
# A
# MOE University
## Course

**MOE U courses are made available for the benefit the FIRST robotics universe by:**

## The Miracle Workerz, FIRST Team 365

**First State Robotics, Inc.**
**Wilmington, Delaware**
**www.moe365.org**

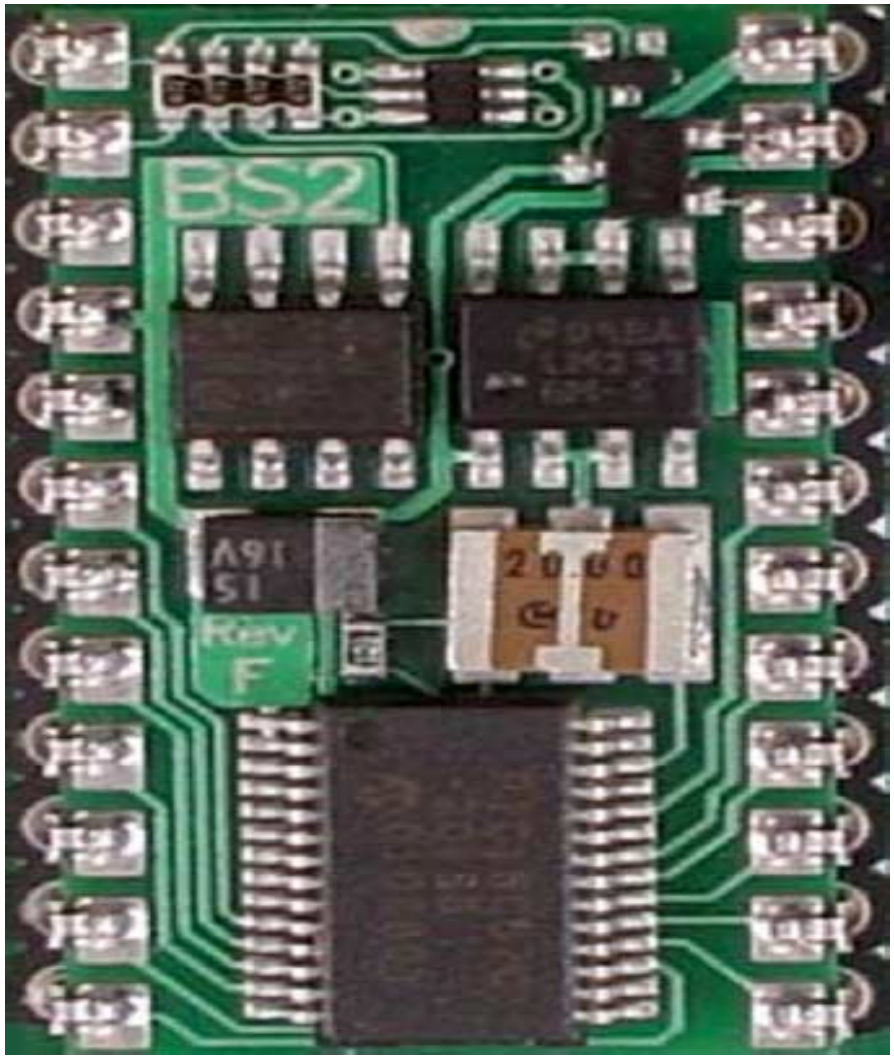
# Basics of the Stamp Processor and the Programming Language

**Shiping Zhang & Devon**

**MOE University**

Oct. 20, 2003

# BASIC Stamp



# BASIC Stamp 2 Pins

Pin 1: $S_{OUT}$

Transmits serial data during programming and using the DEBUG instruction

Pin 2: $S_{IN}$

Receives serial data during programming

Pin 3: ATN

Uses the serial DTR line to gain the Stamps attention for programming.

Pin 4. $V_{SS}$

Communications Ground (0V).

Pins 5-20:

Input/Output (I/O) pins P0 through P15

Pin 24. $V_{IN}$

Un-regulated input voltage (5.5-15V)

Pin 23. $V_{SS}$

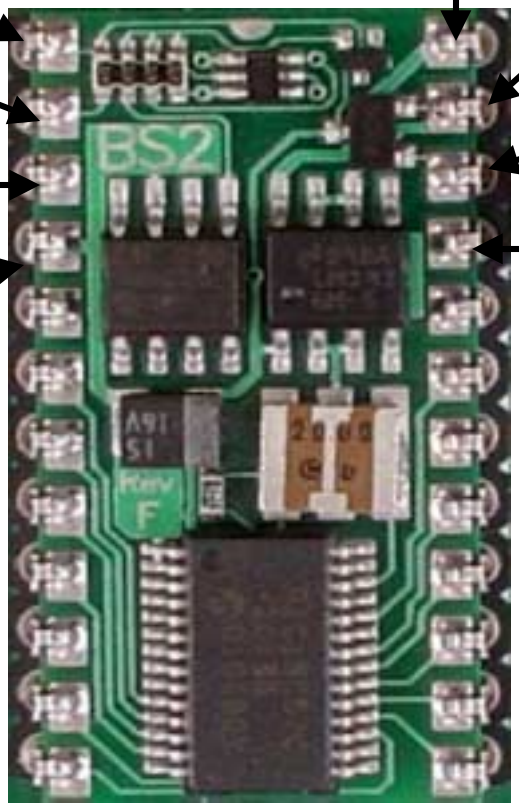
Ground (0V)

Pin 22. RES

Reset- LOW to reset

Pin 21. $V_{DD}$

Regulated 5V.




P0 P1 P2 P3 P4 P5 P6 P7

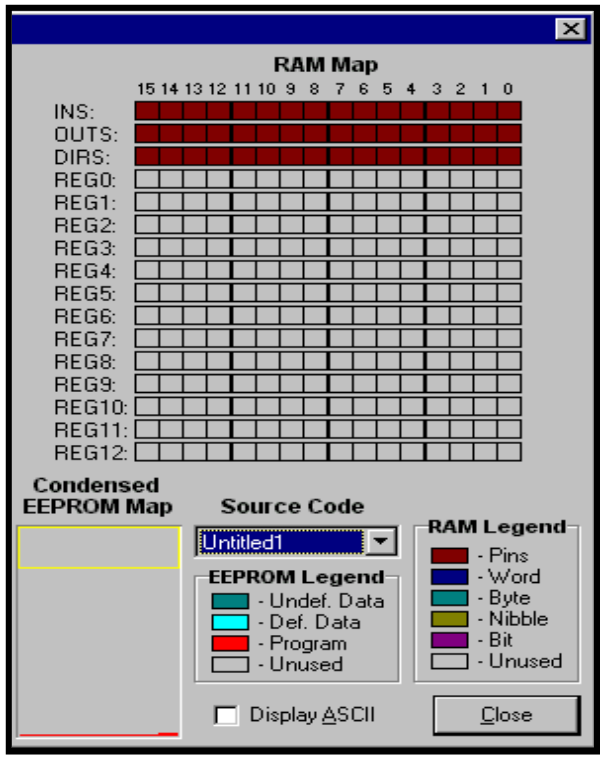
P15 P14 P13 P12 P11 P10 P9 P8

# RAM Memory

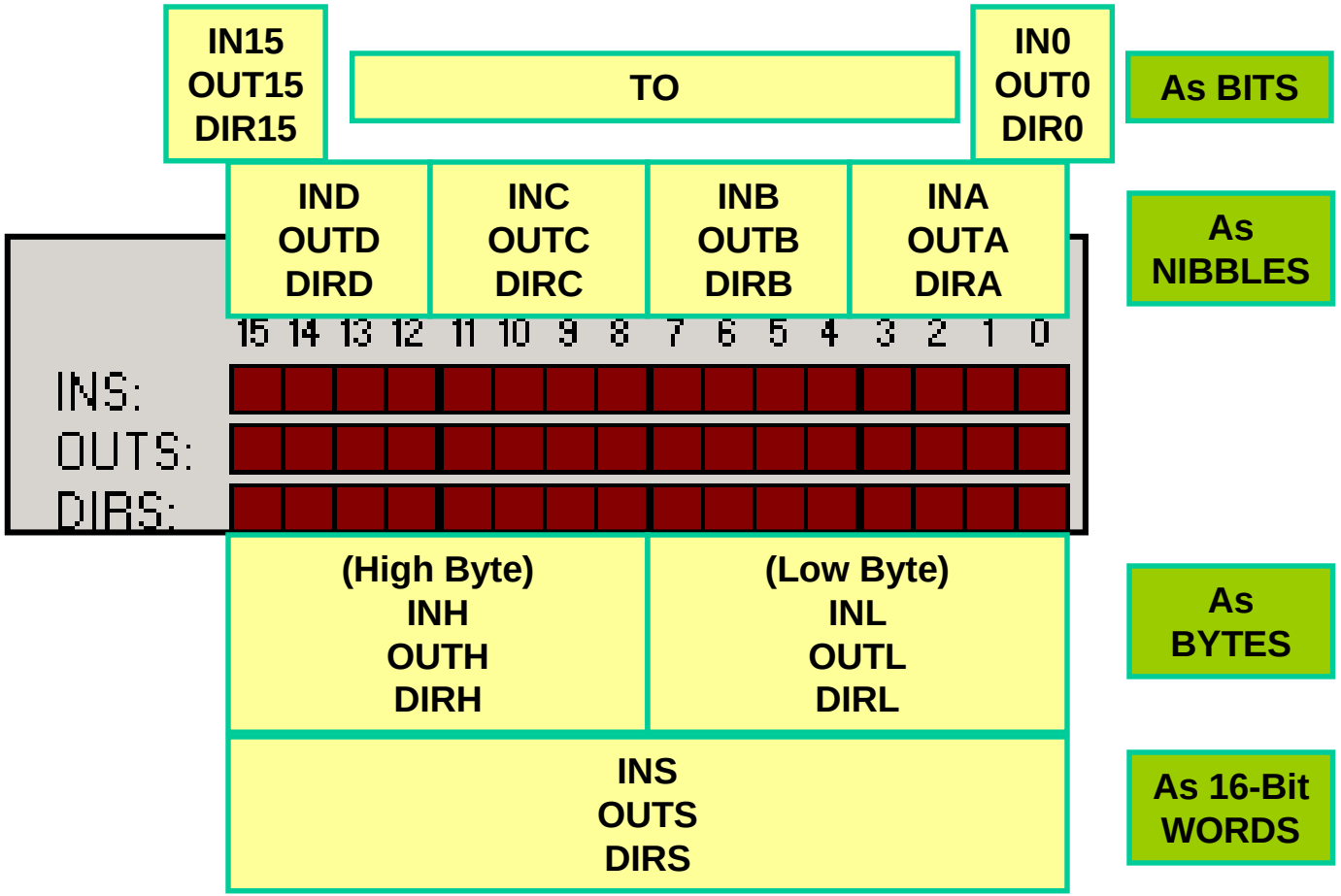
- The code space is 2K bytes (2048 bytes) in size and fills from the bottom up.
- INS, OUTS and DIRS are the registers (RAM locations) which hold the status of the I/O pins.
- REG0 – REG12 are 16-bit registers (word sized) used for general variable storage.
- The variable registers may hold:
  - 13 16-bit variables (Words)
  - 26 8-bit variables (Bytes)
  - 52 4-bit variables (Nibbles)
  - 208 1-bit variables (Bits)

  **OR**

  - Any combination of the above within memory size constraints.



- The I/O can also be addressed as nibbles, bytes or the entire word.



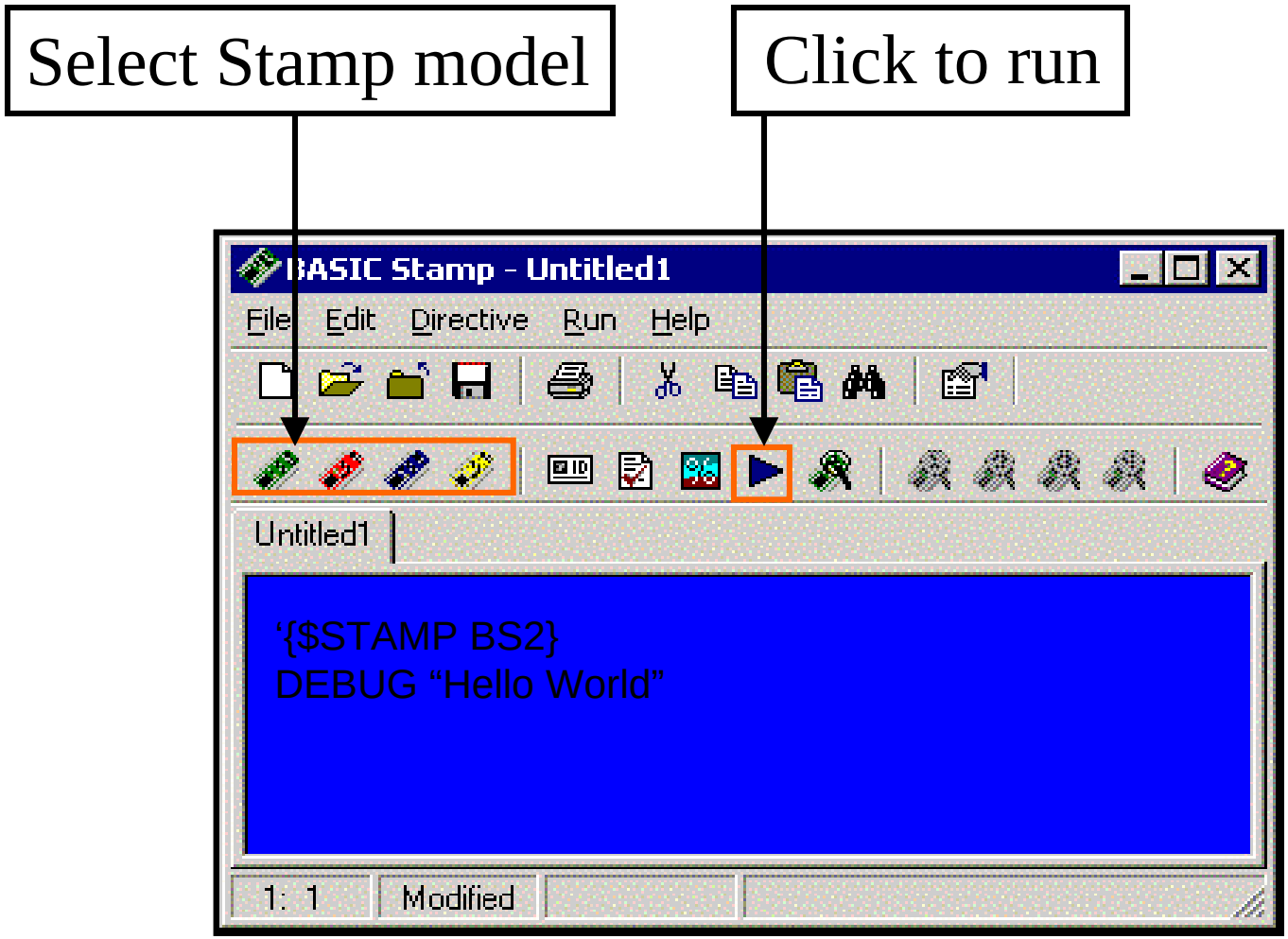
# Programming Languages

- basic
- fortran
- C/C++
- Perl
- Java

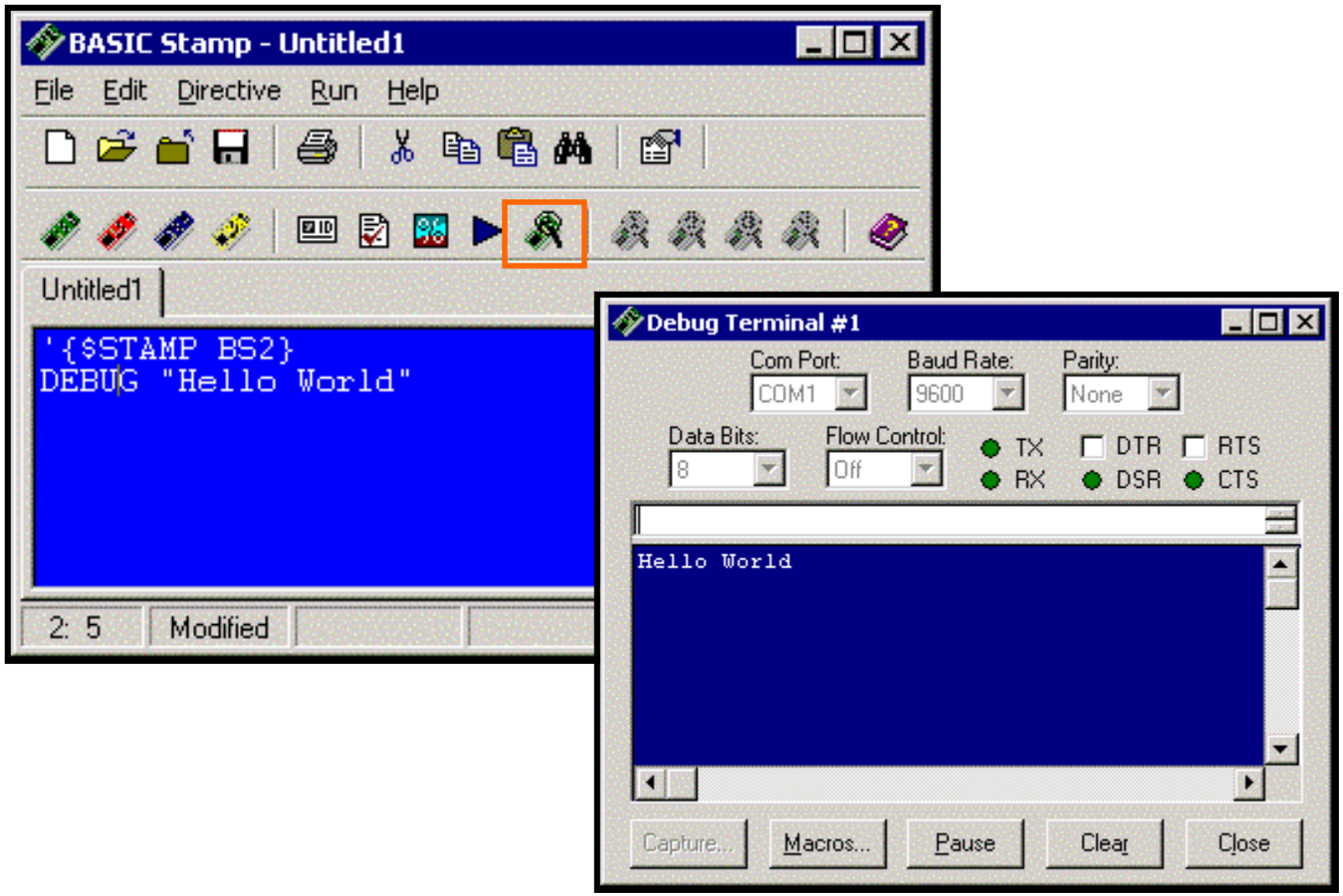
# PBASIC

- Simple
- Easy
- Reach instructions
  - common
  - specialized

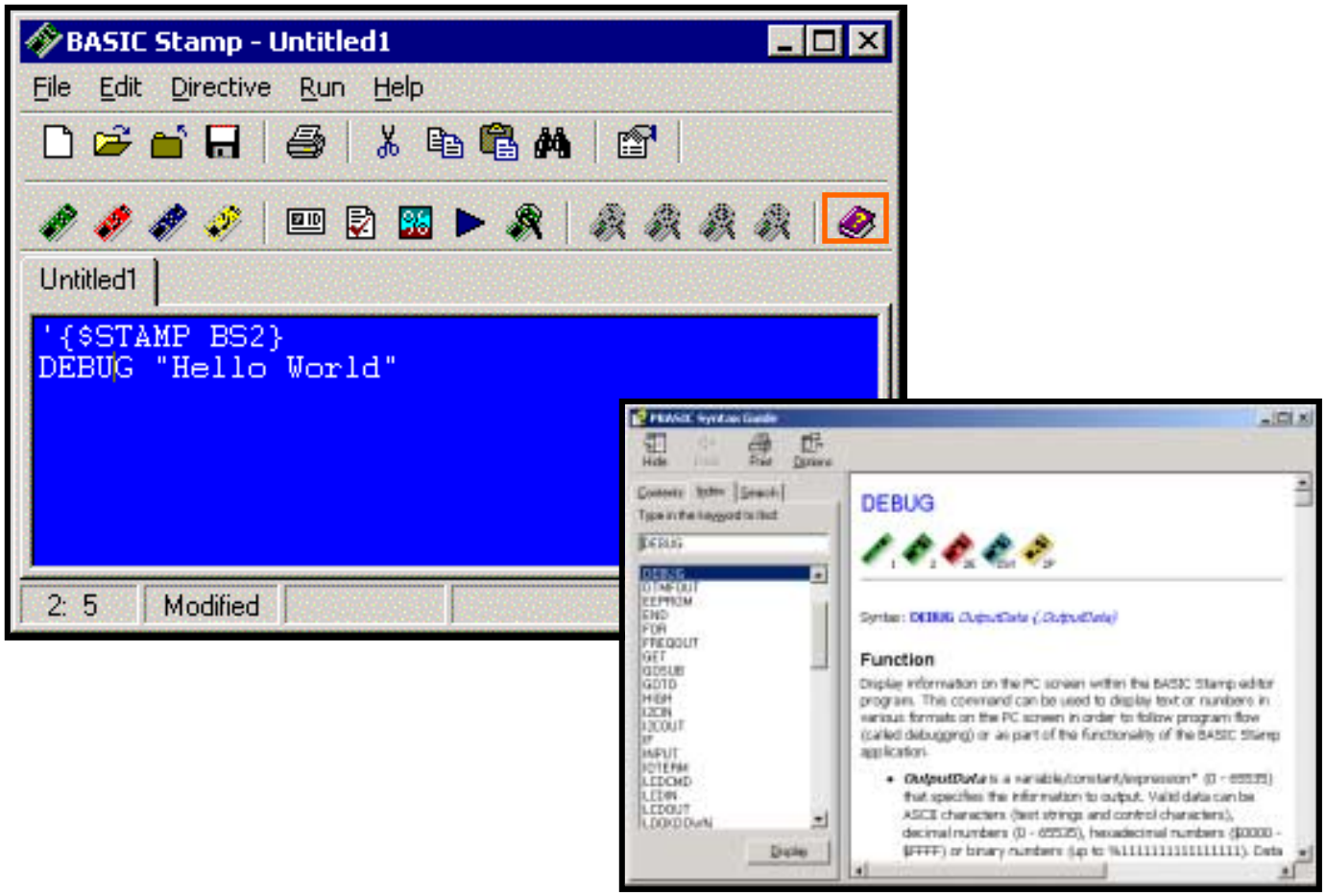
# BASIC Stamp Editor



# DEBUG Window



# Help Files


BASIC Stamp - Untitled1
File Edit Directive Run Help
Untitled1
'{$STAMP BS2}
DEBUG "Hello World"
2: 5
Modified
DEBUG
Function

# Instruction Syntax Convention

- BASIC Stamp instructions follow a common code convention for parameters (parts) of instructions.

- Take for example the FREQOUT instructions, which may be used to generate tones from a speaker:
  FREQOUT Pin, Period, Freq1 {, Freq2}
  - The instruction requires that the Pin, Period, and Freq1 is supplied and that each are separated by commas.
  - Optionally, the user MAY provide Freq2 indicated by braces { }.

- While PBASIC is NOT case-sensitive, the common convention is to capitalize instructions, and use 1st letter upper-case for all other code.

# Rules for Variable Names

- Variables cannot contain special characters such as !,@,$ except for an underscore, _.
- Variables may contain numbers but cannot start with a number.
- Variable names cannot be a PBASIC instruction.
- Declare all variables at the top of your code and comment their use.
- Size the variable appropriate to its use conserving memory whenever possible.

# Example Variable Names

**Examples of legal variable names:**

```
x             VAR BYTE    'General use variable
PressCount    VAR WORD    'Holds number of times
Pot_Value     VAR WORD    'Value of Pot
Switch1       VAR BIT     'Value of switch 1
```

**Examples of illegal variable names:**

```
My Count      Space in name
1Switch       Starts with a value
Stop!         Invalid name character
Count         PBASIC instruction
```

# Variable Modifiers

| Symbol | Definition |
|---|---|
| LOWBYTE | low byte of a word |
| HIGHBYTE | high byte of a word |
| BYTE0 | byte 0 of a word |
| BYTE1 | byte 1 of a word |
| LOWNIB | low nibble |
| HIGHNIB | high nibble |
| NIB0 - NIB3 | individual nibbles |
| LOWBIT | low bit |
| HIGHBIT | high bit |
| BIT0 - BIT15 | individual bits |

Examples:

```
Robot   VAR  WORD
Wheels  VAR  Robot.HIGHNIB    'bits 12-15
Arms    VAR  Robot.NIB0       'bits 0-3
```

# Variables

| Basic unit | Types |
| --- | --- |
| • bit - 1 bit<br>• nibble - 4 bits<br>• byte - 8 bits<br>• word - machine dependent | • char - 8 bits<br>• integer - 16 bits<br>• long integer – 32 bits<br>• long long - 64 bits<br>• float - 32 bits<br>• double float - 64 bits<br>• long double - 128 bits |

# Number Representations

| Types | Examples | Decimal Values |
|---|---|---|
| Bin (base 2) | 101101011 | 363 |
| Oct (base 8) | o15021 | 6673 |
| Hex (base 16) | 0x8A02F5 | 9044725 |

# Number Conversion Table

| Binary | Hex | Decimal |
|---|---|---|
| 0000 | 0 | 0 |
| 0001 | 1 | 1 |
| 0010 | 2 | 2 |
| 0011 | 3 | 3 |
| 0100 | 4 | 4 |
| 0101 | 5 | 5 |
| 0110 | 6 | 6 |
| 0111 | 7 | 7 |
| 1000 | 8 | 8 |
| 1001 | 9 | 9 |
| 1010 | A | 10 |
| 1011 | B | 11 |
| 0100 | C | 12 |
| 1101 | D | 13 |
| 1110 | E | 14 |
| 1111 | F | 15 |

# ASCII Chart (first 128 characters)

| Dec | Hex | Char | Name / Function |
|---|---|---|---|
| 0 | 00 | NUL | Null |
| 1 | 01 | SOH | Start Of Heading |
| 2 | 02 | STX | Start Of Text |
| 3 | 03 | ETX | End Of Text |
| 4 | 04 | EOT | End Of Transmit |
| 5 | 05 | ENQ | Enquiry |
| 6 | 06 | ACK | Acknowledge |
| 7 | 07 | BEL | Bell |
| 8 | 08 | BS | Backspace |
| 9 | 09 | HT | Horizontal Tab |
| 10 | 0A | LF | Line Feed |
| 11 | 0B | VT | Vertical Tab |
| 12 | 0C | FF | Form Feed |
| 13 | 0D | CR | Carriage Return |
| 14 | 0E | SO | Shift Out |
| 15 | 0F | SI | Shift In |
| 16 | 10 | DLE | Data Line Escape |
| 17 | 11 | DC1 | Device Control 1 |
| 18 | 12 | DC2 | Device Control 2 |
| 19 | 13 | DC3 | Device Control 3 |
| 20 | 14 | DC4 | Device Control 4 |
| 21 | 15 | NAK | Non Acknowledge |
| 22 | 16 | SYN | Synchronous Idle |
| 23 | 17 | ETB | End Transmit Block |
| 24 | 18 | CAN | Cancel |
| 25 | 19 | EM | End Of Medium |
| 26 | 1A | SUB | Substitute |
| 27 | 1B | ESC | Escape |
| 28 | 1C | FS | File Separator |
| 29 | 1D | GS | Group Separator |
| 30 | 1E | RS | Record Separator |
| 31 | 1F | US | Unit Separator |

| Dec | Hex | Char |
|---|---|---|
| 32 | 20 | space |
| 33 | 21 | ! |
| 34 | 22 | " |
| 35 | 23 | # |
| 36 | 24 | $ |
| 37 | 25 | % |
| 38 | 26 | & |
| 39 | 27 | ' |
| 40 | 28 | ( |
| 41 | 29 | ) |
| 42 | 2A | * |
| 43 | 2B | + |
| 44 | 2C | , |
| 45 | 2D | - |
| 46 | 2E | . |
| 47 | 2F | / |
| 48 | 30 | 0 |
| 49 | 31 | 1 |
| 50 | 32 | 2 |
| 51 | 33 | 3 |
| 52 | 34 | 4 |
| 53 | 35 | 5 |
| 54 | 36 | 6 |
| 55 | 37 | 7 |
| 56 | 38 | 8 |
| 57 | 39 | 9 |
| 58 | 3A | : |
| 59 | 3B | ; |
| 60 | 3C | < |
| 61 | 3D | = |
| 62 | 3E | > |
| 63 | 3F | ? |

| Dec | Hex | Char |
|---|---|---|
| 64 | 40 | @ |
| 65 | 41 | A |
| 66 | 42 | B |
| 67 | 43 | C |
| 68 | 44 | D |
| 69 | 45 | E |
| 70 | 46 | F |
| 71 | 47 | G |
| 72 | 48 | H |
| 73 | 49 | I |
| 74 | 4A | J |
| 75 | 4B | K |
| 76 | 4C | L |
| 77 | 4D | M |
| 78 | 4E | N |
| 79 | 4F | O |
| 80 | 50 | P |
| 81 | 51 | Q |
| 82 | 52 | R |
| 83 | 53 | S |
| 84 | 54 | T |
| 85 | 55 | U |
| 86 | 56 | V |
| 87 | 57 | W |
| 88 | 58 | X |
| 89 | 59 | Y |
| 90 | 5A | Z |
| 91 | 5B | [ |
| 92 | 5C | \ |
| 93 | 5D | ] |
| 94 | 5E | ^ |
| 95 | 5F | _ |

| Dec | Hex | Char |
|---|---|---|
| 96 | 60 | ` |
| 97 | 61 | a |
| 98 | 62 | b |
| 99 | 63 | c |
| 100 | 64 | d |
| 101 | 65 | e |
| 102 | 66 | f |
| 103 | 67 | g |
| 104 | 68 | h |
| 105 | 69 | i |
| 106 | 6A | j |
| 107 | 6B | k |
| 108 | 6C | l |
| 109 | 6D | m |
| 110 | 6E | n |
| 111 | 6F | o |
| 112 | 70 | p |
| 113 | 71 | q |
| 114 | 72 | r |
| 115 | 73 | s |
| 116 | 74 | t |
| 117 | 75 | u |
| 118 | 76 | v |
| 119 | 77 | w |
| 120 | 78 | x |
| 121 | 79 | y |
| 122 | 7A | z |
| 123 | 7B | { |
| 124 | 7C | \| |
| 125 | 7D | } |
| 126 | 7E | ~ |
| 127 | 7F | delete |

# Data Types

| | |
|---|---|
| Primitive | char, integer, float, etc. |
| Array | vector indexed with numbers |
| Hash | vector indexed with keys |
| Class | complex/combined |

# PBASIC Variable Types

| Name | Size | Values | Value Range |
|---|---|---|---|
| BIT | 1 bit | $2^1$ | 0 or 1 |
| NIB | 4 bits | $2^4$ | 0 - 15 |
| BYTE | 8 bits | $2^8$ | 0 - 255 |
| WORD | 16 bits | $2^{16}$ | 0 - 65535 |

# Binary Operators (not complete)

| Symbol | Description |
|---|---|
| + | Add |
| - | Subtract |
| * | Multiply |
| / | Divide |
| << | Shift left |
| >> | Shift right |
| & | Logical AND |
| \| | Logical OR |
| ^ | Logical XOR |

# Unary Operators (not complete)

| Symbol | Description |
|---|---|
| ABS | Returns absolute value |
| COS | Returns consine |
| ~ | Inverse |
| - | Negative |
| SIN | Returns sine |
| SQR | Returns square root |

# Order of Math Operation

- The BASIC Stamp solves math equationsfrom left to right.
  The steps of computing 12 + 3 * 2 / 4:

  12 + 3 = 15
  15 * 2 = 30
  30 / 4 = 7

- The BASIC Stamp only performs integer math.
  30 / 4 results 7, not 7.5. Be careful with the order.

  3 / 2 * 10 = 10 *(not 15!)*
  10 * 3 / 2 = 15

- Use parentheses to show intention

  (12 + 3) * 2 / 4 *(clear to others what you intend)*
  12 + (3 * 2 / 4)

# Stamp I/O (Input/Output)

- 16 I/O pins on the BS2x labeled P0 to P15.

  These are the pins through which input and output devices may be connected.

- Each pin may act as an input from a device, or as an output to a device.

  Depend on program codes.

# BASIC Stamp I/O

- Serial Input/Output: connect to PC
  - Loading program
  - Debugging
- Pins 0-15: Sense/Set voltage
  - High (5V)
  - Low (0V)

# Flow Control

**Branching**

| | |
|---|---|
| IF…THEN | *Compare and conditional branch* |
| GOTO | *Branch to an address* |
| GOSUB | *Branch to a subroutine* |
| RETURN | *Return from a subroutine* |

**Looping**

| | |
|---|---|
| FOR…NEXT | *Setup a loop* |

**Memory access**

| | |
|---|---|
| READ | *Read a byte from memory* |
| WRITE | *Write a byte to memory* |

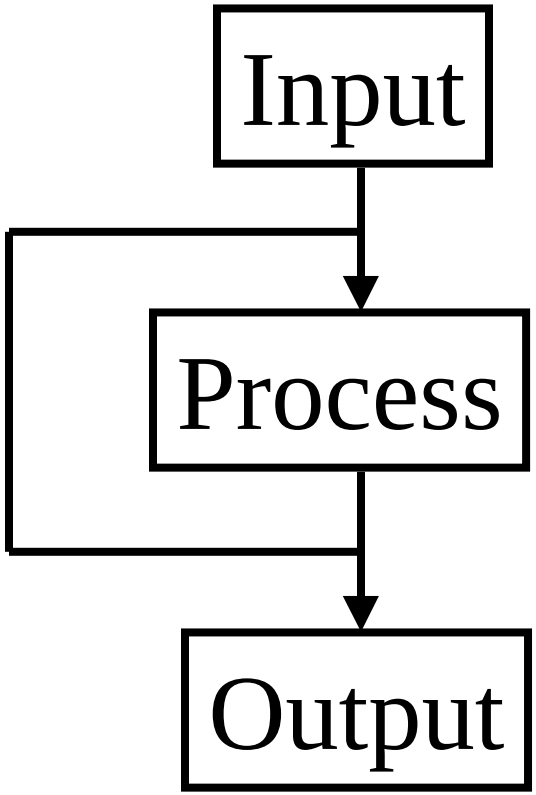
# Instructions for Pin Control

- HIGH defines the pin to be an output and sets it to a HIGH state, digital 1 or 5V.
  - HIGH *pin* (pin takes a value between 0-15, e.g. HIGH 8)
- LOW defines the pin to be an output and sets it to a LOW state, digital 0 or 0V.
  - LOW *pin* (pin takes a value between 0-15, e.g. LOW 8)
- INPUT sets the specified pin to input mode.
  - INPUT *pin* (pin takes a value between 0-15, e.g. INPUT 10)

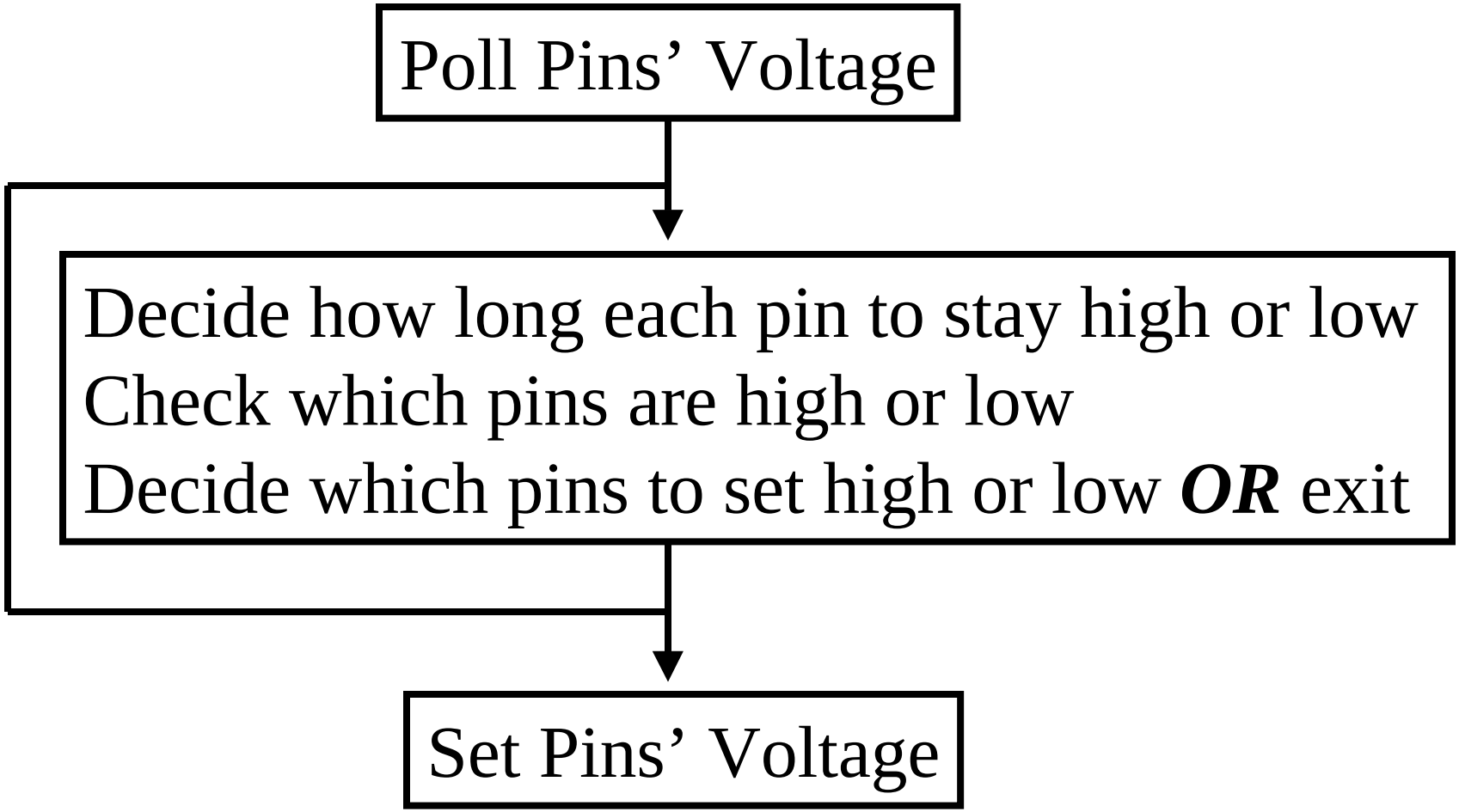
# Program Execution

- Data input
  from files, mouse, keyboard, joystick, etc.
- Data processing
  signal manipulation (math calculation, etc.)
- Data output
  to screen, files, printers, motors, etc.

# Execution Flow



# Stamp Execution Flow



# BASIC Stamp Directive

- Stamps come with several different models
  1, 2, 3e, 2sx, 2p, etc.

- Must specify model type (via one of three methods):
  - Directive: ‘{$STAMP BS2sx, prog2.bsx}
  - File extension: prog1.bsx
  - Predefined default

- Assume program contain the directive
  ‘{$STAMP BS2sx} ‘indicates to use the BASIC tamp 2sx

# A Simple Program

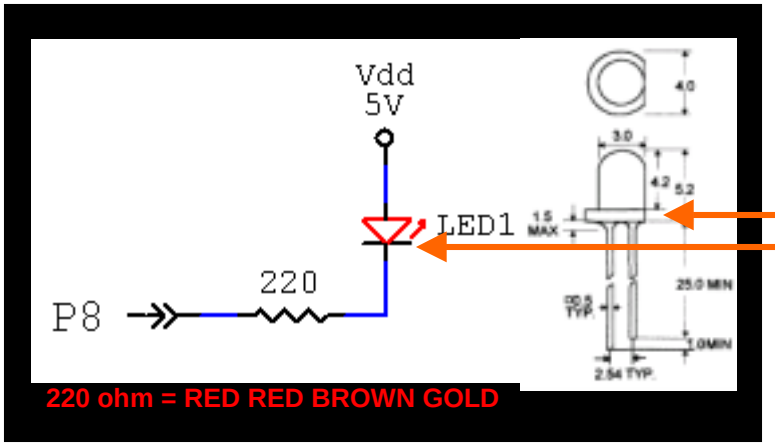
```
‘ a simple demo program
‘ loop through 10 elements of an array (vector)
‘===  declare variables
index VAR NIB                ‘ 4 bits, maximum value 15
vector VAR WORD(10)   ‘array data
‘=== first assign a value to each element
FOR index = 0 TO 9
   vector(index) = index
NEXT
‘=== then print the value of each element
FOR index = 0 TO 9
   DEBUG ? vector(index)  ‘ print to screen
NEXT
```

# A Simple Program (Perl version)

```
#!/usr/local/bin/perl
# ===  not necessary to declare variables
# my ($index, @vector);
# === first assign a value to each element
for $index (0 .. 9)
 {
   $vector[$index] = $index;
}
# === then print the value of each element
for($index = 0; $index < 10; $index++)
{
   print  $vector[$index], “\n”;  #   print to screen
}
```

# Output - Connecting an LED

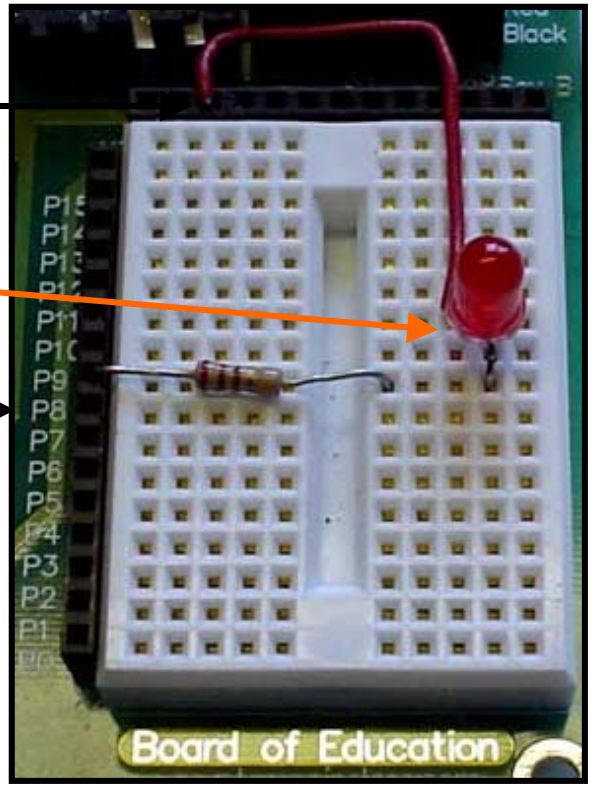
- Connect an LED to P8 as shown:




**Vdd, NOT Vin.**

**Note cathode: the 'flat side' of LED**

**Connected on P8. Angle of shot makes it appear to be on P9.**




**An LED is a diode, meaning electrons can flow in only one direction, so polarity is important. The LED should have a flat side indicating the *cathode* or negative terminal. Also, the *anode* (positive terminal) generally has a longer lead than the *cathode*.**

- In this configuration a LOW, or 0V, at P8 will allow current to flow through the LED to Vdd (+5V) lighting it. When P8 is HIGH (+5V), no current will flow and the LED will not light. The LED is *Active Low*.

# Blinking the LED with HIGH, LOW

- Use the Stamp Editor to enter the following program:

```
‘Prog 4A: Blink LED program

Main:
      HIGH 8            'Turn off LED
      PAUSE 1000        'Wait 1 second
      LOW 8             'Turn on LED
      PAUSE 5000        'Wait 5 seconds
GOTO Main               'Jump back to beginning
```

- Download or run the program.

- Monitor the LED.  It should blink at a rate of 1 second OFF, 5 seconds ON.  If not, check your configuration and code.

# BUTTON Instruction

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

***Pin*** (0-15) specify the I/O pin and set it to input mode.

***DownState*** (0 or 1) specify the logical state when the button is pressed.

***Delay*** (0-255) specify minimum press time before auto-repeat starts.

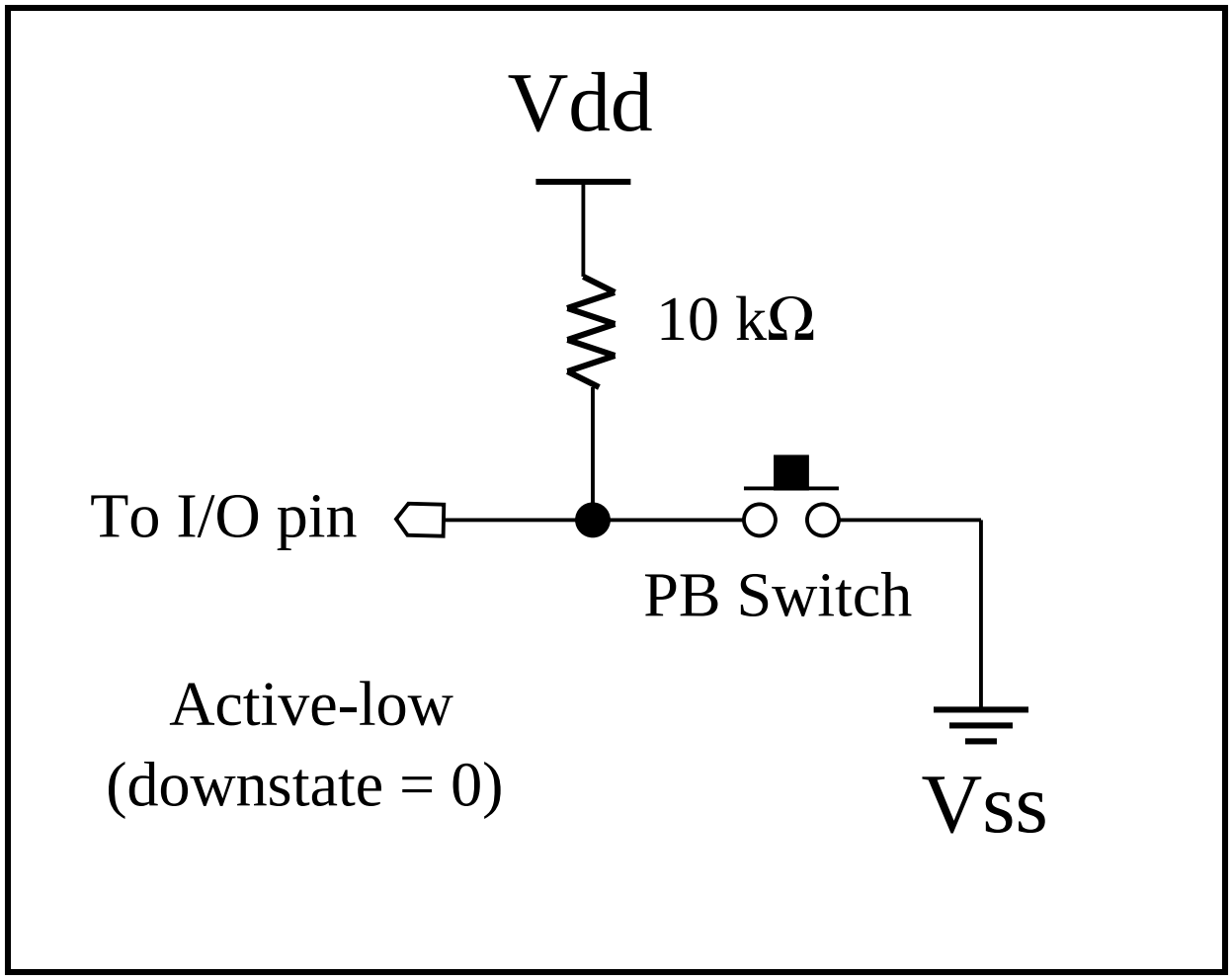
***Rate*** (0-255) specify number of cycles between auto-repeats.

***Workspace*** a byte variable used by BUTTON for workspace.

***TargetState*** (0 or 1) specify the state to branch

***Address*** a label specifying where to branch

# Simple BUTTON circuit



# Demo Program (BUTTON.bas)

```
‘With the active-low circuit connected to pin 0,
‘when you press the button, anasterisk(*) will
‘be printed on the screen.

BtnWrk  VAR BYTE    ‘Workspace
Loop:
   BUTTON 0, 0, 255, 250,BynWrk,0,NoPress
   DEBUG “*”
   ‘can take other actions, such as turn on/off the wheels
NoPress:
  GOTO Loop
```