, 1. Figure 2 shows the port connection diagram used by the programming interface.

**Figure 2: BASIC Stamp to programming port connection diagram.**




Note that the DSR to RTS loopback connection (DB9-pin 6 to DB9-pin 7) is optional. Many BASIC Stamp development boards, such as the Carrier Board and the Board of Education, make this connection. The DOS editors and older Windows editors (version 1.1 and before) use this loopback to automatically detect the serial port to which a BASIC Stamp is connected. The current Windows editor uses a different method and simply notes whether or not the loopback connection is present.

Except the method mentioned above, BASIC Stamp port detection methods are not suggested by this document. Software developers may use whatever method is appropriate for their application.

The circuitry in the BASIC Stamp causes all data sent from the programming device to be echoed back to the programming device. Because of this, for all bytes that the programming device transmits, it must be sure to read the same number of bytes from the receive pin, discard them, and then read the 1-byte response. Assuming the device uses a UART chip, it simply needs to flush the proper number of bytes from the receive buffer before trying to read the BASIC Stamp's response.

## PROTOCOL DETAILS

### RESET SEQUENCE

The BASIC Stamp module cannot be reprogrammed via the serial port without a reset sequence. This feature is designed to protect against accidental reprogramming while using the built-in serial port for normal PBASIC operation.

The reset sequence consists of:

1. Performing a hardware reset of the BASIC Stamp module.
2. In parallel with the hardware reset, a carefully timed "break" condition on the transmit line of the serial port is executed.

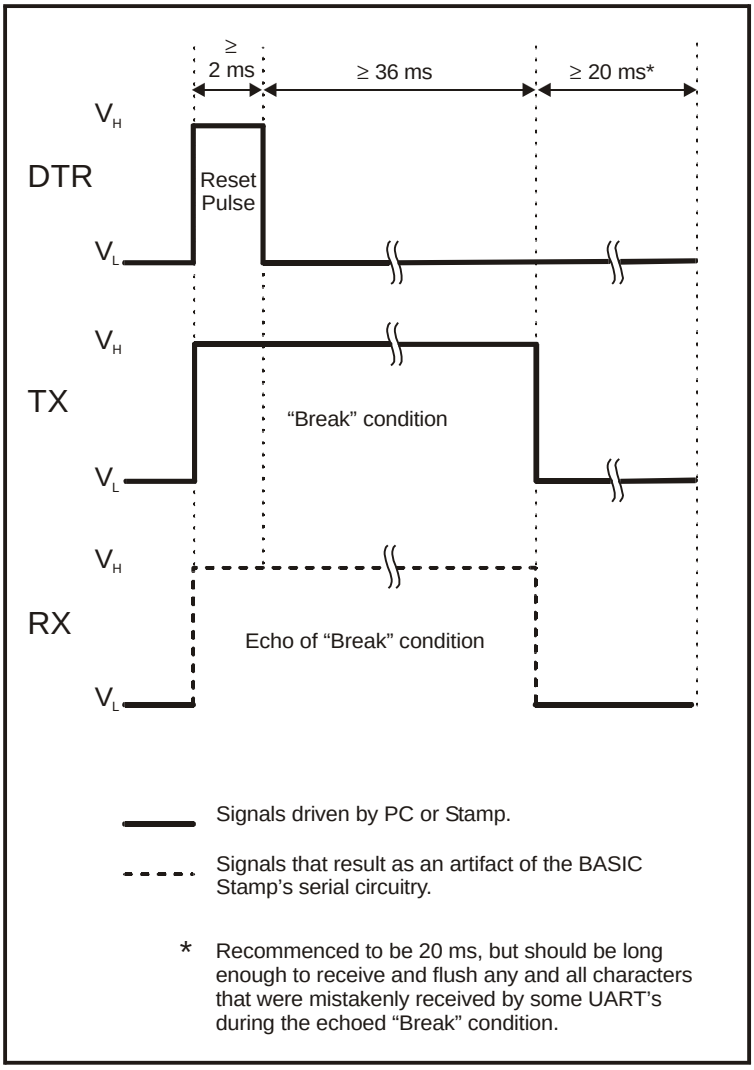
**It is important to note that the reset sequence can be performed at any time to reset the BASIC Stamp. For example, when trying to detect which BASIC Stamp is connected to a given serial port, if one identify routine fails, you can immediately perform a reset sequence followed by a different identify routine.**

The DTR pin (DB9 pin 4) of the serial port is connected to the BASIC Stamp's ATN pin (see Figure 2). Because of this connection, pulsing the DTR pin high then low again performs the hardware reset on the BASIC Stamp.

A "break" condition is a feature of the RS-232c serial protocol where the transmit pin (TX) is set high for a duration longer than the byte-period of the selected baudrate. This is an exception case to the normal use of the transmit pin. **The break condition cannot be emulated! It may seem that transmitting a series of binary 0s will achieve the same effect, however, doing so will still result in each byte being followed by a stop bit; effectively dropping the line low every 1.04 ms.**

Figure 3 shows the timing requirements of the DTR, TX and RX serial port pins during the reset sequence. Note that the transmit line's break condition must remain for at least 36 ms after the DTR pin goes low. This allows plenty of time for the BASIC Stamp to complete its hardware-reset process and for at least 1 byte-period to pass. The BASIC Stamp editor software waits for 50 ms after the fall of DTR before releasing the break condition.

**Figure 3: Reset Timing Diagram**



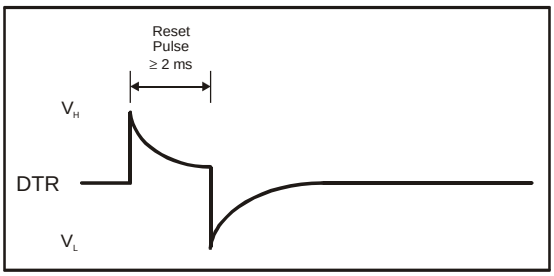

In summary, the reset sequence consists of the following steps:

1. Set DTR high.
2. Set break condition on TX.
3. Pause for at least 2 ms.
4. Set DTR low.
5. Pause for at least 36 ms.
6. Clear break condition on TX.
7. Pause for approximately 20 ms and flush receive buffer.
8. Execute the identify sequence.

**Some UARTs mistakenly receive characters during the echo of the break condition.** These characters must be flushed from the receive buffer after the break condition ends. The 20 ms delay after the end of the break condition is a flexible delay and need only be long enough to receive and flush the receive buffer.

Most BASIC Stamp development boards contain two 0.1 µF capacitors on the DTR pin. This allows the BASIC Stamp to use the programming port for serial communication with standard terminal software. This configuration still allows the reset pulse to pass through, but will change the waveform of the DTR signal to something similar to that of Figure 4.

**Figure 4: Programming Timing Diagram**




## IDENTIFY SEQUENCE

The Identify sequence immediately follows the reset sequence. After the reset sequence, all communication between the programming device and the BASIC Stamp is through normal transmission and reception of bytes.

The identify process is unique to each BASIC Stamp. (In the steps below, all references to "restart" means: start over with another reset sequence and a different identify sequence, or quit).

For a BASIC Stamp 2:

1) Transmit a capital "B" (ASCII 66).
2) Receive and discard echo of "B".
3) Receive byte. If it is not 2's-complement of "B" (ASCII 190), this is not a BS2, restart.
4) Transmit a capital "S" (ASCII 83).
5) Receive and discard echo of "S".
6) Receive byte. If it is not 2's-complement of "S" (ASCII 173), error, restart.
7) Transmit a "2" (ASCII 50).
8) Receive and discard echo of "2".
9) Receive byte. If it is not 2's-complement of "2" (ASCII 206), error, restart.
10) Transmit a 0. This is the "Version Number" command.
11) Receive and discard echo of 0.
12) Receive byte. This is a BCD form of firmware version. Ex: A value of 10 means firmware version 1.0. A value of 12 means firmware version 1.2.
13) Execute the program sequence, the complete sequence, or just quit.

For a BASIC Stamp 2e:

1) Transmit an "e" (ASCII 101).
2) Receive and discard echo of "e".
3) Receive byte. If it is an "e", this is firmware version 1.0, else restart.
4) Execute the program sequence, the complete sequence, or just quit.

For a BASIC Stamp 2sx:
1) Transmit a capital "X" (ASCII 88).
2) Receive and discard echo of "X".
3) Receive byte. If it is an "X", "Y" or "Z" (ASCII 88, 89 or 90), this is firmware version 1.0, 1.1 or 1.2, respectively, else restart.
4) Execute the program sequence, the complete sequence, or just quit.

For a BASIC Stamp 2p:
1) Transmit a capital "P" (ASCII 80).
2) Receive and discard echo of "P".
3) Receive byte.
    a. If it is a "p", "q", "r", "s", (ASCII 112, 113, 114, 115), etc., this is a BS2p24, with firmware version 1.0, 1.1, 1.2, 1.3, etc., respectively.
    b. If it is a "P", "Q", "R", "S", (ASCII 80, 81, 82, 83), etc., this is a BS2p40, with firmware version 1.0, 1.1, 1.2, 1.3, etc., respectively.
4) Execute the program sequence, the complete sequence, or just quit.

For a BASIC Stamp 2pe:
5) Transmit a capital "I" (ASCII 73).
6) Receive and discard echo of "I".
7) Receive byte.
    a. If it is a "i", "j", "k" (ASCII 105, 106, 107), etc., this is a BS2pe24, with firmware version 1.0, 1.1, 1.2, etc., respectively.
    b. If it is a "I", "J", "K" (ASCII 73, 74, 75), etc., this is a BS2pe40, with firmware version 1.0, 1.1, 1.2, etc., respectively.
8) Execute the program sequence, the complete sequence, or just quit.

## PROGRAM SEQUENCE

The program sequence, if needed, immediately follows the identify sequence.

For the BS2, the program sequence consists of 1 or more iterations of the following 3 items:
1. 18-byte packet (transmitted to the BASIC Stamp).
2. Echo of 18-byte packet (received from the BASIC Stamp).
3. 1-byte acknowledge (received from the BASIC Stamp).

For the BS2e, BS2sx, BS2p and BS2pe, the program sequence consists of:
1. 1-byte program slot number (0 to 7) (transmitted to the BASIC Stamp).
2. Echo of 1-byte program slot number (received from the BASIC Stamp).

Followed by 1 or more iterations of the following 3 items:
1. 18-byte packet (transmitted to the BASIC Stamp).
2. Echo of 18-byte packet (received from the BASIC Stamp).
3. 1-byte acknowledge (received from the BASIC Stamp).

The data for the 18-byte packet(s) is provided by the tokenizer. After tokenization, the packetized data is located in the ModuleRec.PacketBuffer structure. The ModuleRec.PacketCount field indicates the number of packets that need to be downloaded.

**No manipulation of the data is necessary.** The downloading routine simply needs to transmit 18 sequential bytes of the packet buffer to the BASIC Stamp, receive and discard the echoes and receive and check the acknowledge byte from the BASIC Stamp.

The acknowledge byte received from the BASIC Stamp should be one of the following values:

0 = Packet received and processed properly.
1 = Communication error (checksum didn't match received data).
2 = EEPROM error (at least one EEPROM location failed to retain data).

In summary, the program sequence consists of the following steps:

1) If the module is a BS2, skip to step 3.
2) Send 1-byte value (0 – 7) to indicate the program slot to download into.
3) Send an 18-byte packet.
4) Discard the 18-byte echo (in receive buffer).
5) Read the acknowledge byte.
6) If acknowledge byte is not 0, error.
7) If acknowledge byte is 0 and there are more packets to send, go to step 3.
8) If all packets have been sent, execute the complete sequence or just quit.

### COMPLETE SEQUENCE

The complete sequence, if used, immediately follows the identify sequence or the program sequence.

The complete sequence is simply the transmission of a 1-byte value (0) to tell the BASIC Stamp that the session is finished. If the complete sequence is not used, the BASIC Stamp will reset after 2.3 seconds (if no serial communication is detected).

## EXAMPLE: PROGRAMMING A BASIC STAMP 2

This example demonstrates the sequence of steps required to program a BASIC Stamp 2 module with a simple PBASIC program (shown below). Even without interfacing to the tokenizer, the packet data presented here can be used to develop and test the downloading routine.

```
DEBUG "HI", CR
STOP
```

The above code, when tokenized, results in a single 18-byte programming packet containing the following data:

$FF $00 $00 $00 $00 $30 $A0 $C7 $92 $66 $48 $13 $84 $4C $35 $07 $C0 $4B

To program the BS2 (assuming we're using the proper serial port), follow the steps below:

1. RESET SEQUENCE
    a. Set DTR high.
    b. Set break condition on TX.

   c. Pause for at least 2 ms.
   d. Set DTR low.
   e. Pause for at least 36 ms.
   f. Clear break condition on TX.
   g. Pause for approximately 20 ms and flush receive buffer.
2. IDENTIFY SEQUENCE
   a. Transmit a capital “B” (ASCII 66).
   b. Receive and discard echo of “B”.
   c. Receive byte. If it is not 2’s-complement of “B” (ASCII 190), error, stop.
   d. Transmit a capital “S” (ASCII 83).
   e. Receive and discard echo of “S”.
   f. Receive byte. If it is not 2’s-complement of “S” (ASCII 173), error, stop.
   g. Transmit a “2” (ASCII 50).
   h. Receive and discard echo of “2”.
   i. Receive byte. If it is not 2’s-complement of “2” (ASCII 206), error, stop.
   j. Transmit a 0. This is the “Version Number” command.
   k. Receive and discard echo of 0.
   l. Receive byte. If it is 10 (meaning this is version 1.0 of BS2 firmware), continue, otherwise: error, stop.
3. PROGRAM SEQUENCE
   a. Send an 18-byte packet.
   b. Discard the 18-byte echo (in receive buffer).
   c. Read the acknowledge byte.
   d. If acknowledge byte is not 0, error.
   e. Since there are no more packets to send (in our example).
4. COMPLETE SEQUENCE
   a. Send a 1-byte value of 0 and then just quit.