Parallax StampMem

The Parallax StampMem board is a simple data storage device that can be directly connected to a BASIC Stamp. The StampMem provides 64K of E²PROM (non-volatile storage) space organized in bytes.

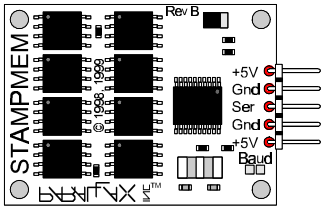
Features

- 64K (65,536) bytes of non-volatile storage (enough to hold approximately 25 pages of text)
- Fast data read/write; less than 5ms writes (per 8 bytes) and less than 1ms reads
- Simple serial connection (2400 or 9600 baud, N,8,1) for easy use with BASIC Stamp SERIN/SEROUT commands
- Three-wire connection (+5v, ground and serial signal). +5v can be supplied from BASIC Stamp's Vdd pin.
- Receives up to 64 bytes of data at a time for writing

Connection and Power Requirements

The StampMem requires a three-wire connection (+5v, ground and serial I/O) to operate. The five-pin header on the StampMem board provides these connections in a symmetric form (with serial I/O in the center) to allow for a reversible connector (see figure 1, below). Power required is 5 volts DC @ 10ma max. ***Always disconnect the external power source from the StampMem before attempting to connect or disconnect from it***.

Figure 1. StampMem board



To connect to a BASIC Stamp, simply connect three wires according to Table 1, below. *Note that it is not necessary to connect to both +5v pins, and both Gnd pins on the StampMem as each pair are electrically equivelant. Two of each are provided only for allowing the use of a non-keyed (reversible) 5-pin connector.*

Table 1. StampMem to BASIC Stamp Connections

StampMem	BASIC Stamp 1 (Rev. D or BS1-IC)	BASIC Stamp 2 or 2SX (BS2-IC or BS2SX-IC)
+5v	+5v (Vdd)	+5v (Vdd)
Gnd	Gnd (Vss)	Gnd (Vss)
Ser	Any I/O pin (P0 – P7)	Any I/O pin (P0 – P15)

Note: The StampMem defaults to a serial communication speed of 9600 baud (no parity, 8 data bits, 1 stop bit). To configure it for 2400 baud (N,8,1) communication (as is needed for connecting to a BASIC Stamp 1), the solder pads marked "baud" must be connected, or shorted, together. Use a soldering iron and a small amount of solder to bridge the gap between the two "baud" solder pads for 2400 baud communication. All other connections should remain the same.

Commands

To control the StampMem, commands and data must be sent serially (by using the BASIC Stamp's SERIN and SEROUT commands). There are two main commands built into the StampMem; Read and Write. Both the Read and Write commands have the same, 4-byte syntax, as follows:

CommandValue, AddrHigh, AddrLow, NumBytes

CommandValue is the value (0 or 1) representing the desired command. A 0 in *CommandValue* means "write to memory" and a 1 means "read from memory". *AddrHigh* and *AddrLow* are the high-byte and low-byte of the starting memory address on which to operate. Together, *AddrHigh* and *AddrLow* form a word value (two bytes) representing memory locations 0 - 65,535. *NumBytes* is the number of bytes (1-64) to read from or write to memory. A maximum of 64 bytes can be read from or written to memory during a single operation. The StampMem commands are summarized in Table 2, below.

Table 2. StampMem Commands

Command	Value	Description	Syntax
Write	0	Write up to 64 bytes to StampMem.	0, AddrHigh, AddrLow, NumBytes
Read	1	Read up to 64 bytes from StampMem.	1, AddrHigh, AddrLow, NumBytes

Writing

To write data to the StampMem, the Write command must be sent (telling the StampMem you wish to write to memory, the address of the starting location to write to, and the number of bytes to write) and then the data must be sent afterwards. This can be done with a single SEROUT command or with multiple SEROUT commands. The following BASIC Stamp 2 code will write the values 10, 30, 150 and 255 to locations 5 though 8:

```
SEROUT 0, 84, [0, 0, 5, 4]          'Send Write command, starting address (5), and number of bytes (4)
SEROUT 0, 84, [10, 30, 150, 255]    'Send Data to be written (4 bytes total)
```

The above code may also be written as just one SEROUT command as follows:

```
SEROUT 0, 84, [0, 0, 5, 4, 10, 30, 150, 255]
```

NOTE: For single-byte writes, the StampMem requires a maximum of 5 ms to complete the operation. For multi-byte writes, the StampMem requires a maximum of 5 ms per group of 8 bytes. For example, when writing one byte at a time, it is necessary to insert a PAUSE 5 command to allow enough time to complete the last write before reading or writing more data. When writing 64 bytes (8 groups of 8 bytes) is it best to insert a PAUSE 40 command to allow enough time to complete the operation before reading or writing more data.

Reading

To read data from the StampMem, the Read command must be sent (telling the StampMem you wish to read from memory, the address of the starting location to read from, and the number of bytes to read) and then the data must be read in afterwards. This must be done with a SEROUT (to send the Read command) followed by a SERIN (to read the data). The following BASIC Stamp 2 code will read the values stored in locations 5 though 8:

```
Value     VAR       BYTE(4)

SEROUT 0, 84, [1, 0, 5, 4]          'Send Read command, starting address (5), and number of bytes (4)
SERIN 0, 84, [STR Value\4]          'Read Data into Value array (a 4-byte array)
```

STPMEM.BS2

The program below writes 16 characters (bytes) to the StampMem starting at location 1000. It then reads them back out and displays them on the DEBUG screen.

```
'Example of StampMem data storage I/O.
'I/O pin 0 should be connected to StampMem's SER pin.
'2/22/99

x       var     byte(17)        'data string
Addr    var     word            'memory location to operate on

PAUSE 500                       'powerup delay
'Write 16 bytes to StampMem
Addr = 1000                                         'start at address 1000
SEROUT 0,84,[0,Addr.HIGHBYTE,Addr.LOWBYTE,16]       'send Write command
SEROUT 0,84,["0123456789abcdef"]                    'send data

PAUSE 10                        'allow time to write data [5ms*2(groups of 8 bytes)]
Addr = 1000                                         'start at address 1000
SEROUT 0,84,[1,Addr.HIGHBYTE,Addr.LOWBYTE,16]       'semd Read command
SERIN 0,84,[STR x\16]                               'receive data
DEBUG STR x,CR                                      'read data on PC screen
```

STPREAD.BS2

The program below reads all the data stored in the StampMem, starting at location 0, and displays it on the DEBUG screen.

```
'This BS2-IC program reads all data stored on StampMEM board.
'I/O pin 0 should be connected to StampMem's SER pin.
'2/22/99

x       VAR     BYTE(16)        'data string
Addr    VAR     WORD            'memory address

PAUSE 500                       'powerup delay

Loop:
  SEROUT 0,84,[1,Addr.HIGHBYTE,Addr.LOWBYTE,16]     'transmit data parameters
  SERIN 0,84,[STR x\16]                             'receive read data
  DEBUG STR x                   'read data on PC screen
  PAUSE 200
Addr =Addr + 16
IF Addr <> 0 THEN Loop

DEBUG "FINISHED READING ALL EEPROM DATA"
```

STPRDPC.BS2

The program works with the QBASIC program stpin.exe to read data from a PC and store it into the StampMem.

```
'Program reads data from PC *.txt file and places data onto StampMEM board
'PC must be running stpin.exe file to activate data input
'After programming BASIC Stamp2 diconnect pin no. 3 to allow data flow
'from the PC when using BASIC Stamp2 programming serial port (p16)

x       var     byte(17)        'data string
n       var     byte            'number data string characters to read
d1      var     byte            'data I/O, 1 = input & 0 = output
d2      var     byte            'high byte location, 0 to 255
d3      var     byte            'low byte location, 0 to 255
d4      var     byte            'number of data bytes

pause 500                       'powerup delay
d1=0:d2=0:d3=0:d4=16:n=16       'indicate data parameters to output
serin 16,84,[wait("START")]     'wait for start command from PC
aa:
serin 16,84,2000,cc,[str x\n] 'receive data from PC
serout 0,84,[d1,d2,d3,d4]       'write data to stamp memory board
serout 0,84,[str x\n]
d3=d3+n                         'get another n data bytes
if d3<n then bb
goto aa
bb:
d2=d2+1
if d2=0 then cc
goto aa
cc:
stop
```

STPVER.BS2

```
'This BS2-IC program reads and displays the StampMem's version number

Version         VAR     BYTE

CheckVersionNumber:
  SEROUT 0,84,["ZERN"]          'Get Version Number of StampMem
  SERIN 0,84,1000,NoVersion,[Version]
  DEBUG "StampMem is version ",DEC1 Version DIG 1, ".", DEC Version DIG 0
STOP

NoVersion:
  'StampMem did not respond
  DEBUG "ERROR! StampMem did not respond!"
STOP
```

STPCLEAR.BS2

```
'This BS2-IC program tells the StampMem to clear all memory.
'(writes FF hex data in each byte location. Requires about 1 minute)

Idx           VAR    BYTE          'General Purpose Index Variable
Value         VAR    BYTE(2)       'Used to read values

  'Send Clear command, (StampMem clears it's own memory (to $FF))
  DEBUG "Clearing memory",CR,"Time Remaining:     Sec",REP 3\4
  SEROUT 0,84,["ZERC"]
  Idx = 60
WaitMore:
  Idx = Idx - 10
  IF Idx = 0 THEN NoClearResponse
  DEBUG REP 3\3,DEC3 Idx
  SERIN 0,84,10000,WaitMore,[STR Value\2]
  IF Value(0) = 255 THEN OK
  DEBUG "ERROR! StampMem failed to Clear all memory."
  DEBUG CR,"EEPROM #",DEC Value(1)/32+1, " Failed."
  STOP
OK:
  DEBUG CR,"StampMem cleared all memory."
STOP

NoClearResponse:
  DEBUG "ERROR! StampMem did not respond to Clear command"
STOP
```