BASIC Stamp I and Stamp II Conversions

BASIC Stamp I and Stamp II Conversions

BASIC Stamp I and Stamp II Conversions

INTRODUCTION

The BASIC Stamp I and BASIC Stamp II have many differences in both hardware and software. While it is trivial to recognize the differences in the Stamp hardware, the modifications to the PBASIC command structure are intricate and not always obvious. This appendix describes the Stamp I and Stamp II PBASIC differences in a detailed manner to aid in the conversion of programs between the two modules. This document may also serve to give a better understanding of how certain features of the two versions can be helpful in problem solving.

TYPOGRAPHICAL CONVENTIONS

This Appendix will use a number of symbols to deliver the needed information in a clear and concise manner. Unless otherwise noted the following symbols will have consistent meanings throughout this document.

TOPIC HEADING

Each discussion of a topic or PBASIC command will begin with a topic heading such as the one above.

MODULE HEADING

When separate discussion of a Stamp I or Stamp II module is necessary it will begin with a module heading such as this one.

-
-

Inside the module section bulleted items will precede information on the properties of various arguments for the indicated command.

CONVERSION:

When conversion between the two versions of PBASIC are necessary, each set of steps will begin under the conversion heading as shown above. This header will always begin with the word “Conversion” and will indicate in which direction the conversion is taking place; i.e. from BS1 to BS2 or from BS2 to BS1.

1. First do this...
2. Next do this...

The most important steps in conversion will be listed in a numeric sequence within the conversion section. The order of the numbered steps may be important in some situations and unimportant in others; it is best to follow the order as closely as possible.

Tips which are not vital to the conversion are listed within the conversion section and are preceded by bullets as shown above. These tips include additional information on valid argument types, properties of the command, etc. and may be used for further optimization of the code if desired.

As an example, using the above conventions, a typical section within this document will look like this:

SAMPLE COMMAND

BASIC Stamp I

Command syntax line shown here

- Argument one is…
- Argument two is…

BASIC Stamp II

Command syntax line shown here

- Argument one is...
- Argument two is...

Conversion: BS1 ⇨ BS2

1. First do this...
2. Next do this...

- You might like to know this...
- You might want to try this...

CONVERSION: BS1 ⇦ BS2

1. First do this...
2. Next do this...

- You might like to know this...
- You might want to try this...

The following symbols appear within command syntax listings or within the text describing them.

UPPER CASE	All command names will be shown is upper case lettering within the command syntax line. Argument names will be in upper case lettering outside of the command syntax line.
lower case	All arguments within the command syntax line will be in lower case lettering.
( )	Parentheses may appear inside a command syntax line and indicate that an actual parenthesis character is required at that location.
[ ]	Brackets may appear inside a command syntax line and indicate that an actual bracket character is required at that location.
[|]	Brackets with an internal separator may appear in the text following a command syntax line and indicate that one, and only one, of the items between the separators may be specified.
{ }	Wavy brackets may appear inside a command syntax line and indicate that the items they surround are optional and may be left out of the command. The wavy bracket characters themselves should not be used within the command, however.

#..#	Double periods between numbers indicate that a contiguous range of numbers are allowed for the given argument. Wherever a range of numbers are shown it usually indicates the valid range which a command expects to see. If a number is given which is outside of this range the Stamp will only use the lowest bits of the value which correspond to the indicated range. For example, if the range 0..7 is required (a 3 bit value) and the number 12 is provided, the Stamp will only use the lowest 3 bits which would correspond to a value of 4.

HOW TO USE THIS APPENDIX

This appendix should be used as a reference for converting specific commands, or other PBASIC entities, from one version of the Stamp to another. While this document will help to convert most of the programs available for the Stamp I and Stamp II, some programs may require logic changes to achieve correct results. The required logic changes are beyond the scope of this document.

In an effort to lessen the time spent in performing a code conversion the following routine should be followed in the order listed for each program.

1. Review the entire code briefly to familiarize yourself with how it functions and the types of commands and expressions which are used.
2. Consult the RAM SPACE AND REGISTER ALLOCATION section in this manual and go through the entire program carefully converting symbols, variables and expressions to the proper format.
3. Go through the code instruction by instruction, consulting the appropriate section in this document, and convert each one to the appropriate form.
4. Make any necessary circuit changes as required by the new stamp code.

COMMAND AND DIRECTIVE DIFFERENCES

Many enhancements to the Stamp I command structure were made in the Stamp II. Commands have also been added, replaced or removed. The following table shows the differences between the two modules.

BASIC Stamp I	BASIC Stamp II	Comments
BRANCH	BRANCH	Syntax Modifications
BSAVE		Removed
BUTTON	BUTTON	
	COUNT	New Command
DEBUG	DEBUG	Enhanced
EEPROM	DATA	Enhanced
	DTMFOUT	New Command
END	END	
(Expressions)	(Expressions)	Enhanced
FOR...NEXT	FOR...NEXT	Enhanced
GOSUB	GOSUB	Enhanced
GOTO	GOTO	
HIGH	HIGH	
IF...THEN	IF...THEN	Enhanced
INPUT	INPUT	
LET	(Expression)	Enhanced
LOOKDOWN	LOOKDOWN	Enhanced
LOOKUP	LOOKUP	Syntax Modifications
LOW	LOW	
NAP	NAP	
OUTPUT	OUTPUT	
PAUSE	PAUSE	
POT	RCTIME	Enhanced
PULSIN	PULSIN	Enhanced
PULSOUT	PULSOUT	Enhanced
PWM	PWM	Enhanced
RANDOM	RANDOM	
READ	READ	
(Register Allocation)	(Register Allocation)	Enhanced
REVERSE	REVERSE	
SERIN	SERIN	Enhanced
SEROUT	SEROUT	Enhanced
	SHIFTIN	New Command
	SHIFTOUT	New Command
SLEEP	SLEEP	
SOUND	FREQOUT	Enhanced
	STOP	New Command
TOGGLE	TOGGLE	
WRITE	WRITE	
	XOUT	New Command

BASIC Stamp I and Stamp II Conversions

RAM SPACE AND REGISTER ALLOCATION

BASIC Stamp I

The RAM space in the BASIC Stamp I consists of eight 16-bit words. Each word has a unique, predefined name as shown in the table below. Each word consists of two 8-bit bytes which have unique, predefined names. Additionally the first two words, PORT and W0, can be accessed as individual bits.

The first word, named PORT, is reserved to allow access and control over the 8 I/O pins on the Stamp I. This word consists of two bytes, PINS and DIRS, which represent the status and the data direction of the pins.

The other seven words are general purpose registers for use by the PBASIC program. They may be used via their direct name or by assigning symbols as aliases to specific registers.

To assign a symbol to a specific register, use the following format:

SYMBOL symbolname = registername

Example: SYMBOL LoopCounter = W0

- SYMBOLNAME is a series of characters (letters, numbers and underscores but not starting with a number) up to 32 characters in length.
- REGISTERNAME is a valid bit, byte or word register name as shown in the table below.

You may assign a symbol to a constant value by using a similar format:

SYMBOL symbolname = constantvalue

Example: SYMBOL MaxLoops = 100

- SYMBOLNAME is a series of characters (letters, numbers and underscores but not starting with a number) up to 32 characters in length.

- CONSTANTVALUE is a valid number in decimal, hexidecimal, binary or ascii.

Stamp I I/O and Variable Space			
Word Name	**Byte Name**	**Bit Names**	**Special Notes**
PORT	PINS	PIN0 - PIN7	I/O pins; bit addressable.
	DIRS	DIR0 - DIR7	I/O pin direction control; bit addressable.
W0	B0	BIT0 - BIT7	Bit addressable.
	B1	BIT8 - BIT15	Bit addressable.
W1	B2		
	B3		
W2	B4		
	B5		
W3	B6		
	B7		
W4	B8		
	B9		
W5	B10		
	B11		
W6	B12		Used by GOSUB instruction.
	B13		Used by GOSUB instruction.

BASIC Stamp II

The RAM space of the BASIC Stamp II consists of sixteen words of 16 bits each. Each word and each byte within the word has a unique, predefined name similar to the Stamp I and shown in the table below.

The first three words, named INS, OUTS and DIRS, are reserved to allow access and control over the 16 I/O pins on the Stamp II. These reserved words represent the input states, output states and directions of the pins respectively and are the Stamp II version of the single control word, PORT, on the Stamp I. In comparison to the Stamp I, the control registers' size has been doubled and the I/O register PINS has been split into two words, INS and OUTS, for flexibility. Each word consists of a predefined name for its byte, nibble and bit parts.

The other thirteen words are general purpose registers for use by the PBASIC program. There are two methods of referencing these registers within the Stamp II as follows:

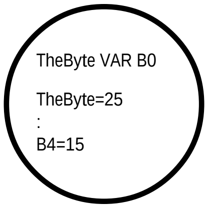
1. They may be referenced via their direct name or by defining symbols as aliases.

   - OR -

2. They may be referenced by defining variables of specific types (byte, word, etc.). The software will automatically assign variables to registers in an efficient manner.

The first method is used in the Stamp I, and supported in the Stamp II, as a means of directly allocating register space. The second method was introduced with the Stamp II as a means of indirectly allocating register space and is the *recommended method*.

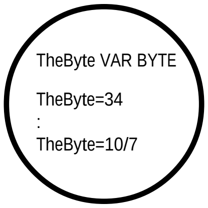
It is important to note that defining variables of specific types in the Stamp II is not directly equivalent to assigning symbols to registers in the Stamp I. Defining variables of specific types on the Stamp II allows the software to efficiently and automatically organize variable space within the general purpose registers while assigning symbols to registers allows you to organize variable space yourself. While both methods of register allocation are legal in the Stamp II, care should be taken to implement only one method of register use within each program. Each PBASIC program should either reference all registers by their predefined names (or symbols assigned to them) or reference all registers by defining variables of specific types and let the software do the organization for you. If you use both methods within the same program, it is likely that variables will *overlap* and your program will behave erratically. The following diagram may serve to clarify the use of the register allocation methods within a single Stamp II program:



```
TheByte VAR B0

TheByte=25
:
B4=15
```

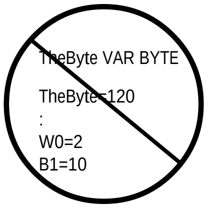
Using only method 1 within a program is safe.



```
TheByte VAR BYTE

TheByte=34
:
TheByte=10/7
```

Using only method 2 within a program is safe.



```
TheByte VAR BYTE

TheByte=120
:
W0=2
B1=10
```

Using both methods within a program leads to erratic execution.

To define a variable of a specific type, use the following format.

```
variablename  VAR  [ type{(arraysize)} | previousvariable{.modifier{.modifier...}} ]
```

Example:

```
LoopCounter     VAR  WORD               'defines LoopCounter as a word.
LoopCounter2    VAR  BYTE(2)            'defines LoopCounter2 as an array of
                                        'two bytes.
FirstBit        VAR  LoopCounter.LOWBIT 'defines FirstBit as the lowest bit
                                        'within the variable LoopCounter.
```

- VARIABLENAME is a series of characters (letters, numbers and underscores but not starting with a number) up to 32 characters in length.
- TYPE is a valid variable type of BIT, NIB, BYTE or WORD.
- ARRAYSIZE is an optional constant value, in parentheses, specifying the number of elements of TYPE to define for the variable VARIABLENAME.
- PREVIOUSVARIABLE is the name of a previously defined variable. This can be used to assign alias names to the same variable space.
- MODIFIER is an optional offset, preceded by a period '.', which indicates which part of a previously defined variable to set VARIABLENAME to. Valid modifiers are: LOWBYTE, HIGHBYTE, BYTE0..1, LOWNIB, HIGHNIB, NIB0..3, LOWBIT, HIGHBIT and BIT0..15.

You may define a constant by using a similar format:

```
constantname  CON  constantexpression
```

Example:

```
MaxLoops     CON  100          'defines MaxLoops as a constant
                               'equivalent to the number 100.
MaxLoops2    CON  50 * 4 / 2   'also defines MaxLoops as a
                               'constant equivalent to the number 100.
```

- CONSTANTNAME is a series of characters (letters, numbers and underscores but not starting with a number) up to 32 characters in length.
- CONSTANTEXPRESSION is a numerical expression in decimal, hexidecimal, binary or ascii using only numbers and the +, -, *, /, &, |, ^, << or >> operators. NOTE: Parentheses are not allowed and expressions are always computed using 16-bits.

Stamp II 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; word, byte, nibble and bit addressable.
OUTS	OUTL OUTH	OUTA, OUTB, OUTC, OUTD	OUT0 - OUT7, OUT8 - OUT15	Output pins; word, byte, nibble and bit addressable.
DIRS	DIRL DIRH	DIRA, DIRB, DIRC, DIRD	DIR0 - DIR7, DIR8 - DIR15	I/O pin direction control; word, byte, nibble and bit addressable.
W0	B0 B1			General Purpose; word, byte, nibble and bit addressable.
W1	B2 B3			General Purpose; word, byte, nibble and bit addressable.
W2	B4 B5			General Purpose; word, byte, nibble and bit addressable.
W3	B6 B7			General Purpose; word, byte, nibble and bit addressable.
W4	B8 B9			General Purpose; word, byte, nibble and bit addressable.
W5	B10 B11			General Purpose; word, byte, nibble and bit addressable.
W6	B12 B13			General Purpose; word, byte, nibble and bit addressable.
W7	B14 B15			General Purpose; word, byte, nibble and bit addressable.
W8	B16 B17			General Purpose; word, byte, nibble and bit addressable.
W9	B18 B19			General Purpose; word, byte, nibble and bit addressable.
W10	B20 B21			General Purpose; word, byte, nibble and bit addressable.
W11	B22 B23			General Purpose; word, byte, nibble and bit addressable.
W12	B24 B25			General Purpose; word, byte, nibble and bit addressable.

SYMBOL CONVERSION: BS1 ⇨ BS2

1. Remove the ‘SYMBOL’ directive from variable or constant declarations.

2. On all variable declarations, replace the predefined register name, to the right of the ‘=’, with the corresponding variable type or register name according to the following table:

BS1 to BS2 Register Allocation Conversion	
Stamp I Register Name	**Stamp II Variable Type / Register Name**
PORT	NO EQUIVALENT*
PINS or PIN0..PIN7	INS / OUTS or IN0..IN7 / OUT0..OUT7**
DIRS or DIR0..DIR7	DIRS or DIR0..DIR7
W0..W6	WORD
B0..B13	BYTE
BIT0..BIT15	BIT

* The PORT control register has been split into three registers, INS, OUTS and DIRS, on the Stamp II. There is no predefined name representing all registers as a group as in the Stamp I. Additional symbol and / or program structure and logic changes are necessary to access all three registers properly.

** The Stamp I PINS register has been split into two registers, INS and OUTS, in the Stamp II. Each register now has a specific task, input or output, rather than a dual task, both input and output, as in the Stamp I. If the Stamp I program used the symbol assigned to PINS for both input and output, an additional symbol is necessary to access both functions. This may also require further changes in program structure and logic.

1. On all variable declarations, replace the equal sign, ‘=’, with ‘VAR’.

2. On all constant declarations, replace the equal sign, ‘=’, with ‘CON’.

VARIABLE OR CONSTANT CONVERSION: BS1 ⇦ BS2

1. Insert the ‘SYMBOL’ directive before the variable’s name or constant’s name in the declaration.

2. On all variable declarations, replace the variable type or register name, to the right of the ‘=’, with the corresponding, predefined register name according to the following table:

BS2 to BS1 Register Allocation Conversion	
Stamp II Variable Type / Register Name	**Stamp I Register Name**
INS	PINS
OUTS	PINS
DIRS	DIRS
WORD	W0..W6
BYTE	B0..B13
NIB	B0..B13*
BIT	BIT0..BIT15**

* There are no registers on the Stamp I which are nibble addressable. The best possible solution is to place one or two nibble variables within a byte register and modify the code accordingly.

** The only general purpose registers on the Stamp I which are bit addressable are B0 and B1. BIT0..BIT7 correspond to the bits within B0 and BIT8..BIT15 correspond to the bits within B1. If you have a set of bit registers in the Stamp II program, you should reserve B0 and B1 for this bit usage; i.e.: do not assign any other symbols to B0 or B1.

3. On all variable and constant declarations, replace the variable or constant directive, ‘VAR’ or ‘CON’, with an equal sign, ‘=’.

ASSIGNMENT CONVERSION: BS1 ⇦ BS2

1. Remove the ‘LET’ command if it is specified.
2. If PINS or PIN0..PIN7 appears to the left, or to the left and right, of the equal sign, ‘=’, replace PINS with OUTS and PIN0..PIN7 with OUT0..OUT7.
3. If PINS or PIN0..PIN7 appears to the right of the equal sign, ‘=’, replace PINS with INS and PIN0..PIN7 with IN0..IN7.
4. If PORT appears in an assignment, determine which byte (PINS or DIRS) is affected and replace PORT with the corresponding Stamp II symbol (INS, OUTS or DIRS). If both bytes are affected, separate assignment statements may be needed to accomplish the equivalent effect in the Stamp II.

BRANCH

BASIC STAMP I

BRANCH **index,(label0, label1,... labeln)**

- INDEX is a constant or a bit, byte or word variable.
- LABEL0..LABELN are valid labels to jump to according to the value of INDEX.

BASIC STAMP II

BRANCH **index,[label0, label1,... labeln]**

- INDEX is a constant, expression or a bit, nibble, byte or word variable.
- LABEL0..LABELN are valid labels to jump to according to the value of INDEX.

CONVERSION: BS1 ⇨ BS2

1. Change open and close parentheses, “(“ and “)”, to open and close brackets, “[“ and “]”.

Example:

```
BS1:        BRANCH  B0, ( Loop1, Loop2, Finish )

BS2:        BRANCH  BranchIdx, [ Loop1, Loop2, Finish ]
```

CONVERSION: BS1 ⇦ BS2

1. Change open and close brackets, “[” and “]”, to open and close parentheses, “(“ and “)”.

Example:

```
BS2:        BRANCH BranchIdx, [ Loop1, Loop2, Finish ]

BS1:        BRANCH  B0, ( Loop1, Loop2, Finish )
```

BSAVE

BASIC Stamp I

BSAVE

- This is a compiler directive which causes the Stamp I software to create a file containing the tokenized form or the associated source code.

BASIC Stamp II

NO EQUIVELANT COMMAND

Conversion:

No conversion possible.

BUTTON

BASIC STAMP I

BUTTON pin, downstate, delay, rate, workspace, targetstate, label

- PIN is a constant or a bit, byte or word variable in the range 0..7.
- DOWNSTATE is a constant or a bit, byte or word variable in the range 0..1.
- DELAY is a constant or a bit, byte or word variable in the range 0..255.
- RATE is a constant or a bit, byte or word variable in the range 0..255.
- WORKSPACE is a byte or word variable.
- TARGETSTATE is a constant or a bit, byte or word variable in the range 0..1.
- LABEL is a valid label to jump to in the event of a button press.

BASIC STAMP II

BUTTON pin, downstate, delay, rate, workspace, targetstate, label

- PIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15.
- DOWNSTATE is a constant, expression or a bit, nibble, byte or word variable in the range 0..1.
- DELAY is a constant, expression or a bit, nibble, byte or word variable in the range 0..255.
- RATE is a constant, expression or a bit, nibble, byte or word variable in the range 0..255.
- WORKSPACE is a byte or word variable.
- TARGETSTATE is a constant, expression or a bit, nibble, byte or word variable in the range 0..1.

- LABEL is a valid label to jump to in the event of a button press.

CONVERSION: BS1 ⇨ BS2

1. PIN may be a constant or a bit, nibble, byte or word variable in the range 0..15.
2. Any or all arguments other than LABEL may be nibble variables for efficiency.

Example:

```
BS1:        BUTTON  0, 1, 255, 0, B0, 1, ButtonWasPressed

BS2:        BUTTON  0, 1, 255, 0, WkspcByte, 1, ButtonWasPressed
```

CONVERSION: BS1 ⇦ BS2

1. PIN must be a constant or a bit, byte or word variable in the range 0..7.
2. No arguments may be nibble variables.

Example:

```
BS2:        BUTTON  12, 1, 255, 0, WkspcByte, 1, ButtonWasPressed

BS1:        BUTTON  7, 1, 255, 0, B0, 1, ButtonWasPressed
```

COUNT

BASIC STAMP I

NO EQUIVELANT COMMAND

BASIC STAMP II

COUNT pin, period, result

- PIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15.
- PERIOD is a constant, expression or a bit, nibble, byte or word variable in the range 0..65535.
- RESULT is a bit, nibble, byte or word variable.

CONVERSION:

No conversion possible.

BASIC Stamp I and Stamp II Conversions

DEBUG

BASIC Stamp I

DEBUG outputdata{,outputdata...}

- OUTPUTDATA is a text string, bit, byte or word variable (no constants allowed).
- If no formatters are specified DEBUG defaults to “variablename = value” + carriage return.

FORMATTERS:
(The following formatting characters may precede the variable name)

#	displays value in decimal followed by a space.
$	displays “variablename = $value ” + carriage return; where value is in hexidecimal.
%	displays “variablename = %value ” + carriage return; where value is in binary.
@	displays “variablename = ‘character’ ” + carriage return; where character is an ascii character.

SPECIAL SYMBOLS:
(The following symbols can be included in the output data)

CLS	causes the debug window to be cleared.
CR	causes a carriage return in the debug window.

BASIC Stamp II

DEBUG outputdata{,outputdata...}

- OUTPUTDATA is a text string, constant or a bit, nibble, byte or word variable.
- If no formatters are specified DEBUG defaults to ascii character display without spaces or carriage returns following the value.

FORMATTERS:
(The following formatting tokens may precede the data elements as indicated below)

ASC?	value	Displays “variablename = ‘character’ ” + carriage return; where character is an ascii character.
STR	bytearray	Displays values as an ascii string until a value of 0 is reached.
STR	bytearray\n	Displays values as an ascii string for n bytes.
REP	value\n	Displays value n times.
DEC{1..5}	value	Displays value in decimal, optionally limited or padded for 1 to 5 digits.
SDEC{1..5}	value	Displays value in *signed* decimal, optionally limited or padded for 1 to 5 digits. Value must not be less than a word variable.
HEX{1..4}	value	Displays value in hexidecimal, optionally limited or padded for 1 to 4 digits.
SHEX{1..4}	value	Displays value in *signed* hexidecimal, optionally limited or padded for 1 to 4 digits. Value must not be less than a word variable.
IHEX{1..4}	value	Displays value in hexidecimal preceded by a “$” and optionally limited or padded for 1 to 4 digits.
ISHEX{1..4}	value	Displays value in *signed* hexidecimal preceded by a “$” and optionally limited or padded for 1 to 4 digits. Value must not be less than a word variable.
BIN{1..16}	value	Displays value in binary, optionally limited or padded for 1 to 16 digits.
SBIN{1..16}	value	Displays value in *signed* binary, optionally limited or padded for 1 to 16 digits. Value must not be less than a word variable.

IBIN{1..16}	value	Displays value in binary preceded by a "%" and optionally limited or padded for 1 to 16 digits.
ISBIN{1..16}	value	Displays value in *signed* binary preceded by a "%" and optionally limited or padded for 1 to 16 digits. Value must not be less than a word variable.

SPECIAL SYMBOLS:
(The following symbols can be included in the output data)

BELL	Causes the computer to beep.
BKSP	Causes the cursor to backup one space.
CLS	Causes the debug window to be cleared.
CR	Causes a carriage return to occur in debug window.
HOME	Causes the cursor in the debug window to return to home position.
TAB	Causes the cursor to move to next tab position.

CONVERSION: BS1 ⇨ BS2

1. Replace all '#' formatters with 'DEC'.
2. Replace all '$' formatters with 'HEX?'.
3. Replace all '%' formatters with 'BIN?'.
4. Replace all '@' formatters with 'ASC?'.
5. If variable has no formatters preceding it, add the 'DEC?' formatter before variable.

- Signs, type indicators, strings and digit limitation formatting options are available for more flexibility.

Example:

```
BS1:      DEBUG #B0, $B1, %B2

BS2:      DEBUG DEC AByte, HEX? AWord, BIN? ANibble
```

CONVERSION: BS1 ⇦ BS2

1. Remove any 'DEC?' formatters preceding variables.
2. Replace all 'DEC' formatters with '#'.
3. Replace all 'HEX?' formatters with '$'.
4. Replace all 'BIN?' formatters with '%'.
5. Replace all 'ASC?' formatters with '@'.
6. Delete any '?' formatting characters.
7. Signs, type indicators, strings and digit limitation formatters are not available in the Stamp I. Manual formatting will have to be done (possibly multiple DEBUG statements) to accomplish the same formatting.

Example:

```
BS2:        DEBUG DEC AByte, HEX? AWord, BIN? ANibble, CR

BS1:        DEBUG #B0, $B1, %B2, CR
```

DATA

BASIC STAMP I

EEPROM {location,}(data{,data...})

- LOCATION is in the range 0..255.
- DATA is a constant in the range 0..255. No variables are allowed.

BASIC STAMP II

{pointer} DATA {@location,} {WORD} {data}{(size)} {, { WORD} {data}{(size)}...}

- POINTER is an optional undefined constant name or a bit, nibble, byte or word variable which is assigned the value of the first memory location in which data is written.
- @LOCATION is an optional constant, expression or a bit, nibble, byte or word variable which designates the first memory location in which data is to be written.
- WORD is an optional switch which causes DATA to be stored as two separate bytes in memory.
- DATA is an optional constant or expression to be written to memory.
- SIZE is an optional constant or expression which designates the number of bytes of defined or undefined data to write / reserve in memory. If DATA is not specified then undefined data space is reserved and if DATA is specified then SIZE bytes of data equal to DATA are written to memory.

CONVERSION: BS1 ⇨ BS2

1. Replace the EEPROM directive with the DATA directive.
2. If LOCATION is specified, insert an at sign, '@', immediately before it.
3. Remove the open and close parentheses, '(' and ')'.

- The POINTER constant and WORD and (SIZE) directives may be used for added flexibility.

Example:

```
BS1:        EEPROM 100, (255, 128, 64, 92)

BS2:        DATA @100, 255, 128, 64, 92
```

CONVERSION: BS1 ⇦ BS2

1. If a POINTER constant is specified, remove it and set it equal to the value of the first location using a Stamp I assign statement.
2. Replace the DATA directive with the EEPROM directive.
3. If LOCATION is specified, remove the at sign, '@', immediately before it.
4. If the WORD directive is given, remove it and convert the data element immediately following it, if one exists, into two bytes of low-byte, high-byte format. If no data element exists immediately following the WORD directive, (the (SIZE) directive must exist) insert zero data element pairs, '0, 0,' for the number of elements given in (SIZE).
5. Add an open parenthesis, '(', just before the first data element and a close parenthesis, ')', after the last data element.
6. If the (SIZE) directive is given, remove it and copy the preceding data element, if available, into the number of SIZE data elements. If data was not given, insert SIZE data elements of zero, '0', separated by commas.

Example:

```
BS2:        MyDataPtr  DATA @100, 255, 128(2), 64, WORD 920, (10)

BS1:        SYMBOL MyDataPtr  =  100
            EEPROM MyDataPtr, (255, 128, 128, 64, 152, 3, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0)
```

BASIC Stamp I and Stamp II Conversions

DTMFOUT

BASIC STAMP I

NO EQUILEVANT COMMAND

BASIC STAMP II

DTMFOUT pin, {ontime, offtime,}[key{,key...}]

- PIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15.
- ONTIME and OFFTIME are constants, expressions or bit, nibble, byte or word variables in the range 0..65535.
- KEY is a constant, expression or a bit, nibble, byte or word variable in the range 0..15.

CONVERSION:

No conversion possible.

EEPROM (See DATA)

BASIC Stamp I and Stamp II Conversions

END

BASIC Stamp I

END

- 20uA reduced current (no loads).

BASIC Stamp II

END

- 50uA reduced current (no loads).

Conversion:

No conversion necessary.

EXPRESSIONS

BASIC STAMP I

{-} value ?? value {?? value...}

- Stamp I expressions are only allowed within an assignment statement. See the LET command for more detail.
- VALUE is a constant or a bit, byte or word variable.
- ?? is +,-,*,**,/,//,MIN,MAX,&,1,^,&/,|/,^/.

BASIC STAMP II

{?} value ?? value {?? {?} value}

- Stamp II expressions are allowed in place of almost any argument in any command as well as within an assignment statement.
- ? is SQR, ABS, ~, -, DCD, NCD, COS, SIN.
- VALUE is a constant or a bit, nibble, byte or word variable.
- ?? is +,-,*,**,*/,/,//,MIN,MAX,&,|,^,DIG,<<,>>,REV.
- Parentheses may be used to modify the order of expression evaluation.

CONVERSION: BS1 ⇨ BS2

1. Remove the LET command. This is not allowed in the Stamp II.

- VARIABLE and VALUE may be nibble variables for efficiency.
- The optional unary operator {-} may now also include SQR, ABS, ~, DCD, NCD, COS and SIN.
- The binary operators can now include */, DIG, <<, >> and REV.

Example:

```
BS1:        LET  b0 = -10 + 16

BS2:        Result = -10 + 16
```

CONVERSION: BS1 ⇦ BS2

1. Remove any unary operator other than minus (-) and modify the equation as appropriate, if possible.
2. The binary operator can not be */, DIG, <<, >> or REV.
3. VARIABLE and VALUE must not be a nibble variable.

Example:

```
BS2:        Result =  ~%0001 + 16

BS1:        b0 = %1110 + 16
```

FOR...NEXT

BASIC STAMP I

FOR variable = start TO end {STEP {-} stepval}...NEXT {variable}

- Up to 8 nested FOR...NEXT loops are allowed.
- VARIABLE is a bit, byte or word variable.
- START is a constant or a bit, byte or word variable.
- END is a constant or a bit, byte or word variable.
- STEPVAL is a constant or a bit, byte or word variable.
- VARIABLE (after NEXT) must be the same as VARIABLE (after FOR).

BASIC STAMP II

FOR variable = start TO end {STEP stepval}...NEXT

- Up to 16 nested FOR...NEXT loops are allowed.
- VARIABLE is a bit, nibble, byte or word variable.
- START is a constant, expression or a bit, nibble, byte or word variable.
- END is a constant, expression or a bit, nibble, byte or word variable.
- STEPVAL is an optional constant, expression or a bit, nibble, byte or word variable and must be positive.

CONVERSION: BS1 ⇨ BS2

1. Remove the minus sign "-" from the step value if given. The Stamp II dynamically determines the direction at run-time depending on the order of START and END. This allows for great flexibility in programming.

2. Remove the VARIABLE name after the NEXT statement if given. The variable is always assumed to be from the most recent FOR statement and is not allowed in the Stamp II.

- VARIABLE, START, END and STEPVAL may be a nibble variable for efficiency.
- Up to 16 nested FOR...NEXT statements may be used.

Example:

```
BS1:        FOR  B0 = 10 TO 1 STEP -1
   {code inside loop}
   NEXT B0

BS2:        FOR  LoopCount = 10 TO 1 STEP 1
   {code inside loop}
   NEXT
```

CONVERSION: BS1 ⇦ BS2

1. VARIABLE, START, END and STEPVAL must not be a nibble.
2. If negative stepping is to be done, a negative STEPVAL must be specified.
3. Must have no more than 8 nested FOR...NEXT loops.

Example:

```
BS2:        FOR  LoopCount = 100 TO 10 STEP 2
   {code inside loop}
   NEXT

BS1:        FOR  B0 = 100 TO 10 STEP -2
   {code inside loop}
   NEXT
```

FREQOUT

BASIC STAMP I

SOUND pin, (note, duration {,note, duration...})

- PIN is a constant or a bit, byte or word variable in the range of 0..7.
- NOTE is a constant or a bit, byte or word variable in the range of 0..255 representing frequencies in the range 94.8 Hz to 10,550 Hz.
- DURATION is a constant or a bit, byte or word variable in the range of 1..255 specifying the duration in 12 ms units.

BASIC STAMP II

FREQOUT pin, milliseconds, freq1 {,freq2}

- PIN is a constant, expression or a bit, nibble, byte or word variable in the range of 0..15.
- MILLISECONDS is a constant, expression or a bit, nibble, byte or word variable.
- FREQ1 and FREQ2 are constant, expression or bit, nibble, byte or word variables in the range 0..32767 representing the corresponding frequencies. FREQ2 may be used to output 2 sine waves on the same pin simultaneously.

CONVERSION: BS1 ⇨ BS2

1. Change command name 'SOUND' to 'FREQOUT'.
2. Remove the parentheses, '(' and ')'.
3. Swap the orientation of DURATION with NOTE and multiply DURATION by 12.
4. (MILLISECONDS = DURATION * 12).
5. Calculate FREQ1 using the formula: FREQ1 = 1 / (95 x 10^{-6} + ((127 - NOTE) * 83 x 10^{-6}).
5. Place successive NOTE and DURATION pairs into separate FREQOUT commands.

- PIN may be in the range 0..15.

Example:

```
BS1:       SOUND  1, (92, 128, 75, 25)

BS2:       FREQOUT  1, 1536, 333
           FREQOUT  1, 300, 226
```

CONVERSION: BS1 ⇦ BS2

1. Change command name ‘FREQOUT’ to ‘SOUND’.
2. PIN must be in the range 0..7.
3. Insert an open parenthesis just before the MILLISECONDS argument.
4. Swap the orientation of MILLISECONDS with FREQ1 and divide MILLISECONDS by 12. (DURATION = MILLISECONDS / 12).
5. Calculate NOTE using the formula: NOTE = 127 - ((1 / FREQ1) - 95 x 10^{-6}) / 83 x $10^{-6.}$
6. Successive FREQOUT commands may be combined into one SOUND command by separating NOTE and DURATION pairs with commas.
7. Insert a close parenthesis, ‘)’, after the last DURATION argument.

- Notes can not be mixed as in the Stamp II

Example:

```
BS2:       FREQOUT  15, 2000, 400
           FREQOUT  15, 500, 600

BS1:       SOUND  7, (98, 167, 108, 42)
```

GOSUB

BASIC STAMP I

GOSUB label

- Up to 16 GOSUBs allowed per program.
- Up to 4 nested GOSUBs allowed.
- Word W6 is modified with every occurrence of GOSUB.

BASIC STAMP II

GOSUB label

- Up to 255 GOSUBs allowed per program.
- Up to 4 nested GOSUBs allowed.

CONVERSION: BS1 ⇨ BS2

- Up to 255 GOSUBs can be used in the program.
- No general purpose variables are modified with the occurrence of GOSUB.

CONVERSION: BS1 ⇦ BS2

1. Only 16 GOSUBs can be used in the program.

- Word W6 is modified with every occurrence of GOSUB.

BASIC Stamp I and Stamp II Conversions

GOTO

BASIC STAMP I

GOTO label

BASIC STAMP II

GOTO label

CONVERSION:

No conversion necessary.

HIGH

BASIC STAMP I

HIGH pin

- PIN is a constant, expression or a bit, byte or word variable in the range 0..7.

BASIC STAMP II

HIGH pin

- PIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15.

CONVERSION: BS1 ⇨ BS2

- PIN may be a constant, expression or a bit, nibble, byte or word variable in the range 0..15.

CONVERSION: BS1 ⇦ BS2

- PIN must be a constant or a bit, byte or word variable in the range 0..7.

Example:

```
BS2:        HIGH  15

BS1:        HIGH  7
```

BASIC Stamp I and Stamp II Conversions

IF...THEN

BASIC STAMP I

IF variable ?? value {AND/OR variable ?? value...} THEN label

- VARIABLE is a bit, byte or word variable. No constants are allowed.
- ?? is =, <>, >, <, >=, <=.
- VALUE is a constant or a bit, byte, or word variable.
- LABEL is a location to branch to if the result is true.

BASIC STAMP II

IF conditionalexpression THEN label

- CONDITIONALEXPRESSION is any valid Boolean expression using the =, <>, >, <, >=, <=, conditional operators and the AND, OR, NOT, and XOR logical operators.
- LABEL is a location to branch to if the result is true.

CONVERSION: BS1 ⇨ BS2

1. If VARIABLE is PINS or PIN0..PIN7 then replace in with INS or IN0..IN7.

CONVERSION: BS1 ⇦ BS2

1. If the INS or OUTS symbol is specified to the left of the conditional operator, replace it with PINS.
2. If the logical operator NOT is specified, remove it and switch the conditional operator to negative logic.
3. If one of the values is an expression, you must perform the calculation in a dummy variable outside of the IF...THEN statement.

Example:

```
BS2:    IF NOT FirstValue > LastValue * (2 + NextValue) THEN Loop

BS1:    Temp = 2 + NextValue * LastValue
        IF FirstValue <= Temp THEN Loop
```

INPUT

BASIC STAMP I

INPUT pin

- PIN is a constant, expression or a bit, byte or word variable in the range 0..7.

BASIC STAMP II

INPUT pin

- PIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15.

CONVERSION: BS1 ⇨ BS2

- PIN may be a nibble variable in the range 0..15.

CONVERSION: BS1 ⇦ BS2

- PIN must not be a nibble variable and must be in the range 0..7 only.

Example:

```
BS2:        INPUT 15

BS1:        INPUT 7
```

LET

BASIC STAMP I

{LET} variable = {-} value ?? value {?? value...}

- VARIABLE is a bit, byte or word variable.
- VALUE is a constant or a bit, byte or word variable.
- ?? is +,-,*,**,/,//,MIN,MAX,&,1,^,&/,|/,^/.

BASIC STAMP II

variable = {?} value ?? value {?? {?} value}

- VARIABLE is a bit, nibble, byte or word variable.
- ? is SQR, ABS, ~, -, DCD, NCD, COS, SIN.
- VALUE is a constant or a bit, nibble, byte or word variable.
- ?? is +,-,*,**,*/,/,//,MIN,MAX,&,|,^,DIG,<<,>>,REV.
- Parentheses may be used to modify the order of expression evaluation.

CONVERSION: BS1 ⇨ BS2

1. Remove the LET command. This is not allowed in the Stamp II.

- VARIABLE and VALUE may be nibble variables for efficiency.
- The optional unary operator {-} may now also include SQR, ABS, ~, DCD, NCD, COS and SIN.
- The binary operators can now include */, DIG, <<, >> and REV.

Example:

```
BS1:        LET  b0 = -10 + 16

BS2:        Result = -10 + 16
```

CONVERSION: BS1 ⇦ BS2

1. Remove any unary operator other than minus (-) and modify the equation as appropriate, if possible.

2. The binary operator can not be */, DIG, <<, >> or REV.
3. VARIABLE and VALUE must not be a nibble variable.

Example:

```
BS2:        Result =  ~%0001 + 16

BS1:        b0 = %1110 + 16
```

BASIC Stamp I and Stamp II Conversions

LOOKDOWN

BASIC Stamp I

LOOKDOWN value, (value0, value1,... valueN), variable

- VALUE is a constant or a bit, byte or word variable.
- VALUE0, VALUE1, etc. are constants or a bit, byte or word variables.
- VARIABLE is a bit, byte or word variable.

BASIC Stamp II

LOOKDOWN value, {??,} [value0, value1,... valueN], variable

- VALUE is a constant, expression or a bit, nibble, byte or word variable.
- ?? is =, <>, >, <, <=, =>. (= is the default).
- VALUE0, VALUE1, etc. are constants, expressions or bit, nibble, byte or word variables.
- VARIABLE is a bit, nibble, byte or word variable.

Conversion: BS1 ⇨ BS2

1. Change all parentheses, "(" and ")", to brackets, "[" and "]"

- Any or all arguments may be nibble variables for efficiency.
- The optional ?? operator may be included for flexibility.

Example:

```
BS1:        LOOKDOWN  b0, ("A", "B", "C", "D"), b1

BS2:        LOOKDOWN  ByteValue, ["A", "B", "C", "D"], Result
```

Conversion: BS1 ⇦ BS2

1. Change all brackets, "[" and "]", to parentheses, "(" and ")".
2. Remove the "??," argument if it exists and modify the list if possible. "=" is assumed in the Stamp I.

- None of the arguments may nibble variables.

Example:

```
BS2:        LOOKDOWN  ByteValue, [1, 2, 3, 4], Result

BS1:        LOOKDOWN  b0, (1, 2, 3, 4), b1
```

LOOKUP

BASIC STAMP I

LOOKUP index, (value0, value1,... valueN), variable

- INDEX is a constant or a bit, byte or word variable.
- VALUE0, VALUE1, etc. are constants or a bit, byte or word variables.
- VARIABLE is a bit, byte or word variable.

BASIC STAMP II

LOOKUP index, [value0, value1,... valueN], variable

- INDEX is a constant, expression or a bit, nibble, byte or word variable.
- VALUE0, VALUE1, etc. are constants, expressions or bit, nibble, byte or word variables.
- VARIABLE is a bit, nibble, byte or word variable.

CONVERSION: BS1 ⇨ BS2

1. Change all parentheses, “(“ and “)”, to brackets, “[“ and “]”

- Any or all arguments may be nibble variables for efficiency.

Example:

```
BS1:        LOOKUP  b0, (1, 2, 3, 4), b1

BS2:        LOOKUP  ByteValue, [1, 2, 3, 4], Result
```

CONVERSION: BS1 ⇦ BS2

1. Change all brackets, “[“ and “]”, to parentheses, “(“ and “)”

- None of the arguments may nibble variables.

Example:

```
BS2:        LOOKUP  ByteValue, [1, 2, 3, 4], Result

BS1:        LOOKUP  b0, (1, 2, 3, 4), b1
```

LOW

BASIC Stamp I

LOW pin

- PIN is a constant or a bit, byte or word variable in the range 0..7.

BASIC Stamp II

LOW pin

- PIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15.

Conversion: BS1 ⇨ BS2

- PIN may be a constant or a bit, nibble, byte or word variable in the range 0..15.

Conversion: BS1 ⇦ BS2

PIN must be a constant or a bit, byte or word variable in the range 0..7.

Example:

```
BS2:        LOW  15

BS1:        LOW  7
```

NAP

BASIC STAMP I

NAP period

- PERIOD is a constant or a bit, byte or word variable in the range 0..7 representing 18ms intervals.
- Current is reduced to 20uA (assuming no loads).

BASIC STAMP II

NAP period

- PERIOD is a constant, expression or a bit, nibble, byte or word variable in the range 0..7 representing 18ms intervals.
- Current is reduced to 50uA (assuming no loads).

CONVERSION:

No conversion necessary.

OUTPUT

BASIC STAMP I

OUTPUT pin

- PIN is a constant or a bit, byte or word variable in the range 0..7.

BASIC STAMP II

OUTPUT pin

- PIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15.

CONVERSION: BS1 ⇨ BS2

- PIN may be a constant or a bit, nibble, byte or word variable in the range 0..15.

CONVERSION: BS1 ⇦ BS2

1. PIN must be a constant or a bit, byte or word variable in the range 0..7.

Example:

```
BS2:        OUTPUT  15

BS1:        INPUT  7
```

PAUSE

BASIC STAMP I

PAUSE milliseconds

- MILLISECONDS is a constant or a bit, byte or word variable in the range 0..65535.

BASIC STAMP II

PAUSE milliseconds

- MILLISECONDS is a constant, expression or a bit, nibble, byte or word variable in the range 0..65535.

CONVERSION:

No conversion necessary.

POT (See RCTIME)

PULSIN

BASIC Stamp I

PULSIN pin, state, variable

- PIN is a constant, expression or a bit, byte or word variable in the range 0..7.
- STATE is a constant, expression or a bit, byte or word variable in the range 0..1.
- VARIABLE is a bit, byte or word variable.
- Measurements are in 10uS intervals and the instruction will time out in 0.65535 seconds.

BASIC Stamp II

PULSIN pin, state, variable

- PIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15.
- STATE is a constant, expression or a bit, nibble, byte or word variable in the range 0..1.
- VARIABLE is a bit, nibble, byte or word variable.
- Measurements are in 2uS intervals and the instruction will time out in 0.13107 seconds.

Conversion: BS1 ⇨ BS2

- Any or all arguments may be a nibble variable for efficiency.
- PIN may be in the range 0..15.
- Returned value is 5 times less than in the Stamp I counterpart.

CONVERSION: BS1 ⇦ BS2

- None of the arguments may be a nibble variable.
- PIN must be in the range 0..7.
- Returned value is 5 times more than in the Stamp I counterpart.

Example:

```
BS2:        PULSIN  15, 1, Result

BS1:        PULSIN  7, 1, W0
```

PULSOUT

BASIC STAMP I

PULSOUT pin, time

- PIN is a constant or a bit, byte or word variable in the range 0..7.
- TIME is a constant or a bit, byte or word variable in the range 0..65535 representing the pulse width in 10uS units.

BASIC STAMP II

PULSOUT pin, period

- PIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15.
- PERIOD is a constant, expression or a bit, nibble, byte or word variable in the range 0..65535 representing the pulse width in 2uS units.

CONVERSION: BS1 ⇨ BS2

1. PERIOD = TIME * 5.

- PIN may be a nibble variable in the range 0..15.

Example:

```
BS1:      PULSOUT  1, 10

BS2:      PULSOUT  1, 50
```

CONVERSION: BS1 ⇦ BS2

1. TIME = PERIOD / 5.

- PIN must be in the range 0..7 and must not be a nibble variable.

Example:

```
BS2:      PULSOUT  15, 25

BS1:      PULSOUT  7, 5
```

PWM

BASIC STAMP I

PWM pin, duty, cycles

- PIN is a constant or a bit, byte or word variable in the range 0..7.
- DUTY is a constant or a bit, byte or word variable in the range 0..255.
- CYCLES is a constant or a bit, byte or word variable in the range 0..255 representing the number of 5ms cycles to output.

BASIC STAMP II

PWM pin, duty, cycles

- PIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15.
- DUTY is a constant, expression or a bit, nibble, byte or word variable in the range 0..255.
- CYCLES is a constant, expression or a bit, nibble, byte or word variable in the range 0..255 representing the number of 1ms cycles to output.

CONVERSION: BS1 ⇨ BS2

1. CYCLES = CYCLES * 5.

- PIN may be a nibble variable in the range 0..15.

Example:

```
BS1:        PWM  0, 5, 1

BS2:        PWM  0, 5, 5
```

CONVERSION: BS1 ⇦ BS2

1. CYCLES = CYCLES / 5.

- PIN must be in the range 0..7 and must not be a nibble variable.

Example:

```
BS2:        PWM  15, 5, 20

BS1:        PWM  7, 5, 4
```

RANDOM

BASIC STAMP I

RANDOM variable

- VARIABLE is a byte or word variable in the range 0..65535.

BASIC STAMP II

RANDOM variable

- VARIABLE is a byte or word variable in the range 0..65535.

CONVERSION: BS1 ⇨ BS2

- The numbers generated for any given input will not be the same on the Stamp II as in the Stamp I.

CONVERSION: BS1 ⇦ BS2

- The numbers generated for any given input will not be the same on the Stamp I as in the Stamp II.

RCTIME

BASIC Stamp I

POT pin, scale, bytevariable

- PIN is a constant or a bit, byte or word variable in the range 0..7.
- SCALE is a constant or a bit, byte or word variable in the range 0..255.
- BYTEVARIABLE is a byte variable.

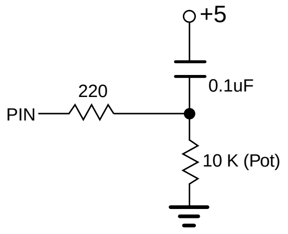
BASIC Stamp II

RCTIME pin, state, variable

- PIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15.
- STATE is a constant, expression or a bit, nibble, byte or word variable in the range 0..1.
- VARIABLE is a bit, nibble, byte or word variable.

Conversion: BS1 ⇨ BS2

1. Modify the circuit connected to PIN to look similar to the following diagram. (Note, your values for the resistors and capacitor may be different).



2. Insert two lines before the POT command as follows:

HIGH pin ; where PIN is the same PIN in the POT command.

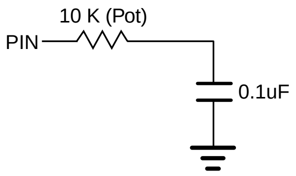
PAUSE delay ; where DELAY is an appropriate time in milliseconds to allow the capacitor to; fully discharge. You may have to try different DELAY values to find an optimal; value.

3. Change the command's name from 'POT' to 'RCTIME'.
4. Replace the SCALE argument with a STATE argument; our example requires a 1.

• PIN may be a nibble variable in the range 0..15.

CONVERSION: BS1 ⇦ BS2

1. Modify the circuit connected to PIN to look similar to the following diagram. (Note, your values for the resistor and capacitor may be different).



2. Delete the code before the RCTIME command which discharges the capacitor. This code usually consists of two lines as follows:

HIGH pin ; where PIN is the same PIN in the RCTIME command.

PAUSE delay ; where DELAY is an appropriate time in milliseconds to allow the capacitor to; fully discharge.

3. Change the command's name from 'RCTIME' to 'POT'.
4. Use the ALT-P key combination to determine the appropriate scale factor for the POT you are using as described in the BASIC Stamp I manual.

5. Replace the STATE argument with a SCALE argument.
6. Make VARIABLE a byte variable.

- PIN must be in the range 0..7 and must not be a nibble variable.

READ

BASIC STAMP I

READ location, variable

- LOCATION is a constant or a bit, byte or word variable in the range 0..255.
- VARIABLE is a bit, byte or word variable.

BASIC STAMP II

READ location, variable

- LOCATION is a constant, expression or a bit, nibble, byte or word variable in the range 0..2047.
- VARIABLE is a bit, nibble, byte or word variable.

CONVERSION: BS1 ⇨ BS2

- LOCATION and VARIABLE may be a nibble variable for efficiency.
- LOCATION may be in the range 0..2047.

CONVERSION: BS1 ⇦ BS2

- LOCATION and VARIABLE must not be a nibble variable.
- LOCATION must be in the range 0..255.

BASIC Stamp I and Stamp II Conversions

REVERSE

BASIC STAMP I

REVERSE pin

- PIN is a constant or a bit, byte or word variable in the range 0..7.

BASIC STAMP II

REVERSE pin

- PIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15.

CONVERSION: BS1 ⇨ BS2

- PIN may be a constant, expression or a bit, nibble, byte or word variable in the range 0..15.

CONVERSION: BS1 ⇦ BS2

- PIN must be a constant or a bit, byte or word variable in the range 0..7.

Example:

```
BS2:        REVERSE  15

BS1:        REVERSE  7
```

SERIN

BASIC STAMP I

SERIN pin, baudmode {,(qualifier {,qualifier...}) } {,{#} variable...}

- PIN is a constant or a bit, byte or word variable in the range 0..7.
- BAUDMODE is a constant or a bit, byte or word variable in the range 0..7 or a symbol with the following format: [T | N][2400 | 1200 | 600 | 300].
- QUALIFIERs are optional constants or a bit, byte or word variables which must be received in the designated order for execution to continue.
- VARIABLE is a bit, byte or word variable.
- # will convert ascii numbers to a binary equivalent.

BASIC STAMP II

SERIN rpin{\fpin}, baudmode, {plabel,} {timeout, tlabel,} [inputdata]

- RPIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..16.
- FPIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15.
- BAUDMODE is a constant, expression or a bit, nibble, byte or word variable in the range 0..65535.
- PLABEL is a label to jump to in case of a parity error.
- TIMEOUT is a constant, expression or a bit, nibble, byte or word variable in the range 0..65535 representing the number of milliseconds to wait for an incoming message.
- TLABEL is a label to jump to in case of a timeout.
- INPUTDATA is a set of constants, expressions and variable names separated by commas and optionally proceeded by the formatters available in the DEBUG command, except the ASC and REP

formatters. Additionally, the following formatters are available:

STR	bytearray\L{\E}	input a string into bytearray of length L with optional end-character of E. (0's will fill remaining bytes).
SKIP	L	input and ignore L bytes.
WAITSTR	bytearray{\L}	Wait for bytearray string (of L length, or terminated by 0 if parameter is not specified and is 6 bytes maximum).
WAIT	(value {,value...})	Wait for up to a six-byte sequence.

CONVERSION: BS1 ⇨ BS2

1. BAUDMODE is a constant or a bit, nibble, byte or word variable equal to the bit period of the baud rate plus three control bits which specify 8-bit / 7-bit, True / Inverted and Driven / Open output. The following table lists the Stamp I baudmodes and the corresponding Stamp II baudmode:

SERIN Baudmode Conversion

Stamp I Baudmode		Stamp II Baudmode
0	T2400	396
1	T1200	813
2	T600	1646
3	T300	3313
4	N2400	396 + $4000
5	N1200	813 + $4000
6	N600	1646 + $4000
7	N300	3313 + $4000

2. INPUTDATA includes QUALIFIERS and VARIABLES and must

be encased in brackets, "[" and "]". If QUALIFIERS are present, insert the modifier "WAIT" immediately before the open parenthesis before the first QUALIFIER.

3. Replace any optional "#" formatters with the equivalent "DEC" formatter.

- RPIN = PIN and may be in the range 0..16
- BAUDMODE may be any bit period in between 300 baud and 50000 baud and can be calculated using the following formula: INT(1,000,000 / Baud Rate) - 20.
- The optional formatter may include any formatter listed for INPUTDATA above.

Example:

```
BS1:        SERIN  0, 1, ("ABCD"), #B0, B1

BS2:        SERIN  0, 813, [WAIT("ABCD"), DEC FirstByte, SecondByte]
```

CONVERSION: BS1 ⇦ BS2

1. PIN = RPIN and must be in the range 0..7.
2. Remove the FPIN argument "\fpin" if it is specified. No flow control pin is available on the Stamp I.
3. BAUDMODE is a constant or a symbol or a bit, byte or word variable representing one of the predefined baudmodes. Refer to the BAUDMODE Conversion table above for Stamp II baudmodes and their corresponding Stamp I baudmodes. While the Stamp II baudmode is quite flexible, the Stamp I can only emulate specific baud rates.
4. Remove the PLABEL argument if it is specified. No parity error checking is done on the Stamp I.
5. Remove the TIMEOUT and TLABEL arguments if they are specified. No timeout function is available on the Stamp I; the program will halt at the SERIN instruction until satisfactory data arrives.
6. Remove the brackets, "[" and "]".

7. If QUALIFIERS are specified within a WAIT modifier, remove the word “WAIT”.
8. IF QUALIFIERS are specified within a WAITSTR modifier, replace the word “WAITSTR” with an open parenthesis, “(“. Convert the bytearray into a constant text or number sequence separated by commas if necessary (remove the length specifier “\L” if one exists) and insert a close parenthesis, “)”, immediately afterward.
9. If a variable is preceded with a DEC formatter, replace the word “DEC” with “#”.
10. Any formatter other than DEC and WAIT or WAITSTR has no direct equivalent in the Stamp I and must be removed. Additional variables or parsing routines will have to be used to achieve the same results in the Stamp I as with the Stamp II.

Example:

```
BS2:      SERIN  15, 813, 1000, TimedOut, [WAIT(“ABCD”), DEC
          FirstByte, SecondByte]

BS1:      SERIN  7, 1, (“ABCD”), #B0, B1
```

SEROUT

BASIC Stamp I

SEROUT pin, baudmode, ({#} data {, {#} data...})

- PIN is a constant or a bit, byte or word variable in the range 0..7.
- BAUDMODE is a constant or a bit, byte or word variable in the range 0..15 or a symbol with the following format: {O}[T | N][2400 | 1200 | 600 | 300].
- DATA is a constant or a bit, byte or word variable.
- # will convert binary numbers to ascii text equivalents up to 5 digits in length.

BASIC Stamp II

SEROUT tpin{\fpin}, baudmode, {pace,} {timeout, tlabel,} [outputdata]

- TPIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..16.
- FPIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15.
- BAUDMODE is a constant, expression or a bit, nibble, byte or word variable in the range 0..60657.
- PACE is a constant, expression or a bit, nibble, byte or word variable in the range 0..65535 specifying a time (in milliseconds) to delay between transmitted bytes. This value can only be specified if the FPIN is not specified.
- TIMEOUT is a constant, expression or a bit, nibble, byte or word variable in the range 0..65535 representing the number of milliseconds to wait for the signal to transmit the message. This value can only be specified if the FPIN is specified.
- TLABEL is a label to jump to in case of a timeout. This can only be specified if the FPIN is specified.

- OUTPUTDATA is a set of constants, expressions and variable names separated by commas and optionally proceeded by the formatters available in the DEBUG command.

CONVERSION: BS1 ⇨ BS2

1. BAUDMODE is a constant or a bit, nibble, byte or word variable equal to the bit period of the baud rate plus three control bits which specify 8-bit / 7-bit, True / Inverted and Driven / Open output. The following table lists the Stamp I baudmodes and the corresponding Stamp II baudmode:

SEROUT Baudmode Conversion

Stamp I Baudmode		Stamp II Baudmode
0	T2400	396
1	T1200	813
2	T600	1646
3	T300	3313
4	N2400	396 + $4000
5	N1200	813 + $4000
6	N600	1646 + $4000
7	N300	3313 + $4000
8	OT2400	396 + $8000
9	OT1200	813 + $8000
10	OT600	1646 + $8000
11	OT300	3313 + $8000
12	ON2400	396 + $C000
13	ON1200	813 + $C000
14	ON600	1646 + $C000
15	ON300	3313 + $C000

1. Replace the parentheses, “(“ and “)”, with brackets, “[“ and “]”.
2. Replace any optional “#” formatters with the equivalent “DEC” formatter.

- TPIN = PIN and may be in the range 0..16.
- BAUDMODE may be any bit period in between 300 baud and 50000 baud and can be calculated using the following formula: INT(1,000,000 / Baud Rate) - 20.
- The optional formatter may include any valid formatter for the DEBUG command.

Example:

```
BS1:        SEROUT  3, T2400, (“Start”, #B0, B1)

BS2:        SEROUT  3, 396, [“Start”, DEC FirstByte, SecondByte]
```

CONVERSION: BS1 ⇦ BS2

1. PIN = TPIN and must be in the range 0..7.
2. Remove the FPIN argument “\fpin” if it is specified. No flow control pin is available on the Stamp I.
3. BAUDMODE is a constant or a symbol or a bit, byte or word variable representing one of the predefined baudmodes. Refer to the BAUDMODE Conversion table above for Stamp II baudmodes and their corresponding Stamp I baudmodes. While the Stamp II baudmode is quite flexible, the Stamp I can only emulate specific baud rates.
4. Remove the PACE argument if it is specified. No pace value is allowed on the Stamp I.
5. Remove the TIMEOUT and TLABEL arguments if they are specified. No timeout function is available on the Stamp I; the program will transmit data regardless of the status of the receiver.
6. Replace the brackets, “[“ and “]”, with parentheses, “(“ and “)”.
7. If a variable is preceded with a DEC formatter, replace the word “DEC” with “#”.
8. Any formatter other than DEC has no direct equivalent in the Stamp I and must be removed. Additional variables or constants will have to be used to achieve the same results in the Stamp I as with the Stamp II.

Example:

BS2:	SEROUT 15, 3313, 1000, TimedOut, [“Start”, DEC FirstByte, SecondByte]
BS1:	SEROUT 7, T300, (“Start”, #B0, B1)

SHIFTIN

BASIC STAMP I

NO EQUIVELANT COMMAND

BASIC STAMP II

SHIFTIN dpin, cpin, mode, [result{\bits} { ,result{\bits}... }]

- DPIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15 specifying the data pin.
- CPIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15 specifying the clock pin.
- MODE is a constant, symbol, expression or a bit, nibble, byte or word variable in the range 0..4 specifying the bit order and clock mode. 0 or MSBPRE = msb first, pre-clock, 1 or LSBPRE = lsb first, pre-clock, 2 or MSBPOST = msb first, post-clock, 3 or LSBPOST = lsb first, post-clock.
- RESULT is a bit, nibble, byte or word variable where the received data is stored.
- BITS is a constant, expression or a bit, nibble, byte or word variable in the range 1..16 specifying the number of bits to receive in RESULT. The default is 8.

CONVERSION: BS1 ⇨ BS2

- Code such as the following:

```
SYMBOL  Value = B0           'Result of shifted data
SYMBOL  Count = B1           'Counter variable
SYMBOL  CLK = 0              'Clock pin is pin 0
SYMBOL  DATA = PIN1          'Data pin is pin 1

DIRS  =  %00000001           'Set Clock pin as output and Data pin as input

FOR  Count = 1 TO 8
   PULSOUT CLK,1             'Preclock the data
   Value = Value * 2 + DATA  'Shift result left and grab next data bit
NEXT Count
```

May be converted to the following code:

```
Value       VAR      BYTE          'Result of shifted data
CLK         CON      0             'Clock pin is pin 0
DATA        CON      1             'Data pin is pin 1

DIRS  =  %0000000000000001         'Set Clock pin as output and Data pin
as input

SHIFTIN  DATA, CLK, MSBPRE, [Value\8]
```

CONVERSION: BS1 ⇦ BS2

- Code such as the following:

```
Value       VAR      BYTE          'Result of shifted data

DIRS  =  %0000000000000001         'Clock pin is 0 and Data pin is 1

SHIFTIN  1, 0, LSBPOST, [Value\8]
```

May be converted to the following code:

```
SYMBOL  Value = B0                 'Result of shifted data
SYMBOL  Count = B1                 'Counter variable

DIRS  =  %00000001                 'Clock pin is 0 and Data pin is 1

FOR  Count = 1 TO 8
   Value = DATA * 256 + Value / 2  'Shift grab next data bit and shift right
   PULSOUT CLK,1                   'Postclock the data
NEXT Count
```

SHIFTOUT

BASIC STAMP I

NO EQUIVELANT COMMAND

BASIC STAMP II

SHIFTOUT dpin, cpin, mode, [data{\bits} {, data{\bits}... }]

- DPIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15 specifying the data pin.
- CPIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15 specifying the clock pin.
- MODE is a constant, symbol, expression or a bit, nibble, byte or word variable in the range 0..1 specifying the bit order. 0 or LSBFIRST = lsb first, 1 or MSBFIRST = msb first.
- DATA is a constant, expression or a bit, nibble, byte or word variable containing the data to send out.
- BITS is a constant, expression or a bit, nibble, byte or word variable in the range 1..16 specifying the number of bits of DATA to send. The default is 8.

CONVERSION: BS1 ⇨ BS2

Code such as the following:

```
SYMBOL  Count = B1          'Counter variable
SYMBOL  CLK = 0             'Clock pin is pin 0
SYMBOL  DATA = PIN1         'Data pin is pin 1

DIRS  =  %00000011          'Set Clock and Data pins as outputs

   B0 = 125                 'Value to be shifted out

FOR  Count = 1 TO 8
   DATA = BIT7              'Send out MSB of B0
   PULSOUT CLK,1            'Clock the data
   B0 = B0 * 2              'Shift the value left; note that this causes us
                            'to lose the value
NEXT Count                  'when we're done shifting
```

May be converted to the following code:

```
Value         VAR       BYTE                       'Value to be shifted out
CLK           CON       0                          'Clock pin is pin 0
DATA          CON       1                          'Data pin is pin 1

DIRS  =  %0000000000000011                         'Set Clock and Data pins as
                                                   'outputs

Value = 125

SHIFTOUT  DATA, CLK, MSBFIRST, [Value\8]           'Note that value is still intact
                                                   'after were done shifting
```

CONVERSION: BS1 ⇦ BS2

Code such as the following:

```
Value         VAR       BYTE                       'Value to be shifted out
CLK           CON       0                          'Clock pin is pin 0
DATA          CON       1                          'Data pin is pin 1

DIRS  =  %0000000000000011                         'Set Clock and Data pins as
                                                   'outputs

Value = 220

SHIFTOUT  DATA, CLK, LSBFIRST, [Value\8]           'Note that value is still intact
                                                   'after were done shifting
```

May be converted to the following code:

```
SYMBOL  Count = B1                                 'Counter variable
SYMBOL  CLK = 0                                    'Clock pin is pin 0
SYMBOL  DATA = PIN1                                'Data pin is pin 1

DIRS  =  %00000011                                 'Set Clock and Data pins as
                                                   'outputs

              B0 = 220                             'Value to be shifted out

FOR  Count = 1 TO 8
              DATA = BIT0                          'Send out LSB of B0
              PULSOUT CLK,1                        'Clock the data
              B0 = B0 / 2                          'Shift the value left; note that
                                                   'the value is lost after were
                                                   'done
NEXT Count                                         'shifting
```

SLEEP

BASIC STAMP I

SLEEP seconds

- SECONDS is a constant or a bit, byte or word variable in the range 1..65535 specifying the number of seconds to sleep.

BASIC STAMP II

SLEEP seconds

- SECONDS is a constant, expression or a bit, nibble, byte or word variable in the range 0..65535 specifying the number of seconds to sleep.

CONVERSION:

No conversion necessary.

SOUND (See FREQOUT)

STOP

BASIC STAMP I

NO EQUIVELANT COMMAND

BASIC STAMP II

STOP

- Execution is frozen, such as with the END command, however, low-power mode is not entered and the I/O pins never go into high impedance mode.

CONVERSION: BS1 ⇨ BS2

Code such as the following:

```
StopExecution:          GOTO StopExecution
```

May be converted to the following code:

```
StopExecution:          STOP
```

CONVERSION: BS1 ⇦ BS2

Code such as the following:

```
Quit:          STOP
```

May be converted to the following code:

```
Quit:          GOTO Quit
```

TOGGLE

BASIC STAMP I

TOGGLE pin

- PIN is a constant or a bit, byte or word variable in the range 0..7.

BASIC STAMP II

TOGGLE pin

- PIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15.

CONVERSION: BS1 ⇨ BS2

- PIN may be a nibble variable and may be in the range 0..15.

CONVERSION: BS1 ⇦ BS2

- PIN must not be a nibble variable and must be in the range 0..7.

Example:

```
BS2:        TOGGLE  15

BS1:        TOGGLE  7
```

WRITE

BASIC STAMP I

WRITE location, data

- LOCATION is a constant or a bit, byte or word variable in the range 0..255.
- DATA is a constant or a bit, byte or word variable.

BASIC STAMP II

WRITE location, data

- LOCATION is a constant, expression or a bit, nibble, byte or word variable in the range 0..2047.
- DATA is a constant, expression or a bit, nibble, byte or word variable.

CONVERSION: BS1 ⇨ BS2

- LOCATION and DATA may be a nibble variable for efficiency.
- LOCATION may be in the range 0..2047.

CONVERSION: BS1 ⇦ BS2

- LOCATION and DATA must not be a nibble variable.
- LOCATION must be in the range 0..255.

XOUT

BASIC STAMP I

NO EQUIVELANT COMMAND

BASIC STAMP II

**XOUT mpin, zpin, [house\keyorcommand{\cycles}
{, house\keyorcommand{\cycles}... }]**

- MPIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15 specifying the modulation pin.
- ZPIN is a constant, expression or a bit, nibble, byte or word variable in the range 0..15 specifying the zero-crossing pin.
- HOUSE is a constant, expression or a bit, nibble, byte or word variable in the range 0..15 specifying the house code A..P respectively.
- KEYORCOMMAND is a constant, expression or a bit, nibble, byte or word variable in the range 0..15 specifying keys 1..16 respectively or is one of the commands in the following table:

X-10 Commands

X-10 Command (symbol)	Value
UNITON	%10010
UNITOFF	%11010
UNITSOFF	%11100
LIGHTSON	%10100
DIM	%11110
BRIGHT	%10110

- CYCLES is a constant, expression or a bit, nibble, byte or word variable in the range 2..65535 specifying the number of cycles to send. (Default is 2).

CONVERSION:

No conversion possible.