Parallax BASIC Stamp® IIsx

The Parallax BASIC Stamp IIsx module is an extended form of the BASIC Stamp II that incorporates several key features of the Scenix Semiconductor SX microcontroller and an advanced EEPROM. Like the BASIC Stamp II, the BS2-SX module comes in a 24-pin, 600 mil-wide DIP package with 16 I/O pins and 2 dedicated serial input/output pins. The BS2-SX executes PBASIC instructions 2.5 times faster than the BS2-IC, resulting in an average speed of 10,000 PBASIC instructions per second. The BS2-SX also includes 16K of EEPROM space organized in 8 blocks of 2K bytes each. This memory organization allows for downloading up to 8 different PBASIC programs into the BS2-SX that can each share RAM and pass execution between them.

A comparison of some of the key differences between BASIC Stamp modules is shown below:

	Rev. D / BS1-IC	BS2-IC	BS2-SX
Microcontroller	Microchip PIC16C56	Microchip PIC16C57	Scenix SX28AC
Program Execution Speed	2,000 instructions/sec.	4,000 instructions/sec.	10,000 instructions/sec.
Processor Speed	4 Mhz	20 Mhz	50 Mhz
Program Memory Size	256 Bytes	2k Bytes	8 programs x 2k Bytes ea. (16k Bytes)*
RAM Size	16 Bytes (2 for I/Os and 14 for variables)	32 Bytes (6 for I/Os and 26 for variables)	32 Bytes (6 for I/Os and 26 for variables)
Scratch Pad RAM	N/A	N/A	64 Bytes (1 for program ID and 63 for user)
Number of Inputs / Outputs	8	16 + 2 dedicated serial I/O	16 + 2 dedicated serial I/O
Current @5v	1.4ma Run / 40µa Sleep	8ma Run / 100µa Sleep	60ma Run / 200µA Sleep
Source / Sink Current per I/O	20 mA / 25 mA	20 mA / 25 mA	30 mA / 30 mA
Source / Sink Current per unit	40 mA / 50 mA	40 mA / 50 mA per group of 8 I/O pins (0-7 and 8-15)**	60 mA / 60 mA per group of 8 I/O pins (0-7 and 8-15)**
Connector Socket	14 Pin Sip	24 Pin Dip	24 Pin Dip
PBASIC Commands	32	36	39
PC Programming Interface	Parallel Port	Serial Port (9600 Baud)	Serial Port (9600 Baud)
PC Software Text Editor	STAMP.EXE	STAMP2.EXE	STAMP2SX.EXE

* BS2-SX EEPROM memory can only be accessed in pages of 2K. Each of the eight programs within the BS2-SX can only access its own 2K page.

** Max current (source or sink) can only be realized though the use of an external 5-volt regulator.

Power Requirements / Setup

The BASIC Stamp IIsx requires 6-9 VDC @ 65 mA max. The BS2-SX should operate for about seven hours on a 9-volt battery, assuming the battery can deliver 450 milliamp-hours. The on-board 5-volt regulator conditions the input voltage on the Vin pin to 5-volts for the internal BS2-SX circuitry and provides this voltage on the Vdd pin. This regulator can deliver 150 mA of current at 5-volts maximum. The Vdd pin may be used to provide power to external devices as long as the overall current demand does not exceed 150 mA total, including the BS2-SX itself and any circuitry on it's I/O pins.

The BS2-SX has a pin configuration identical to that of the BS2-IC. The BASIC Stamp 2 Carrier Board and identical programming connections may be used with the BS2-SX. Note: the external power source should always be disconnected from the BASIC Stamp IISX before attempting to install or remove it from a board.

Memory Organization

The BASIC Stamp IIsx contains RAM for temporary workspace storage and EEPROM for non-volatile workspace and PBASIC program storage. RAM is organized in two groups called Variable RAM and Scratch Pad RAM.

Variable RAM consists of 16 words (32 bytes) of register space that can be organized into words, bytes, nibbles or bits, or any combination necessary. This area of RAM is usually accessed through the declaration of symbols and aliases whose sizes are specifically defined as a word, byte nibble or bit. The first 3 words (6 bytes) of Variable RAM are reserved for I/O registers and the remaining 13 words (26 bytes) are for general-purpose use. Variable RAM is identical in use to that of the BASIC Stamp II (see pages 213–224 in BASIC Stamp Programming Manual v1.9).

BASIC Stamp IIsx Variable RAM (I/O and variable space)				
Word Name	**Byte Name**	**Nibble Names**	**Bit Names**	**Special Notes**
INS	INL INH	INA, INB, INC, IND	IN0–IN7, IN8–IN15	Input pins
OUTS	OUTL OUTH	OUTA, OUTB, OUTC, OUTD	OUT0–OUT7, OUT8–OUT15	Output pins
DIRS	DIRL DIRH	DIRA, DIRB, DIRC, DIRD	DIR0–DIR7, DIR8–DIR15	I/O pin direction control
W0	B0 B1			General Purpose
W1	B2 B3			General Purpose
W2	B4 B5			General Purpose
… (continues W3 though W11 and B6 through B23) …				
W12	B24 B25			General Purpose

Note that predefined register names W0–W12 and B0–B25 are available for use, but not recommended. Preferred method of register allocation is through the use of symbol and alias names as with the BS2-IC. See pages 217–224 in BASIC Stamp Programming Manual v1.9.

ScratchPad RAM consists of 64 bytes (0-63) of register space that can only be accessed through the use of the BS2-SX's GET and PUT commands (see below). Locations 0 through 62 are available for general purpose use while location 63 is a read-only register whose value always contains the ID of the currently running program (0-7).

Both Variable and Scratch Pad RAM is cleared to zero upon power-up or reset conditions.

BASIC Stamp IIsx Scratch Pad RAM	
RAM location	**Special Notes**
0	General Purpose
1	General Purpose
62	General Purpose
63	Contains ID of currently running program (0-7)

The EEPROM included with the BASIC Stamp IIsx provides for 16 kilobytes of non-volatile storage of both programs and data. Up to eight different PBASIC programs (each of up to 2K in size) can be downloaded into the BS2-SX at one time. Each program receives its own 2K section of the EEPROM based on the program's numerical ID (0-7). Each PBASIC program operates within its own 2K section similar to a program in the BASIC Stamp II. Any space

Stamp Programming Manual v1.9).

Program 0 is the default program and is the first to run upon a power-up or reset condition. A RUN command may be used to pass execution to another program (see the RUN command below). Note that the flow of program execution is sequential in nature; only one program can be active at any time.

Organization of BS2-SX Programs	
EEPROM Locations	**Contents**
$0000 - $07FF	Program 0 (logical locations $0000 - $07FF)
$0800 - $0FFF	Program 1 (logical locations $0000 - $07FF)
$1000 - $17FF	Program 2 (logical locations $0000 - $07FF)
$1800 - $1FFF	Program 3 (logical locations $0000 - $07FF)
$2000 - $27FF	Program 4 (logical locations $0000 - $07FF)
$2800 - $2FFF	Program 5 (logical locations $0000 - $07FF)
$3000 - $37FF	Program 6 (logical locations $0000 - $07FF)
$3800 - $3FFF	Program 7 (logical locations $0000 - $07FF)

Note: A PBASIC program can only access EEPROM memory within its own 2K space. Though the program could be located in any physical chunk of EEPROM, it must always access its memory using the logical addresses of $0000 - $07FF, or decimal 0 – 2047)

Software Commands

The BASIC Stamp IIsx and its text editor recognize all the BASIC Stamp II commands plus three new commands: PUT, GET and RUN. The PUT and GET commands are used to write to and read from Scratch Pad RAM and the RUN command selects one of the eight programs to run. Of the 39 BS2-SX commands available, 3 are new, 25 are identical in function as with the BS2-IC, and 11 vary from the BS2-IC only in their timing. Details of the similarities and differences are included in the following chart. All page numbers refer to the BASIC Stamp Programming Manual v1.9.

BASIC Stamp IIsx Command Set	
Command	**Comments**
BRANCH	Identical to the BS2-IC. (See pages 247 – 248).
BUTTON	Identical to the BS2-IC. (See pages 249 – 250).
COUNT	(See pages 251 – 252). Differences are in timing only, as follows: • **Period** is a variable / constant (1 to 65535) specifying the time in units of 0.4 ms during which to count. For example, COUNT 0, 1000, I counts 0-1-0 or 1-0-1 transitions on pin 0 for 400 ms and puts the resulting count into the variable i.
DEBUG	Identical to the BS2-IC. (See pages 253 – 256).
DTMFOUT	(See pages 257 – 259). Differences are in timing only, as follows: • **Ontime** and **Offtime** parameters are optional variables or constants (0 to 65535) specifying the duration of the tone or pause in-between tones in units of 0.4 ms.
END	Identical to the BS2-IC. (See page 260).

	Identical to the BS2-IC. (See pages 261 - 263).
FREQOUT	(See pages 264 - 265). Differences are in timing and frequency, as follows: • **Duration** is a variable / constant specifying the length (1 to 65535) of the tone(s) in units of 0.4 ms. • **Freq1** and **Freq2** are variables / constants specifying the frequency (0 to 32767) in units of 2.5 Hz. Freq2 is an optional parameter. The range in frequency is 0 Hz to 81.917 KHz. For example, FREQOUT 0,10,1000 produces a signal on pin 0 for 4 ms (10 x 0.4 ms) at 2500 Hz (1000 x 2.5 Hz).
GET	New command, see below.
GOSUB	Identical to the BS2-IC. (See pages 266 - 267).
GOTO	Identical to the BS2-IC. (See page 268).
HIGH	Identical to the BS2-IC. (See page 269).
IF…THEN	Identical to the BS2-IC. (See pages 270 - 275).
INPUT	Identical to the BS2-IC. (See pages 276 - 277).
LOOKDOWN	Identical to the BS2-IC. (See pages 278 - 281).
LOOKUP	Identical to the BS2-IC. (See pages 282 - 283).
LOW	Identical to the BS2-IC. (See page 284).
NAP	Identical to the BS2-IC. (See pages 285 - 286).
OUTPUT	Identical to the BS2-IC. (See page 287).
PAUSE	Identical to the BS2-IC. (See page 288).
PULSIN	(See pages 289 - 290). Differences are in timing only, as follows: • **ResultVariable** is a variable in which the pulse duration (in 0.8 us units) will be stored. If the state of the pin doesn't change, PULSIN won't trigger. PULSIN waits a maximum of 0.0524 seconds for a trigger, then returns with 0 in ResultVariable. If the pulse is longer than 0.0524 seconds, PULSIN returns a 0 in ResultVariable.
PULSOUT	(See pages 291 - 292). Differences are in timing only, as follows: • **Time** is a variable / constant (0 - 65535) that specifies the duration of the pulse in 0.8 us units. The maximum pulse possible is 0.0524 seconds.
PUT	New command, see below.
PWM	(See pages 293 - 295). Differences are in timing only, as follows: • **Cycles** is a variable / constant (0 - 255) specifying an approximate duration (in units of 0.4 ms) of PWM output.
RANDOM	Identical to the BS2-IC. (See pages 296 - 297).
RCTIME	(See pages 298 - 301). Differences are in timing only, as follows: • **ResultVariable** is a variable in which the time measurement (0 to 65535 in 0.8 us units) will be stored. If pin remains in state for longer than 52.4 ms, RCTIME returns zero.
READ	Identical to the BS2-IC. (See pages 302 - 303).
RETURN	Identical to the BS2-IC. (See page 304).
REVERSE	Identical to the BS2-IC. (See pages 305 - 306).
RUN	New command, see below.
SERIN	(See pages 307 - 318). Differences are in timing only, as follows: Range of serial input is 305.2 baud to 115.2K baud. To calculate proper bit period value for any baud rate, use the formula: Bit_Period = INT(2,500,000 / baud rate) - 8 Use Revised Table I-6, below, for common baud modes.

SEROUT	(See pages 319 –329). Differences are in timing only, as follows: Range of serial output is 305.2 baud to 115.2K baud. To calculate proper bit period value for any baud rate, use the formula: Bit_Period = INT(2,500,000 / baud rate) - 8 Use Revised Table I-6, below, for common baud modes.
SHIFTIN	(See pages 330 –333). Differences are in timing only, as follows: Data is clocked in at approximately 42 KHz.
SHIFTOUT	(See pages 334 –335). Data is clocked out at approximately 42 KHz.
SLEEP	Identical to the BS2-IC. (See pages 336 –337).
STOP	Identical to the BS2-IC. (See page 338).
TOGGLE	Identical to the BS2-IC. (See pages 339 –340).
WRITE	Identical to the BS2-IC. (See pages 341 –343).
XOUT	Identical to the BS2-IC. (See pages 344 –346).

Table I-6 (revised for BASIC Stamp IIsx)

Common Data Rates and Corresponding Baudmode Values

Data Speed	Direct Connection		Through Line Driver	
Baud Rate	8 data bits, no parity	7 data bits, even parity	8 data bits, no parity	7 data bits, even parity
600	20542	28734	4158	12350
1200	18459	26651	2075	10267
2400	17417	25609	1033	9225
4800	16896	25088	512	8704
9600	16636	24828	252	8444
19200	16506	24698	122	8314
38400	16441	24633	57	8249

Software Interface

The BASIC Stamp IIsx text editor is similar to the BASIC Stamp II (stamp2.exe) DOS program. Each unique program (of a maximum of eight) to be downloaded to the BS2-SX must be downloaded separately with a unique program ID. Pressing ALT+# (where # is a number from 0 to 7) will change the ID (shown at the top of the window) of the currently visible source code in the editor. This ID is not saved with the program and must be set manually each time it is loaded from disk and verified before each download.

The sequence of keystrokes to load and store two programs would consist of the following:

1. ALT+0 sets editor to program 0.
2. ALT+L loads program 0 into editor.
3. ALT+R downloads program 0 in the BS2-SX's EEPROM.
4. ALT+1 sets editor to program 1.
5. ALT+L loads program 1 into editor.
6. ALT+R downloads program 1 in the BS2-SX's EEPROM.

The shortcut key ALT+R downloads only one program at a time. Note that you must load each program separately.

PUT

PUT location,value

Store value in Scratch Pad RAM location.

- ***Location*** is a variable / constant (0-62) that specifies the Scratch Pad RAM location to store a value into.
- ***Value*** is a variable / constant (0-255) to store into Scratch Pad RAM.

The BS2-SX utilizes 63 bytes of Scratch Pad RAM. Only byte-size values can be placed in this RAM. The PUT command places the value in a particular Scratch Pad RAM position. Any program running on the BASIC Stamp IIsx may use the GET command to retrieve any values in Scratch Pad RAM. *Locations* specified above 63 will wrap around to the start of Scratch Pad RAM (64 reads location 0, 65 reads location 1, etc).

Scratch Pad RAM location 63 holds the active program ID (0-7). All RAM locations are initialized to zero upon power-up and reset conditions. You could check that the BS2-SX has been reset by reading a Scratch Pad RAM location which was previously written with a non-zero value.

The sample program below places a word variable into two Scratch Pad RAM locations and retrieves the word variable in another program under another name.

```
'Program #0
rhino           var         word
head            var         rhino.highbyte
tail            var         rhino.lowbyte
x               var         byte

rhino=45678
put 2,head
put 3,tail
get 63,x                    'get program number
debug "You are in Program #",dec x,cr
debug "rhino= ",dec rhino,cr
pause 1000
run 4                       'go to program #4


'Program #4
monkey          var         word
head            var         monkey.highbyte
tail            var         monkey.lowbyte
x               var         byte

get 2,head
get 3,tail
get 63,x                    'get program number
debug "You are in Program #",dec x,cr
debug "monkey= ",dec monkey,cr
debug "rhino and monkey are = ?"
pause 1000
```

GET

GET *location,variable*

Read Scratch Pad RAM location and place value in variable.

- ***Location*** is a variable / constant (0-63) that specifies the Scratch Pad RAM location to read from.
- ***Variable*** is a variable where the value will be stored.

The GET command reads a value from the specified Scratch Pad RAM location and stores it in *variable*. Any values placed in the 63 bytes of Scratch Pad RAM can be retrieved by all programs installed on the BASIC Stamp IIsx. The demo program below places byte values into RAM locations and retrieves the byte values under a different variable name.

```
x                   var             byte
y                   var             byte
z                   var             byte
w                   var             byte

x=67
y=990
put 4,x                             'put in RAM
put 5,y
debug "x= ",dec x,cr
debug "y= ",dec y,cr

get 4,z                             'get from RAM
get 5,w
debug "z= ",dec z,cr
debug "w= ",dec w,cr
debug "z=x and w=y"
```

RUN

RUN *program*

Switches execution to another BASIC Stamp IIsx program.

- ***Program*** is a variable / constant (0-7) that specifies the program to be run.

A total of 16k bytes of code space may only be used by running up to eight 2 kbyte programs in the BASIC Stamp IIsx. The RUN command activates the specified program and stays in the newly activated program until it receives another RUN command, or until a power-down or reset condition occurs.

The I/O pins retain their current state (directions and output latches) and all Variable and Scratch Pad RAM locations retain their current data during a transition between programs with the RUN command. If sharing data between programs within Variable RAM, make sure to keep similar variable declarations (defined in the same order) in all programs so that the variables align themselves on the proper word, byte, nibble and bit boundaries across programs.

On power-up or reset events, the first program to run is program 0. Normally, a master-type program will be used in this position and will control initial execution of the other programs.

Any program number specified above 7 will wrap around and result in running one of the 8 programs (RUN 8 will run program 0, RUN 9 will run program 1, etc).

The program below uses Scratch Pad RAM location 0 to store the number of the calling program. Program 0 interprets a 0 in this RAM location as a signal that the BS2-SX has been reset (any other calling program would have set RAM 0 to a non-zero value). These two demo programs below illustrates moving from program 0 into program 6 and returning back to the starting program. Both programs are included on the disk.

```
'Program0.bs2
x           var         byte
y           var         byte
get 63,x                                    'get current program number
                                            'from RAM location 63
debug "You are in Program #",dec x,cr
get 0,y
if y=0 then powerup                         'scratch pad RAM is cleared on reset
debug "You just came from Program #", dec y,cr

continue:
pause 1000
put 0,0                                     'current program number
run 6                                       'go to Program #6

powerup:
debug "Stamp has been reset",cr
goto continue
end
.
.
.
'Program6.bs2
x                       var                 byte
get 63,x                                    ' location 63 contains current Program #
debug "You are in Program #",dec x,cr
get 0,y                                     ' get previous program #
debug "You just came from Program #", dec y,cr
pause 1000
put 0,x                                     'stores current program # in RAM
run 0                                       'return to Program #0
end
```