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 BS2SX module comes in a 24-pin, 600 mil-wide DIP package with 16 I/O pins and 2 dedicated serial input/output pins. The BS2SX executes PBASIC instructions 2.5 times faster than the BS2-IC, resulting in an average speed of 10,000 PBASIC instructions per second. The BS2SX 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 BS2SX 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	BS2SX
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	65ma 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 - or - STAMPW.EXE	STAMP2SX.EXE - or - STAMPW.EXE (v1.091+)

* BS2SX EEPROM memory can only be accessed in pages of 2K. Each program within the BS2SX 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 BS2SX 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 BS2SX 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 BS2SX itself and any circuitry on it's I/O pins.

The BS2SX 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 BS2SX. 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 BS2SX'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
… (continues 2 though 61) …	
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 BS2SX 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 within the 2K section that

is not used by the program's code may be used to store and retrieve non-volatile data with the DATA directive and the READ and WRITE commands (see pages 228 – 230, 302 – 303, and 341 – 343 in the BASIC 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 BS2SX 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 editors 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 BS2SX 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).
FOR…NEXT	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 304.5 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) – 20. Timeout value is in 0.4ms units. 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 304.5 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) – 20. Timeout value

	is in 0.4ms units. 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	20530	28722	4146	12338
1200	18447	26639	2063	10255
2400	17405	25597	1021	9213
4800	16884	25076	500	8692
9600	16624	24816	240	8432
19200	16494	24686	110	8302
38400	16429	24621	45	8237

Software Interface (Windows)

There are two editors available for the BASIC Stamp IIsx, a Windows version and a DOS version. **This section describes the Windows version (see the *Software Interface (DOS)* section for information on using the DOS version). The Windows version is recommended for most tasks.** The BASIC Stamp IIsx Windows editor (stamp2w.exe v1.091+) is the same editor that supports the BASIC Stamp II.

The BASIC Stamp Windows Interface was designed to be easy to use and mostly intuitive. Those that are familiar with the DOS-only version of the interface as well as standard Windows software should feel comfortable using the BASIC Stamp Windows Interface.

Editor Window

The Interface consists of one main editor window that can be used to view and modify up to 16 different source code files at once. Each source code file that is loaded into the editor will have its own tab at the top of the page labeled with the name of the file. Source code that has never been saved to disk will default to "Untitled#"; where # is an automatically generated number. A user can switch between source code files by simply pointing and clicking on a file's tab.

The status of the active source code page is indicated in a status bar below it and the full path to the source code (if it has been loaded from or saved to disk) will appear in the title bar of the BASIC Stamp interface. The status bar contains information such as cursor position, file save status, download status and syntax error/download messages.

After entering the desired source code in the editor window, selecting Run -> Run (or pressing Ctrl-R) will tokenize and download the code to the BASIC Stamp IIsx (assuming the code is correct).

BASIC Stamp IIsx Programs
For BASIC Stamp 2SX programs, each editor page can be a separate project, or part of a single project. Any "project" (consisting of up to eight files) can be downloaded to the BASIC Stamp 2SX module.

Because the Windows editor supports more than one model of the BASIC Stamp, it is necessary to tell the editor which model you are trying to program. The best way to do this is with the $STAMP directive, as described below.

STAMP 2SX Directive:
The $STAMP directive is a special command which can be included (usually near the top) in a program to indicate the model of BASIC Stamp targeted. For BASIC Stamp 2SX programs, the directive should look similar to:

```
' {$STAMP BS2SX}
```

Note that the directive appears on a comment line (the apostrophe (') indicates the remaining text to the right is a comment). This is for compatibility with the DOS Stamp editors. The DOS Stamp editors will ignore the line as if it were just a comment. The entire directive must be enclosed in brackets, {...} and may contain additional spaces in certain areas. For example: ' { $STAMP BS2SX } , '{$STAMP BS2SX} and '{$STAMP BS2SX } are all acceptable variations, however: '{$ STAMP BS2SX} and '{$STAMPBS2SX} are not acceptable and will be ignored.

This directive is read and acted upon by the Stamp Windows editor any time a source code file containing it is loaded, tokenized, run (downloaded) or viewed in the Memory Map.

Using Multiple BS2SX Program Slots:
For BASIC Stamp 2SX projects (those consisting of multiple programs), the $STAMP directive has an option to specify additional filenames. The syntax below demonstrates this form of the $STAMP directive:

```
' {$STAMP BS2SX, file1, file2, …, file7}
```

Use this form of the $STAMP directive if a BS2SX project, consisting of multiple files, is desired. The *file1*, *file2*, etc. items should be the actual name (and optionally the path) of the other files in the project. *file1* refers to BS2SX program number 1, *file2* is BS2SX program number 2, etc. If no path is given, the path of program 0 (the program in which the $Stamp directive is entered) is used. Up to seven filenames can be included, bringing the total to eight files in the project all together. Upon tokenizing, running or viewing program 0 in the Memory Map, the editor will read the directive, determine if the indicated files exist, will load them if necessary and change their captions to indicate the project they belong to and their associated program number. After the directive is tokenized properly, and all associated files are labeled properly, tokenizing, running or viewing any program in the Memory Map will result in that program's entire project being tokenized, downloaded or viewed.

When a file that is part of a BS2SX project is closed, the entire project (all the associated files) will be closed as well. When program #0 of a project is opened from diskette, the entire project will be loaded as well.

Source Code Tabs:
The tabs at the top of each open source code page will indicate the name of the file and, in the case of BS2SX files and projects, the program's logical number and project it belongs to. For example, if a tab displays "0:Test.bsx", it is a BS2SX file, called Test, is logical program number 0 and will be downloaded as such. If a tab displays "[Test] 1:Process.bsx", it is a BS2SX file, called Process, is logical program number 1 and belongs to a BS2SX project called Test. It will be downloaded into program slot #1 in the BS2SX immediately after 0:Test.bsx is downloaded into program slot #0.

BS2SX Download Modes:
The editor has the ability to treat BS2SX projects as one logical unit and can download each of the associated source code files to the BS2SX at once. In order to minimize download time for large projects a BS2SX Default Download Mode is available in the Preferences window. The available modes are: "Modified" (the default), "All" or "Current" and are explained below. This item only affects download operations for the BS2SX.

Download Mode	Function
Modified (default)	This mode will cause only the source code files that were modified since the last download to be downloaded next time. If no files have been modified since the last download, or the entire project has just been loaded into the editor, all the files will be downloaded next time. This mode decreases the delay during downloading BS2SX projects and should help speed development and testing.
All	This mode will cause all the source code files to be downloaded each time.
Current	This mode will cause only the current source code file to be downloaded, ignoring all the others.

Regardless of the download mode selected, the BS2SX programs will be downloaded into the program slot indicated in their tab.

Shortcut Keys (Windows Editor)

The following table lists the available keyboard shortcuts within the BASIC Stamp Windows Interface.

File Functions	
Shortcut Key	Function
Ctrl+O	Open a source code file into the Editor window.
Ctrl+S	Save current source code file to disk.
Ctrl+P	Print current source code.
Editing Functions	
Shortcut Key	Function
Ctrl+Z	Undo last action.
Ctrl+X	Cut selected text to the clipboard.
Ctrl+C	Copy selected text to the clipboard.
Ctrl+V	Paste text from clipboard to selected area.
Ctrl+A	Select all text in current source code.
Ctrl+F	Find or Replace text.
F3	Find text again.
F5	Open Preferences window.
Coding Functions	
Shortcut Key(s)	Function
F6 or Ctrl+I	Identify BASIC Stamp firmware.
F7 or Ctrl+T	Perform a syntax check on the code and display any error messages.
F8 or Ctrl+M	Open Memory Map window.
F9 or Ctrl+R	Tokenize code, download to the BASIC Stamp and open Debug window if necessary.
F11 or Ctrl+D	Open a new Debug window.
F12	Switch to next window (Editor, Debug #1, Debug #2, Debug #3 or Debug #4)
Ctrl+1, Ctrl+2, Ctrl+3, Ctrl+4	Switch to Debug Terminal #1, Debug Terminal #2, etc. if that Terminal window is open.
Ctrl+`	Switch to Editor window.
ESC	Close current window.

Software Interface (DOS)

There are two editors available for the BASIC Stamp IIsx, a Windows version and a DOS version. **This section describes the DOS version (see the *Software Interface (Windows)* section for information on using the Windows version).** The BASIC Stamp IIsx DOS editor (stamp2sx.exe) is similar to the BASIC Stamp II (stamp2.exe) DOS editor. Pressing ALT-L will allow a source code file to be loaded from disk. The DOS editor can only load up one source code file at a time. Each unique BS2SX program (of a maximum of eight) to be downloaded to the BS2SX must be downloaded separately with a unique program ID. NOTE: The Windows version does not have this limitation. 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 and verified manually each time it is loaded from disk and before each download.**

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

1. ALT+L loads program 0 into editor.
2. ALT+0 sets editor to program 0.
3. ALT+R downloads program 0 in the BS2SX's EEPROM.
4. ALT+L loads program 1 into editor.
5. ALT+1 sets editor to program 1.
6. ALT+R downloads program 1 in the BS2SX's EEPROM.

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

Shortcut Keys (DOS Editor)

The following table lists the available keyboard shortcuts within the BASIC Stamp DOS Interface.

File Functions	
Shortcut Key	Function
Alt+L	Load a source code file into the Editor window.
Alt+S	Save current source code file to disk.
Alt+Q	Close the editor.
Editing Functions	
Shortcut Key	Function
Alt+X	Cut selected text to the clipboard.
Alt+C	Copy selected text to the clipboard.
Alt+V	Paste text from clipboard to selected area.
Alt+F	Find or Replace text.
Alt+N	Find text again.
Coding Functions	
Shortcut Key(s)	Function
Alt+0..7	Set Program Slot # to download to
Alt+I	Identify BASIC Stamp firmware.
Alt+M	Open Memory Map window.
Alt+R	Tokenize code, download to the BASIC Stamp and open Debug window if necessary.

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 BS2SX 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 BS2SX 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 BS2SX 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
```