The BASIC Stamp 2
The BASIC Stamp 2

**Serial Signal Conditioning**
Conditions voltage signals between PC serial connection (± 12V) and BASIC Stamp (5V)

**EEPROM**
Stores the tokenized PBASIC program.

**Interpretor Chip**
Reads the BASIC program from the EEPROM and executes the instructions.

**5V Regulator**
Regulates voltage to 5V with a supply of 5.5VDC to 15VDC
BASIC Stamp 2 (BS2)

The BASIC Stamp 2 (BS2) is an embedded system that includes:

1. Microcontroller (PIC16C57) – Brains of the system, provides BASIC interpreter, serial communication, and I/O.

2. Memory (EEPROM) – User program storage and long term data storage.

3. Voltage Regulator – Generates 5Vdc from supply power of 5.5Vdc to 15Vdc.


5. Miscellaneous Support Components – 4v brown out detector, transistors, and resistors.
BASIC Stamp 2 microcontroller

PIC16C57

- 8-bit CMOS Microcontroller
- RISC CPU (only 33 single word, single cycle instructions)
- 28 pin, DIP
- 12 bit wide instructions
- 8 bit wide data
- 8 bit clock counter
- 72 bytes of RAM
- 1 watch dog timer

16 I/O pins + 2 dedicated serial
2K EEPROM/ROM

See http://www.microchip.com
# BASIC Stamp Pin Descriptions

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOUT</td>
<td>Serial Out: connects to PC serial port RX pin (DB9 pin 2 / DB25 pin 3) for programming.</td>
</tr>
<tr>
<td>2</td>
<td>SIN</td>
<td>Serial In: connects to PC serial port TX pin (DB9 pin 3 / DB25 pin 2) for programming.</td>
</tr>
<tr>
<td>3</td>
<td>ATN</td>
<td>Attention: connects to PC serial port DTR pin (DB9 pin 4 / DB25 pin 20) for programming.</td>
</tr>
<tr>
<td>4</td>
<td>VSS</td>
<td>System ground: (same as pin 23) connects to PC serial port GND pin (DB9 pin 5 / DB25 pin 7) for programming.</td>
</tr>
<tr>
<td>5-20</td>
<td>P0-P15</td>
<td>General-purpose I/O pins: each can sink 25 mA and source 20 mA. However, the total of all pins should not exceed 50 mA (sink) and 40 mA (source) if using the internal 5-volt regulator. The total per 8-pin groups (P0 – P7 or P8 – 15) should not exceed 50 mA (sink) and 40 mA (source) if using an external 5-volt regulator.</td>
</tr>
<tr>
<td>21</td>
<td>VDD</td>
<td>5-volt DC input/output: if an unregulated voltage is applied to the VIN pin, then this pin will output 5 volts. If no voltage is applied to the VIN pin, then a regulated voltage between 4.5V and 5.5V should be applied to this pin.</td>
</tr>
<tr>
<td>22</td>
<td>RES</td>
<td>Reset input/output: goes low when power supply is less than approximately 4.2 volts, causing the BASIC Stamp to reset. Can be driven low to force a reset. This pin is internally pulled high and may be left disconnected if not needed. Do not drive high.</td>
</tr>
<tr>
<td>23</td>
<td>VSS</td>
<td>System ground: (same as pin 4) connects to power supply’s ground (GND) terminal.</td>
</tr>
<tr>
<td>24</td>
<td>VIN</td>
<td>Unregulated power in: accepts 5.5 - 15 VDC (6-40 VDC on BS2-IC rev. e), which is then internally regulated to 5 volts. May be left unconnected if 5 volts is applied to the VDD (+5V) pin.</td>
</tr>
</tbody>
</table>
BASIC Stamp 2 microcontroller - 2

PIC16C57
16 I/O pins + 2 dedicated serial
2K EEPROM/ROM
72 bytes of RAM on PIC
  • Only 26 bytes available
20 MHz clock, 4000 instructions per second
20 mA current (source) limit at each pin (25 mA sink limit)

Caution!

PIC16F84
  - Discussed extensively in book
  - F = flash memory (a type of EEPROM)
BASIC Stamp 2 Memory

Memory (space for code)
- 2K Bytes
- 500 lines of code

Speed
- 20 MHz
- 4000 instructions/second

RAM (space for storing variables)
- 26 Bytes
- REG0 – REG12 are 16-bit registers
  - 208 1-bit variables (Bits)
  - 52 4-bit variables (Nibbles)
  - 26 8-bit variables (Bytes)
  - 13 16-bit variables (Words)

+6 bytes for storing I/O

Program
- Stored in EEPROM memory
BASIC Stamp 2 Proto Board
### Stamp Specifications (revised 09/02)

<table>
<thead>
<tr>
<th>Released Products</th>
<th>BS2i-IC</th>
<th>BS2i2-IC</th>
<th>BS2i2x-IC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Package</strong></td>
<td>24-pin DIP</td>
<td>24-pin DIP</td>
<td>24-pin DIP</td>
</tr>
<tr>
<td><strong>Package Size (L x W x H)</strong></td>
<td>2.5” x 0.5” x 0.5”</td>
<td>2.5” x 0.5” x 0.5”</td>
<td>2.5” x 0.5” x 0.5”</td>
</tr>
<tr>
<td><strong>Environment</strong></td>
<td>0° - 70°C (32° - 158°F) **</td>
<td>0° - 70°C (32° - 158°F) **</td>
<td>0° - 70°C (32° - 158°F) **</td>
</tr>
<tr>
<td><strong>Microcontroller</strong></td>
<td>Microchip PIC16C546</td>
<td>Microchip PIC16C576</td>
<td>Scenix SX28AC</td>
</tr>
<tr>
<td><strong>Processor Speed</strong></td>
<td>4 MHz</td>
<td>20 MHz</td>
<td>20 MHz</td>
</tr>
<tr>
<td><strong>Program Execution Speed</strong></td>
<td>~2,000 instructions/sec</td>
<td>~4,000 instructions/sec</td>
<td>~10,000 instructions/sec</td>
</tr>
<tr>
<td><strong>RAM Size</strong></td>
<td>16 Bytes (2 I/O, 14 Variable)</td>
<td>32 Bytes (6 I/O, 26 Variable)</td>
<td>64 Bytes</td>
</tr>
<tr>
<td><strong>Scratch Pad RAM</strong></td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>EEPROM (Program) Size</strong></td>
<td>256 Bytes, ~80 instructions</td>
<td>256 Bytes, ~50 instructions</td>
<td>8 x 2K Bytes, ~4,000 inst.</td>
</tr>
<tr>
<td><strong>Number of I/O pins</strong></td>
<td>8</td>
<td>16 + 2 Dedicated Serial</td>
<td>16 + 2 Dedicated Serial</td>
</tr>
<tr>
<td><strong>Voltage Requirements</strong></td>
<td>5 - 15 vdc</td>
<td>5 - 15 vdc</td>
<td>5 - 12 vdc</td>
</tr>
<tr>
<td><strong>Current Draw @ 5V</strong></td>
<td>2mA Run / 20mA Sleep</td>
<td>8 mA Run / 100 uA Sleep</td>
<td>20 mA Run / 100 uA Sleep</td>
</tr>
<tr>
<td><strong>Source / Sink Current per I/O</strong></td>
<td>20 mA / 25 mA</td>
<td>30 mA / 30 mA</td>
<td>150 mA / 150 mA</td>
</tr>
<tr>
<td><strong>DOS Text Editor</strong></td>
<td>StampExE</td>
<td>StampExE</td>
<td>StampExE</td>
</tr>
<tr>
<td><strong>Windows Text Editor</strong></td>
<td>N/A</td>
<td>StampwExE (v1.04 and up)</td>
<td>StampwExE (v1.06 and up)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Released Products</th>
<th>BS2p24-IC</th>
<th>BS2p24-IC</th>
<th>Javelin Stamp</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Package</strong></td>
<td>24-pin DIP</td>
<td>24-pin DIP</td>
<td>24-pin DIP</td>
</tr>
<tr>
<td><strong>Package Size (L x W x H)</strong></td>
<td>1.2” x 0.5” x 0.4”</td>
<td>1.2” x 0.5” x 0.4”</td>
<td>1.2” x 0.5” x 0.4”</td>
</tr>
<tr>
<td><strong>Environment</strong></td>
<td>0° - 70°C (32° - 158°F) **</td>
<td>0° - 70°C (32° - 158°F) **</td>
<td>0° - 70°C (32° - 158°F) **</td>
</tr>
<tr>
<td><strong>Microcontroller</strong></td>
<td>Scenix SX484AC</td>
<td>Ubicom SX484AC</td>
<td>Javelin Stamp IDE</td>
</tr>
<tr>
<td><strong>Processor Speed</strong></td>
<td>20 MHz Turbo</td>
<td>20 MHz Turbo</td>
<td>20 MHz Turbo</td>
</tr>
<tr>
<td><strong>Program Execution Speed</strong></td>
<td>-12,000 instructions/sec</td>
<td>8,000 instructions/sec</td>
<td>25 MHz Turbo</td>
</tr>
<tr>
<td><strong>RAM Size</strong></td>
<td>38 Bytes (12 I/O, 26 Variable)</td>
<td>38 Bytes (12 I/O, 26 Variable)</td>
<td>-5,500 instructions/sec</td>
</tr>
<tr>
<td><strong>Scratch Pad RAM</strong></td>
<td>128 Bytes</td>
<td>128 Bytes</td>
<td>32K Bytes</td>
</tr>
<tr>
<td><strong>EEPROM (Program) Size</strong></td>
<td>8 x 2K Bytes, ~4,000 inst.</td>
<td>8 x 2K Bytes, ~4,000 inst.</td>
<td>32K Bytes</td>
</tr>
<tr>
<td><strong>Number of I/O pins</strong></td>
<td>16 + 2 Dedicated Serial</td>
<td>16 + 2 Dedicated Serial</td>
<td>16</td>
</tr>
<tr>
<td><strong>Voltage Requirements</strong></td>
<td>5 - 12 vdc</td>
<td>5 - 12 vdc</td>
<td>5 - 12 vdc</td>
</tr>
<tr>
<td><strong>Current Draw @ 5V</strong></td>
<td>40 mA Run / 400 uA Sleep</td>
<td>40 mA Run / 400 uA Sleep</td>
<td>6 vdc (reg), 6 - 24 vdc (unreg)</td>
</tr>
<tr>
<td><strong>Source / Sink Current per I/O</strong></td>
<td>30 mA / 30 mA</td>
<td>30 mA / 30 mA</td>
<td>60 mA Run / 13uA Sleep</td>
</tr>
<tr>
<td><strong>DOS Text Editor</strong></td>
<td>StampExE</td>
<td>StampExE</td>
<td>StampExE</td>
</tr>
<tr>
<td><strong>Windows Text Editor</strong></td>
<td>N/A</td>
<td>StampwExE (v1.1 and up)</td>
<td>StampwExE (v1.3 and up)</td>
</tr>
</tbody>
</table>

*Industrial Models Available - Environment is 40° - 65°C (-40° - 185°F)**

**70% Non-Condensing Humidity**
IMPORTANT: PIN CHARACTERISTICS

- **Vss (System ground)** – Any external circuitry connected to the BS2 must have a common ground with the BS2.

- **P0-P15 (general purpose I/O)** – Maximum current limits for I/O pins must be obeyed.

- **Vdd (5Vdc supply)** – If user has 5Vdc available, connect it to this pin.

- **Vin (5.5Vdc-15Vdc supply)** – If user has 5.5-15Vdc, connect it to this pin. Note that the onboard regulator will step this down to 5Vdc for use by the BS2 circuitry. MAKE SURE THIS 5Vdc IS NOT USED BY EXTERNAL CIRCUITRY OR DAMAGE TO THE BS2 WILL OCCUR.
Precautions

• Static sensitive device, use grounded wrist strap or touch a grounded surface before handling the BS2.

• Verify power is off before connecting and disconnecting the BS2 and any external circuitry.

• Verify BS2 orientation before inserting it into the carrier board.

• Do not overdrive I/O pins, abide by maximum limits.

• If using an unregulated supply (5.5-15Vdc), do not drive external circuitry with the regulated 5Vdc generated by the BS2.
• Communicate with the BS2 via the PC serial port (RS232).

• Download program from PC to BS2, cannot upload a program that is already in the BS2.

• Can upload data from the BS2 to the PC using the DEBUG command.
Memory Configuration

RAM:
- storage of program variables
- loses contents at power down
- 32 bytes available
  6 – I/O pin status
  26 – general purpose use

EEPROM:
- 2K bytes, ~500 instructions
- program storage
- long term data storage
- retains data at power down
Variable Definition

Mouse  VAR  BIT  'variable can be 0 thru 1
Cat    VAR  NIB  'variable can be 0 thru 15
Dog    VAR  BYTE 'variable can be 0 thru 255
Rhino  VAR  WORD 'variable can be 0 thru 65535

Horse  VAR  Rhino.BIT9 'bit 9 of Rhino
Cow    VAR  Rhino.HIGHBYTE 'highest 8 bit of Rhino

Constant Definition

Cheers  CON  5
Number Representation

1 Decimal

$1A6 Hex

%1011 Binary

Examples:

LetterA CON “A” ‘ASCII code for A (65)
Cheers CON 3 ‘decimal
Hex128 CON $80 ‘hex
FewBits CON %1101 ‘binary
Mathematical Expressions

- Addition and Subtraction, order is not important
  
  \[ 12+7-3+22=38 \]
  \[ 22-3+12+7=38 \]

- Multiplication and Division, order is important
  
  \[ 12+3\times2/4=7 \]
  \[ 2\times12/4+3=9 \]

- Note that the BS2 performs integer math only, as shown in **
  
  \[ 30/4=7 \text{ not } 7.5 \]

- BS2 solves equations in the order they are written, left to right

- Use parenthesis to change order of evaluation
  
  \[ 12+((3\times2)/4)=13 \]
http://www.parallax.com

http://groups.yahoo.com/group/basicstamps/

University of Pennsylvania
# BASIC Stamp 2 Commands

## BRANCHING
- **IF...THEN**
  - Compare and conditionally branch.
- **BRANCH**
  - Branch to address specified by offset.
- **GOTO**
  - Branch to address.
- **GOSUB**
  - Branch to subroutine at address.
- **RETURN**
  - Return from subroutine.

## LOOPING
- **FOR...NEXT**
  - Establish a FOR-NEXT loop.

## EEPROM ACCESS
- **DATA**
  - Store data in EEPROM before downloading PBASIC program.
- **READ**
  - Read EEPROM byte into variable.
- **WRITE**
  - Write byte into EEPROM.

## NUMERICS
- **LOOKUP**
  - Lookup data specified by offset and store in variable. This instruction provides a means to make a lookup table.
- **LOOKDOWN**
  - Find target’s match number (0-N) and store in variable.
- **RANDOM**
  - Generate a pseudo-random number.
BASIC Stamp 2 Commands

**DIGITAL I/O**
- **INPUT** Make pin an input.
- **OUTPUT** Make pin an output.
- **REVERSE** Reverse direction of a pin. If pin is an output, make it an input. If pin is an input, make it an output.
- **LOW** Make pin output low.
- **HIGH** Make pin output high.
- **TOGGLE** Make pin an output and toggle state.
- **PULSIN** Measure an input pulse.
- **PULSCOUT** Output a timed pulse by inverting a pin for some time.
- **BUTTON** Debounce button, perform auto-repeat, and branch to address if button is in target state.
- **COUNT** Count cycles on a pin for a given amount of time.
- **XOUT** Generate X-10 power line control codes. For use with TW523 or TW513 power line interface module.

**ASYNCHRONOUS SERIAL I/O**
- **SERIN** Input data in an asynchronous serial stream.
- **SEROUT** Output data in an asynchronous serial stream.

**SYNCHRONOUS SERIAL I/O**
- **SHIFTIN** Shift data in from synchronous serial device.
- **SHIFTOUT** Shift data out to synchronous serial device.
BASIC Stamp 2 Commands

**ANALOG I/O**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWM</td>
<td>Output PWM, then return pin to input. This can be used to output analog voltages (0-5V) using a capacitor and resistor.</td>
</tr>
<tr>
<td>RCTIME</td>
<td>Measure an RC charge/discharge time. Can be used to measure potentiometers.</td>
</tr>
</tbody>
</table>

**TIME**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAUSE</td>
<td>Pause execution for 0–65535 milliseconds.</td>
</tr>
</tbody>
</table>

**SOUND**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREQOUT</td>
<td>Generate one or two sine waves of specified frequencies.</td>
</tr>
<tr>
<td>DTMFOUT</td>
<td>Generate DTMF telephone tones.</td>
</tr>
</tbody>
</table>

**POWER CONTROL**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAP</td>
<td>Nap for a short period. Power consumption is reduced.</td>
</tr>
<tr>
<td>SLEEP</td>
<td>Sleep for 1-65535 seconds. Power consumption is reduced.</td>
</tr>
<tr>
<td>END</td>
<td>Sleep until the power cycles or the PC connects. Power consumption is reduced.</td>
</tr>
</tbody>
</table>

**PROGRAM DEBUGGING**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEBUG</td>
<td>Send information to the PC for viewing.</td>
</tr>
</tbody>
</table>
PULSIN

PULSIN Pin, State, Variable

Function
Measure the width of a pulse on Pin described by State and store the result in Variable.

- **Pin** is a variable/constant/expression (0 – 15) that specifies the I/O pin to use. This pin will be set to input mode.

- **State** is a variable/constant/expression (0 – 1) that specifies whether the pulse to be measured is low (0) or high (1). A low pulse begins with a 1-to-0 transition and a high pulse begins with a 0-to-1 transition.

- **Variable** is a variable (usually a word) in which the measured pulse duration will be stored. The unit of time for Variable is described in Table 5.62.

  units in Variable – 2 microseconds
  maximum pulse width – 131.07 milliseconds
**PULSOUT**

**PULSOUT Pin, Period**

**Function**
Generate a pulse on *Pin* with a width of *Period*.

- **Pin** is a variable/constant/expression (0 – 15) that specifies the I/O pin to use. This pin will be set to output mode.
- **Period** is a variable/constant/expression (0 – 65535) that specifies the duration of the pulse.

- units in Period – 2 microseconds
- maximum pulse width – 131.07 milliseconds
BUTTON

BUTTON Pin, DownState, Delay, Rate, Workspace, TargetState, Address

Function
Debounce button input, perform auto-repeat, and branch to address if button is in target state. Button circuits may be active-low or active-high.

- **Pin** is a variable/constant/expression (0–15) that specifies the I/O pin to use. This pin will be set to input mode.

- **DownState** is a variable/constant/expression (0 or 1) that specifies which logical state occurs when the button is pressed.

- **Delay** is a variable/constant/expression (0 – 255) that specifies how long the button must be pressed before auto-repeat starts. The delay is measured in cycles of the Button routine. Delay has two special settings: 0 and 255. If Delay is 0, Button performs no debounce or auto-repeat. If Delay is 255, Button performs debounce, but no auto-repeat.

- **Rate** is a variable/constant/expression (0 – 255) that specifies the number of cycles between auto-repeats. The rate is expressed in cycles of the BUTTON routine.

- **Workspace** is a byte variable used by BUTTON for workspace. It must be cleared to 0 before being used by BUTTON for the first time and should not be adjusted outside of the BUTTON command. NOTE: All RAM is cleared to 0 by default upon power-up or reset of the BASIC Stamp.

- **TargetState** is a variable/constant/expression (0 or 1) that specifies which state the button should be in for a branch to occur. (0=not pressed, 1=pressed)

- **Address** is a label that specifies where to branch if the button is in the target state.
SHIFTIN

SHIFTIN Dpin, Cpin, Mode, [ Variable {Bits} {, Variable {Bits}…} ]

Function
Shift data in from a synchronous serial device.

- **Dpin** is a variable/constant/expression (0 – 15) that specifies the I/O pin that will be connected to the synchronous serial device’s data output. This pin will be set to input mode.

- **Cpin** is a variable/constant/expression (0 – 15) that specifies the I/O pin that will be connected to the synchronous serial device’s clock input. This pin will be set to output mode.

- **Mode** is a variable/constant/expression (0 – 3), or one of four predefined symbols, that tells SHIFTIN the order in which data bits are to be arranged and the relationship of clock pulses to valid data. See Table 5.90 for value and symbol definitions.

- **Variable** is a variable in which incoming data bits will be stored.

- **Bits** is an optional variable/constant/expression (1 – 16) specifying how many bits are to be input by SHIFTIN. If no **Bits** entry is given, SHIFTIN defaults to 8 bits.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSBPRE</td>
<td>0</td>
<td>Data is msb-first; sample bits before clock pulse</td>
</tr>
<tr>
<td>LSBPRE</td>
<td>1</td>
<td>Data is lsb-first; sample bits before clock pulse</td>
</tr>
<tr>
<td>MSBPOST</td>
<td>2</td>
<td>Data is msb-first; sample bits after clock pulse</td>
</tr>
<tr>
<td>LSBPOST</td>
<td>3</td>
<td>Data is lsb-first; sample bits after clock pulse</td>
</tr>
</tbody>
</table>

(Msb is most-significant bit; the highest or leftmost bit of a nibble, byte, or word. Lsb is the least-significant bit; the lowest or rightmost bit of a nibble, byte, or word.)
SHIFTOUT
SHIFTOUT Dpin, Cpin, Mode, [OutputData {Bits}, OutputData {Bits}…]

Function
Shift data out to a synchronous serial device.

• **Dpin** is a variable/constant/expression (0 – 15) that specifies the I/O pin that will be connected to the synchronous serial device’s data input. This pin will be set to output mode.

• **Cpin** is a variable/constant/expression (0 – 15) that specifies the I/O pin that will be connected to the synchronous serial device’s clock input. This pin will be set to output mode.

• **Mode** is a variable/constant/expression (0 – 1), or one of two predefined symbols, that tells SHIFTOUT the order in which data bits are to be arranged.

• **OutputData** is a variable/constant/expression containing the data to be sent.

• **Bits** is an optional variable/constant/expression (1 – 16) specifying how many bits are to be output by SHIFTOUT. If no Bits entry is given, SHIFTOUT defaults to 8 bits.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSBFIRST</td>
<td>0</td>
<td>Data is shifted out lsb-first</td>
</tr>
<tr>
<td>MSBFIRST</td>
<td>1</td>
<td>Data is shifted out msb-first</td>
</tr>
</tbody>
</table>

(Msb is most-significant bit; the highest or leftmost bit of a nibble, byte, or word. Lsb is the least-significant bit; the lowest or rightmost bit of a nibble, byte, or word.)
PWM

Function
Convert a digital value to analog output via pulse-width modulation.
- **Pin** is a variable/constant/expression (0 – 15) that specifies the I/O pin to use. This pin will be set to output mode initially then set to input mode when the command finishes.
- **Duty** is a variable/constant/expression (0 - 255) that specifies the analog output level (0 to 5V).
- **Cycles** is a variable/constant/expression (0 - 255) that specifies the duration of the PWM signal.

units in Cycle – 1millisecond
Average voltage – avg. volt.=(Duty/255)*5
Required charge time – 4*R*C
RCTIME

RCTIME Pin, State, Variable

Function
Measure time while Pin remains in State, usually to measure the charge/discharge time of resistor/capacitor (RC) circuit.

- **Pin** is a variable/constant/expression (0 – 15) that specifies the I/O pin to use. This pin will be placed into input mode.

- **State** is a variable/constant/expression (0 - 1) that specifies the desired state to measure. Once Pin is not in State, the command ends and stores the result in Variable.

- **Variable** is a variable (usually a word) in which the time measurement will be stored.

---

![Diagram](image)

a) use with state = 1 (preferred - see text)

b) use with state = 0
DEBUG

DEBUG OutputData (, OutputData)

Function
Display information on the PC screen within the BASIC Stamp editor program. This command can be used to display text or numbers in various formats on the PC screen in order to follow program flow (called debugging) or as part of the functionality of the BASIC Stamp application.

- **OutputData** is a variable/constant/expression (0 – 65535) that specifies the information to output. Valid data can be ASCII characters (text strings and control characters), decimal numbers (0 - 65535), hexadecimal numbers ($0000 - $FFFF) or binary numbers (up to %1111111111111111). Data can be modified with special formatters as explained below.

<table>
<thead>
<tr>
<th>Formatter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>Displays &quot;symbol = x&quot; + carriage return; where x is a number. Default format is decimal, but may be combined with number formatters below (ex: bin ? x to display &quot;x = binary_number&quot;).</td>
</tr>
<tr>
<td>ASC ?</td>
<td>Displays &quot;symbol = x&quot; + carriage return; where x is an ASCII character.</td>
</tr>
<tr>
<td>DEC[1..5]</td>
<td>Decimal text, optionally fixed for 1 to 5 digits.</td>
</tr>
<tr>
<td>SDEC[1..5]</td>
<td>Signed decimal text, optionally fixed for 1 to 5 digits.</td>
</tr>
<tr>
<td>HEX[1..4]</td>
<td>Hexadecimal text, optionally fixed for 1 to 4 digits.</td>
</tr>
<tr>
<td>SHEX[1..4]</td>
<td>Signed hex text, optionally fixed for 1 to 4 digits.</td>
</tr>
<tr>
<td>IHEX[1..4]</td>
<td>Indicated hex text ($ prefix; ex.: $7A3), optionally fixed for 1 to 4 digits.</td>
</tr>
<tr>
<td>ISHEX[1..4]</td>
<td>Indicated, signed hex text, optionally fixed for 1 to 4 digits.</td>
</tr>
<tr>
<td>BIN[1..16]</td>
<td>Binary text, optionally fixed for 1 to 16 digits.</td>
</tr>
<tr>
<td>SBIN[1..16]</td>
<td>Signed binary text, optionally fixed for 1 to 16 digits.</td>
</tr>
<tr>
<td>IBIN[1..16]</td>
<td>Indicated binary text (% prefix; ex.: %1001), optionally fixed for 1 to 16 digits.</td>
</tr>
<tr>
<td>ISBIN[1..16]</td>
<td>Indicated, signed binary text, optionally fixed for 1 to 16 digits.</td>
</tr>
<tr>
<td>STR bytarray</td>
<td>ASCII string from bytarray until byte ≠ 0.</td>
</tr>
<tr>
<td>STR bytarray\n</td>
<td>ASCII string consisting of n bytes from bytarray.</td>
</tr>
<tr>
<td>REP byte\n</td>
<td>Display ASCII character n times.</td>
</tr>
</tbody>
</table>
Connect the active-low circuit shown in Figure 5.1 to pin P0 of the 832. When you press the button, the Debug screen will display an asterisk [*]. Feel free to modify the program to see the effects of your changes on the way BUTTON responds.

(((STAMP 832) 'STAMP directive [specifies a 832]

@Some VAR Byte 1 Backspace for BUTTON instruction.

Loop:
1 Try changing the delay value [255] in BUTTON to see the effect of
2 Use number 0 to simulate 1-255 varying delays before auto-repeat.
3 255 no auto-repeat (one action per button press).
4 BUTTON 0,255,256,MarkB,0,GoPress
5 Go to GoPress unless P0 = 0.
6 DESCR " "

@Goto:
7 GOTO LOOP
This program illustrates the use of BCTIME as a fast stopwatch. The program energizes a relay coil, then measures how long it takes for the relay contacts to close. Figure 5.13 shows the circuit. Note that BCTIME doesn’t start timing instantly.

*(BCTIME 0.0)  \*BCTIME directive [specifies a BSS]*

Declare VARS Word

Again:

02 GET  \*Energize relay coil.
03 BCTIME 0.1, RESULT  \*Readout time to contact closure.
04 DEBUG “Time to close: “, X, DEC PRINT, CR
05 HICK  \*Release the relay.
06 PLOT 1000  \*Wait a second.
07 GOTO Again  \*Do it again.
* This program uses the SHIFITIN instruction to interface with the ADC0801 8-bit analog-to-digital converter from National Semiconductor.

* The SHIFITIN command does most of the work. The mode argument in the SHIFITIN command specifies end or line-first and whether to sample data before or after the clock. In this case, we chose end-first, post-clock. The ADC0801 provides its data output with a dummy bit, which we take care of by specifying 9 bits of data instead of 8.

* Actuate the ADC081.

* Shift in the data.

* Deactivate ADC081.

* Show us the conversion result.

* Wait a second.

* Do it again.