PBASIC Sample programs explanation - IVE StampBot is the BASIC Stamp carrier board

1. **P1.bs2 - Port B (P4 – P7) LEDs blink**
2. **P2.bs2 - Port B (P4 – P7) LEDs blink in 0101 then 1010 format (0-LED off, 1-LED on)**
3. **P3.bs2 - Four pins Port A (P0 to P3) set to input. Display the status of all pins Port A (P0 to P3) on the Debug terminal.**
4. **P4.bs2 - Make use of INA(P0 to P3) to control OUTB (P4 to P7 LEDs)**
5. **P5.bs2 - Use of Exculsive OR "^"**
6. **P6.bs2 - Define Tunes – FREQOUT**
7. **P7.bs2 - Moving the StampBot robot car forward and turn right**

8A. **P8A.bs2 - StampBot robot car runs a square**

8B. **P8B.bs2 - StampBot robot car runs a square using GOSUB and REURN commands**

9. **P9.bs2 - StampBot robot car control by serial port programming – using commands SERIN, SEROUT to receive and send asynchronous serial data through RS232 cable.**
10. **P10.bs2 - Ultrasonic modules control StampBot robot car motion**

P1.bs2 – Port B (P4 – P7) LEDs blink

```
'{$STAMP BS2}          'To indicate BASIC Stamp 2 module is used
'{$PBASIC 2.5}         ‘To indicate which version of the PBASIC language is used
‘Compiler Directives are special instructions to ensure the the code is tokenized for the correct PBASIC ‘version and for the correct BASIC Stamp.

                       ‘All Ports initialised
DIRA=0                 'Define (Port A) P0 to P3 is input port %0000=0, bitwise, 0 – input, 1 - output
DIRB=15                'Define (Port B) P4 to P7 is output port %1111=15
DIRC=15                'Define (Port C) P8 to P11 is output port (Motor Control port) %1111=15
DIRD=15                'Define (Port B) P12 to P15 is output port %1111=15
OUTC=0                 'Ouptput 0 (%0000) to Port C, turn off motor %0000=0

MAIN:                  ' A Label
OUTB = 15              'Output %1111 = 15 to PortB (P4-P7), therefore all LEDs of Port B is on
PAUSE 2000             'Delay 2000msec = 2 sec, hold LED on for 2 sec
OUTB = 0               'Output %0000 = 0 to PortB (P4-P7), therefore all LEDs of Port B is off
PAUSE 2000             'Delay 2000msec = 2 sec, hold LED off for 2 sec
GOTO MAIN              'Go back to label main, therefore LED is blinking.
```

P2.bs2 – Port B (P4 – P7) LEDs blink in 0101 then 1010 format (0-LED off, 1-LED on)

```
'{$STAMP BS2}
'{$PBASIC 2.5}

DIRA = 0
DIRB = 15
DIRC = 15
DIRD = 15
OUTC= 0

MAIN:
OUTB = %0101           'Output %0101 or 7 to Port B (P4-P7), LEDs lit in this pattern
PAUSE 2000             'Delay 2000msec = 2 sec, hold LEDs lit in this pattern
OUTB = %1010           'Output %1010 or 10 to port B (P4-P7), LEDs lit in this pattern
PAUSE 2000             'Delay 2000msec = 2 sec, hold LEDs lit in this pattern
GOTO MAIN              'Repeat the process
```

P3.bs2 - Four pins Port A (P0 to P3) set to input. Display the status of all pins Port A (P0 to P3) on the Debug terminal.

```
‘{$STAMP BS2}
'{$PBASIC 2.5}

DIRA = 0
DIRB = 15
DIRC = 15
DIRD = 15
OUTC= 0
```

```
MAIN:
DEBUG BIN INA, CR       ‘Debug terminal displays individual pin of Port A(P0 to P3) status, display 1111 when no
                        ‘button is pressed, CR carriage return means next line
PAUSE 2000
GOTO MAIN
```

Explanation:

DEBUG BIN INA, CR
INA is input the value from PortA (P0-P3). IN0 is input the value from PortA P0 only. Therefore the debug terminal displays the status of (Port A) P0 to P3 together instead of P0 only. Port A is set to INPUT as DIRA is 0.
The pins will show “0” as long as you press the corresponding push button.

P4.bs2 - Make use of INA(P0 to P3) to control OUTB (P4 to P7 LEDs)

```
'{$STAMP BS2}
'{$PBASIC 2.5}
DIRA = 0
DIRB = 15
DIRC = 15
DIRD = 15
OUTC= 0

MAIN:
OUTB = INA              'Input the value from PortA (P0-P3) and output this value to PortB (P4-P7). Ie. PortB
                        ‘LEDs on or off base on the status of PortA (P0-P3). Any button is pressed, corresponding LED
                        ‘will be off
PAUSE 2000
GOTO MAIN

‘Add these two lines into after MAIN to view the status.
‘DEBUG “OUTB LED    = “, BIN OUTB, CR
‘DEBUG “INA Push-Button = “, BIN INA, CR, CR
```

P5.bs2 - Use of Exculsive OR “^”

```
'{$STAMP BS2}
'{$PBASIC 2.5}

DIRA = 0
DIRB = 15
DIRC = 15
DIRD = 15
OUTC= 0

MAIN:                   ‘BIN4 means display all 4 bits status on the debug terminal
DEBUG BIN4 INA^15,CR    'INA^15, Port A (P0 - P3) exclusive OR with 15 (%1111) bitwise same as it inverts
                        ‘the input bitwise, the result displays on the debug terminal.
OUTB = INA^15           ‘It displays the result on PortB (P4 - P7), LED on is 1, off is 0
PAUSE 2000              ‘delay 2000 msec = 2 sec
GOTO MAIN
```

(^ means Bitwise XOR. The Exclusive OR operator returns the bitwise exclusive OR of two values. Each bit of the values is subject to the following logic:

Ø ^ Ø = Ø
Ø ^ 1 = 1
1 ^ Ø = 1
1 ^ 1 = Ø)

P6.bs2 - Define Tunes – FREQOUT

```
'{$STAMP BS2}
'{$PBASIC 2.5}

C    CON    523*4       ‘PBASIC allows several numbering systems.
D    CON    587*4       ‘For example: 99 (decimal), %1010 (binary), $FE (hex), "A" (ASCII code for A (65)
E    CON    659*4       ‘BS1 - SYMBOL   Cheers   = 3, the syntax same as BS2 -   Cheers    CON     3
```

```
G    CON      784*4
DIRA=0
DIRB=15
DIRC=15
DIRD=15
OUTC=0

MAIN:
FREQOUT 12,500,E,E*2          ‘ Out put the frequency (sound) on PortD (P12)
FREQOUT 12,500,D,D*2
FREQOUT 12,500,C,C*2
FREQOUT 12,500,D,D*2
FREQOUT 12,500,E,E*2
FREQOUT 12,500,E,E*2
FREQOUT 12,500,E,E*2
PAUSE 2000
GOTO MAIN
```

Explanation

C, D, E, G are constants - while the program is running, nothing can change those numbers.
The difference between constants and variables is that contents/values of variables can be changed while the program is running.
(FREQOUT Pin, Duration, Freq1 {, Freq2}
Pin is 0 - 15 that specifies the I/O pin. This pin will be set to output mode.
Duration - 0 – 65535 that specifies the amount of time to generate the tone(s).
Freq1 is 0 – 32767 specifies frequency of the first tone.
Freq2 is an optional argument exactly like Freq1.
When specified, two frequencies will be mixed together on the specified I/O pin.

	BS2
Units in Duration	1 ms
Units in Freq1 and Freq2	1 Hz
Range of frequency	0 to 32767

Example : FREQOUT 2, 1000, 2500, 3000
This will generate a 2500 Hz and 3000 Hz tone (on the BS2) for 1 second. The frequencies will mix together for a chord- or bell-like sound. To generate a silent pause, specify frequency value(s) of 0.

P7.bs2 - Moving the StampBot robot car forward and turn right

```
' {$STAMP BS2}
' {$PBASIC 2.5}

DIRA = 0
DIRB = 15
DIRC = 15
DIRD = 15
OUTC= 0

MAIN:
OUTC= %1111          'FORWARD
PAUSE 1000
OUTC= 0              'Out PortC (P8 - P11) =0 (%0000) is stop the car
PAUSE 200
OUTC= %1101          'TURN RIGHT
PAUSE 1000
OUTC= 0              'STOP
PAUSE 200
GOTO MAIN
```

P8A.bs2 - StampBot robot car runs a square

```
'{$STAMP BS2}
'{$PBASIC 2.5}

DIRA = 0
DIRB = 15
DIRC = 15
```

```
DIRD = 15
OUTC= 0

MAIN:
OUTC= %1111    'FORWARD
PAUSE 1000
OUTC= 0
PAUSE 200

OUTC= %1101    'TURNRIGHT
PAUSE 1000
OUTC= 0
PAUSE 200

OUTC= %1111    'FORWARD
PAUSE 1000
OUTC= 0
PAUSE 200

OUTC= %1101    'TURNRIGHT
PAUSE 1000
OUTC= 0
PAUSE 200
GOTO MAIN
```

P8B.bs2 - StampBot robot car runs a square use GOSUB and REURN commands

```
'{$STAMP BS2}
'{$PBASIC 2.5}
DIRA = 0
DIRB = 15
DIRC = 15
DIRD = 15
OUTC= 0

MAIN:
GOSUB FORWARD          'It goes to the label and executes the lines in FROWARD
GOSUB TURNRIGHT        'It goes to the label and executes the lines in TURNRIGHT
GO SUB FORWARD
GO SUB TURNRIGHT
GO SUB FORWARD
GO SUB TURNRIGHT
GOTO MAIN

FORWARD:
OUTC= %1111
PAUSE 1000
OUTC= 0
PAUSE 200
RETURN

TURNRIGHT:
OUTC= %1101
PAUSE 1000
OUTC= 0
PAUSE 200
RETURN
```

P9.bs2 – StampBot robot car control by serial port programming – use command SERIN, SEROUT to receive and send asynchronous serial data through RS232 cable.

```
'{$STAMP BS2}
'{$PBASIC 2.5}
SerString    VAR    Byte      ‘SerSttring variable type is Byte (Variable type can be - Bit, Nib, Byte, or Word)
DIRA = 0
DIRB = 15
DIRC = 15
DIRD = 15
OUTC= 0
```

```
DEBUG "MAIN",CR          'Debug terminal display Keyboard Input, CR – Carriage Return : next line

MAIN:

SerString = 0            ‘the equal sign (=) does not means SerString equals to 0 but to empty it. Otherwise it may
                         ‘store an unwanted value.

SERIN 16,16468,100,NODATA,[SerString]
IF SerString = "F" THEN GOSUB FORWARD
IF SerString = "B" THEN GOSUB BACKWARD
IF SerString = "L" THEN GOSUB LEFT
IF SerString = "R" THEN GOSUB RIGHT
GOTO MAIN
                                  ‘“IF SerString = "B" THEN BACKWARD” means that if the content of the variable
                                  SerString is “F” then it will execute the lines in Label FORWARD.
FORWARD:
OUTC= %1111
PAUSE 1000       'Go forward for 1000msec = 1 sec
OUTC= 0          ‘Output 0 (%0000) to PortC (P8-P11), motor stop
PAUSE 200
RETURN

BACKWARD:
OUTC= %0101
PAUSE 1000       'Go backward for 1000msec = 1 sec
OUTC = 0
PAUSE 200
RETURN

LEFT:
OUTC= %0111
PAUSE 500        'Turn left angle: large no. means turn large angle & small no. means turn small angl
OUTC = 0
PAUSE 200
RETURN

RIGHT:
OUTC= %1101
PAUSE 500        ‘Turn right angle: large no. means turn large angle & small no. means turn small angle
OUTC = 0
PAUSE 200
RETURN

NODATA:
GOTO MAIN
```

Explanation

SERIN 16,16468,100,NODATA,[SerString]
SERIN Rpin, Baudmode, {Timeout, Tlabel,}, [InputData]

Rpin - specifies the I/O pin that receives serial data. This pin will be set to input mode. For BS2, if Rpin is set to 16, the BASIC Stamp uses the dedicated serial-input pin (SIN, physical pin 2 of Basic Stamp), which is normally used by the Stamp Editor during the download process.

Baudmode - specifies serial timing and configuration.

Baud Rate	8-bit No Parity INVERTED
9600	16468

Timeout - is an optional variable (0 - 65535) that tells SERIN how long (100msec=0.1 sec in this example) to wait for incoming data. If data does not arrive in time, the program will jump to the address specified by Tlabel.

Tlabel - is an optional label (NODATA in this example) that must be provided along with Timeout, indicating where the program should go whent data does not arrive within the period specified by Timeout.

InputData - is list of variables and formatters that tells SERIN what to do with incoming data.

P10.bs2 - Ultrasonic modules control StampBot robot car motion

```
'{$STAMP BS2}
'{$PBASIC 2.5}
ECHO_L     CON     12          'P12 - Input pin, to recevie echo output from left ultrasonic module
```

```
INIT_L      CON     13          'P13 - Ouput pin, to output pulse to the input of the left ultrasonic module
ECHO_R      CON     14          'P14 - Input pin, to recevie echo output from right ultrasonic module
INIT_R      CON     15          'P15 - Output pin, to output pulse to the input of the right ultrasonic module
wDistL      VAR     Word
wDistR      VAR     Word
                                'CONVERSION FACORS
                                'DISTANCE = (ECHO TIME) / (CONVERSION FACTOR)
                                'use 29 for centimeters  (29.0333 s per 1 cm)
Convfac     CON     29          'use cm


DIRA = 0
DIRB = 15
DIRC = 15
DIRD = %1010                    'PortD - Pin12 & 14 are 0 ( Input). Pin 13 & 15 are 1 (Ouptut)
OUTC= 0

MAIN:
DEBUG "MAIN",CR                 ‘Degbug terminal displays the “MAIN”
GO SUB READ_ENVIR               ‘Go to subroutine to measure ths distance
IF ( wDistL > 15 ) AND ( wDistR > 15 ) THEN FORWARD
IF ( wDistL < 15 ) AND ( wDistR < 15 ) THEN BACKWARD
IF ( wDistL < 15 ) AND ( wDistR > 15 ) THEN RIGHT
IF ( wDistL > 15 ) AND ( wDistR < 15 ) THEN LEFT
GOTO MAIN
‘Base on 15cm, compare the ultrasonic module measured distance to determine the motion of the robot car.

READ _ENVIR:
GOSUB SR_SONAR_L                'Jump to SR_SONAR_L to measure the distance
DEBUG "wDistL = ", DEC wDistL,CR 'Display decimal value of wDistL, left ultrasonic module measured distance
PAUSE 10
GOSUB SR_SONAR_R                'Jump to SR_SONAR_R to measure the distance
DEBUG "wDistR = ", DEC wDistR,CR ‘Display decimal value of wDistR, right ultrasonic module measured ‘distance
PAUSE 10
RETURN

FORWARD:                        ‘Robot car moving forward
OUTC= %1111
PAUSE 250
OUTC = 0
RETURN

BACKWARD:                       ‘Robot car moving backward
OUTC= %0101
PAUSE 250
OUTC = 0
RETURN

LEFT:                           ‘Robot car left turn
DEBUG "LEFT",CR
OUTC= %1101
PAUSE 150
OUTC = 0
RETURN

RIGHT:                          ‘Robot car right turn
DEBUG "RIGHT",CR
OUTC= %0111
PAUSE 150
OUTC = 0
RETURN

SR _SONAR _L:                   ‘Subroutine - to measure the distance between the barrier and the ultrasonic module
PULSOUT     INIT_L,5            ' Send pulse to left ultrasonic module
RCTIME ECHO_L,1,wDistL          ' Measure Echo Time from the left ultrasonic module
wDistL = wDistL / Convfac       ' convert to cm
PAUSE   10
```

```
RETURN

SR_SONAR_R:                 'Subroutine - to measure the distance between the barrier and the ultrasonic module
PULSOUT     INIT_R,5        ' Send pulse to right ultrasonic module
RCTIME ECHO_R,1,wDistR      ' Measure Echo Time from the right ultrasonic module
wDistR = wDistR / Convfac   ' convert to cm
PAUSE   10
RETURN
```

Explanation:

```
ECHO_L      CON     12
INIT_L      CON     13
ECHO_R      CON     14
INIT_R      CON     15
```

Two ultrasonic modules Left (L) and Right (R) are used to detect the barrier like human eyes.
Left module, ECHO_L and INIT_L are declared as constants. I/O Pin 12 and Pin 13 are declared as the input and output pins respectively..
Right module, ECHO_R and INIT_R are declared as constants. I/O Pin 14 and Pin 15 are declared as the input and output pins respectively..
ECH_L and ECHO_R are used to receive signals from the Echo Output of the ultrasonic module.
NIT_L and INIT_R generate pulses to the Pulse Trigger Inputs of the ultrasonic modules.

```
wDistL VAR Word
wDistR VAR Word
Convfac  CON 29
```

wDistL and wDistR are the variables that hold the calculated distance from labels " SR _SONAR _L:" and "SR _SONAR _R:" respectively.
Convfac is a conversion factor used to convert time in µs to distance in cm.

```
DIRD = %1010
```

Define pin 12 and pin 14 are inputs, and pin 13 and pin 15 are outputs.

```
GOSUB READ_ENVIR
```

It will jump to the label called READ_ENVIR which has another two GOSUBs.

```
IF ( wDistL > 15 ) AND ( wDistR > 15 ) THEN FORWARD
IF ( wDistL < 15 ) AND ( wDistR < 15 ) THEN BACKWARD
IF ( wDistL < 15 ) AND ( wDistR > 15 ) THEN RIGHT
IF ( wDistL > 15 ) AND ( wDistR < 15 ) THEN LEFT
```

They make the decision based on the values of wDistL and wDistR that can fulfil the conditions.

```
SR_SONAR_L:
PULSOUT INIT_L,5
RCTIME ECHO_L,1,wDistL
wDistL = wDistL / convfac
PAUSE 10
RETURN
```

```
PULSOUT INITL,5
```

Pin 13 generates a 10µS pulse to the Pulse trigger Input of the module. See Note 1

```
RCTIME ECHO L,1,wDistL.
```

Pin 12 reads the signal from the Echo Output of the module to change the state of the Pin. Once Pin is not in State, the command ends and stores the result in Variable wDistL. See Note 2.

```
wDistL = wDistL / convfac
```

This line convets the time stored the result of wDistL from RCTIME to distance and update the wDistL. It works from right to left of the equal sign. The same principle applies to SR_SONAR_R.

Note 1
PULSOUT Pin, Duration
Function: Generate a pulse on Pin with unit of Duration.
Pin is 0 - 15 that specifies the I/O pin to use. This pin will be set to output mode.
Duration is 0 - 65535 that specifies the duration of the pulse. The unit of time for Duration is described below.

	BS2 and BS2e
Units in Duration	2 µs
Maximum pulse width Duration	131.07 ms

PULSOUT sets Pin to output mode, inverts the state of that pin; waits for the specified Duration; then inverts the state of the pin again; returning the bit to its original state. The unit of Duration is described above.

PULSOUT INIT_L,5
'INIT_L is P13, 5 unit duration x 2µs = 10µs, It generates a 10 µs pulse on I/O pin 13.

Note 2
RCTIME Pin, State, Variable
Function: Measure time while Pin remains in State; the charge/discharge time of resistor/capacitor (RC) circuit is used.
Pin is 0 – 15 that specifies the I/O pin to use. This pin will be defined as input mode.
State is 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 - (usually a word) the time measurement will be stored. The unit of time for Variable is described below.

	BS2 and BS2e
Units in Variable	2 µs
Maximum pulse width	131.07 ms

RCTIME can also serve as a fast, precise stopwatch for events of very short duration.
When RCTIME executes, it starts a counter (unit of time is shown above). It stops this counter as soon as the specified pin is no longer in State (0 or 1). If pin is not in State when the instruction executes, RCTIME will return 1 in Variable, since the instruction requires one timing cycle to discover this fact. If pin remains in State longer than 65535 timing cycles RCTIME returns 0.

RCTIME ECHO_L,1,wDistL
ECHO_L is I/O Pin 12 and is defined as an input pin used to receive signal from the left ultrasonic module.
1 is the defined state value
wDistL store the no. of variable until state changed.