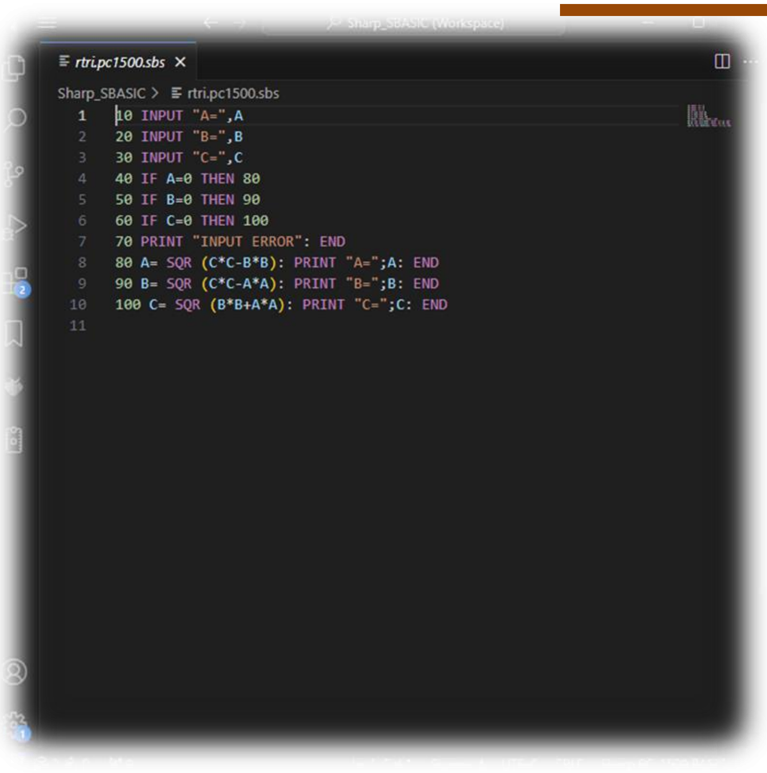
Sharp SBASIC for VS Code

Visual Studio Code Extension for the Sharp SBASIC (Hey Birt!)

User's Manual V1.0



```
10 INPUT "A=",A
20 INPUT "B=",B
30 INPUT "C=",C
40 IF A=0 THEN 80
50 IF B=0 THEN 90
60 IF C=0 THEN 100
70 PRINT "INPUT ERROR": END
80 A= SQR (C*C-B*B): PRINT "A=";A: END
90 B= SQR (C*C-A*A): PRINT "B=";B: END
100 C= SQR (B*B+A*A): PRINT "C=";C: END
```

Introduction

The Sharp SBASIC extension for Visual Studio Code adds syntax highlighting for various incarnations of SBASIC (PC-1500, PC-1600 at this time) and a suite or helper scripts to perform such tasks as tokenizing ASCII source files, detokenizing to ASCII, fixing up line endings to CR, etc.

Installation

The Sharp SBASIC extension for VS Code can be installed via the marketplace just as any other extension. It can also be installed by downloading the .vsix file from the GitHub repository (link at end of this document) for the project and installing it manually. It should be noted that additional support files, including this PDF are available from the repository that do not come as a part of the installation through the Marketplace.

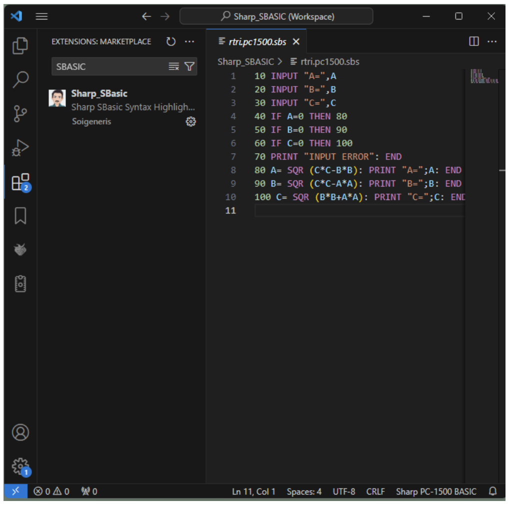
We'll first show the steps required to install the extension and then how to set things up to use it. Finally, we'll assemble an example project.

Marketplace installation

In VS Code open the Extension menu as shown and type 'SBASIOC' in the search box. Click the 'Install' button for 'Sharp SBASIC by Soigeneris.

You will then be presented with the 'readme.md' for the extension in the main window to the right. It will also show up in your list of installed extension.

You should also now be nottified when an update is available.

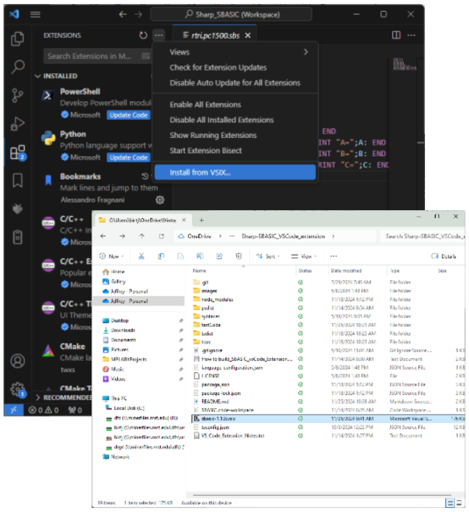


```
10 INPUT "A=",A
20 INPUT "B=",B
30 INPUT "C=",C
40 IF A=0 THEN 80
50 IF B=0 THEN 90
60 IF C=0 THEN 100
70 PRINT "INPUT ERROR": END
80 A= SQR (C*C-B*B): PRINT "A=";A: END
90 B= SQR (C*C-A*A): PRINT "B=";B: END
100 C= SQR (B*B+A*A): PRINT "C=";C: END
```

Local installation

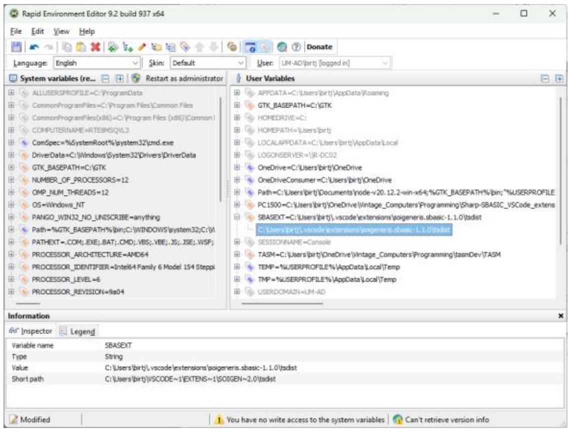
Find the 'sbasic-#.#.#.vsix' from the file set you downloaded from the GitHub repository. The "#.#.#" is version number which will change as the extension is updated.

In VS Code open the Extension menu as shown. Click the ellipsis … and then click on 'Install from VSIX' and navigate to the VSIX file you downloaded



Creating SBAS Environment Variable

To make our lives easier we will set up an Environment Variable called SBAS which points to the installation directory of the Sharp SBASIC extension. Specifically, we'll point it at the 'tsdist' folder where out helper scripts live. I like to use a free program called RapidEE for editing Windows Environment variables.



```
ALLUSERSPROFILE=C:\ProgramData
CommonProgramFiles=C:\Program Files\Common Files
CommonProgramFiles(x86)=C:\Program Files (x86)\Common I
COMPUTERNAME=RTEBMSQVL3
ComSpec=%SystemRoot%\system32\cmd.exe
DriverData=C:\Windows\System32\Drivers\DriverData
GTK_BASEPATH=C:\GTK
NUMBER_OF_PROCESSORS=12
OMP_NUM_THREADS=12
OS=Windows_NT
PANGO_WIN32_NO_UNISCRIBE=anything
Path=%GTK_BASEPATH%\bin;C:\WINDOWS\system32;C:\W
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;
PROCESSOR_ARCHITECTURE=AMD64
PROCESSOR_IDENTIFIER=Intel64 Family 6 Model 154 Steppi
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=9a04
```

```
APPDATA=C:\Users\birtj\AppData\Roaming
GTK_BASEPATH=C:\GTK
HOMEDRIVE=C:
HOMEPATH=\Users\birtj
LOCALAPPDATA=C:\Users\birtj\AppData\Local
LOGONSERVER=\\R-DC02
OneDrive=C:\Users\birtj\OneDrive
OneDriveConsumer=C:\Users\birtj\OneDrive
Path=C:\Users\birtj\Documents\node-v20.12.2-win-x64;%GTK_BASEPATH%\bin;"%USERPROFILE
PC1500=C:\Users\birtj\OneDrive\Vintage_Computers\Programming\Sharp-SBASIC_VSCode_extens
SBASEXT=C:\Users\birtj\.vscode\extensions\soigeneris.sbasic-1.1.0\tsdist
    C:\Users\birtj\.vscode\extensions\soigeneris.sbasic-1.1.0\tsdist
SESSIONNAME=Console
TASM=C:\Users\birtj\OneDrive\Vintage_Computers\Programming\tasmDev\TASM
TEMP=%USERPROFILE%\AppData\Local\Temp
TMP=%USERPROFILE%\AppData\Local\Temp
USERDOMAIN=UM-AD
```

Variable name	SBASEXT
Type	String
Value	C:\Users\birtj\.vscode\extensions\soigeneris.sbasic-1.1.0\tsdist
Short path	C:\Users\birtj\VSCODE~1\EXTENS~1\SOIGEN~2.0\tsdist

On Windows the installation folder is as shown below. On other platforms the path will be different.

```
SBAS=C:\Users\<user>\.vscode\extensions\soigeneris.sbasic-1.1.0\tsdist
```

For example, to change line endings of a text file to CR (when you are in the same folder as the source file):

```
node %SBAS%/2CR.js CRLF.txt ASC
```

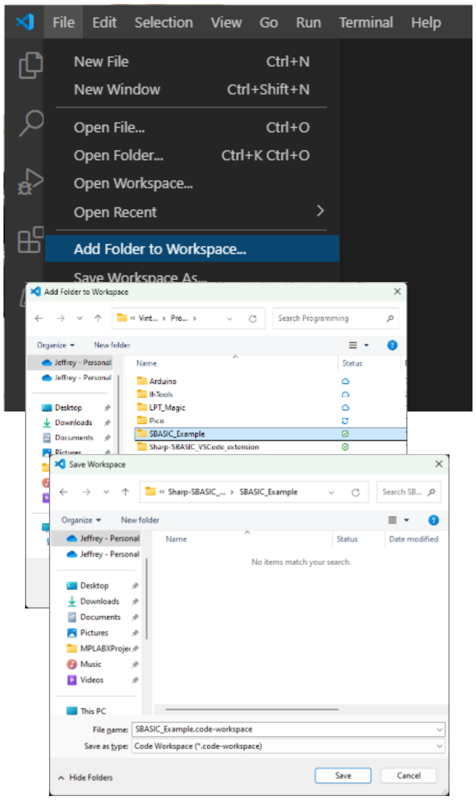
Setting up a workspace

VS Code uses the idea of a 'Workspace' to organize projects. By setting up a workspace you are telling VS Code that your project lives in a certain folder which allows VS Code to keep track of everything. VS Code will then set that folder as the Current Working Directory (CWD). Commands which are run from the VS Code terminal, including our helper scripts file we'll cover in a bit, will be run from the CWD.

Let's set up a workspace for the 'SBASIC_Example' which is included in the GitHub repository example directory structure folder set. First copy the 'SBASIC_Example' folder to a convenient location. Don't modify the original folder.

In VS Code, from the File menu close any open workspace, the select 'Add folder to Workspace' and navigate to the 'SBASIC_Example' folder copy you just made.

Next select 'File->Save Workspace As…'. I like to name my workspace with the same name as the containing folder as shown.



File name: SBASIC_Example.code-workspace
Save as type: Code Workspace (*.code-workspace)

Suggested file extensions

In an effort to make the format of a file more evident I have adopted the following conventions. Given that the Sharp SBASIC extension is PC-1500 centric at this time this works well. As other computers are added it may need to be revisited.

filename.asc -> Plain text (ASCII) BASIC
filename.bas -> Tokenized BASIC, with CE-158 header
filename.pc1500.sbs -> SBASIC source for PC-1500
filename.bin -> Binary file, no CE-158 Header
filename.mlb -> Binary file, with CE-158 header
filename.rsv -> Reserve memory file, with CE-158 header

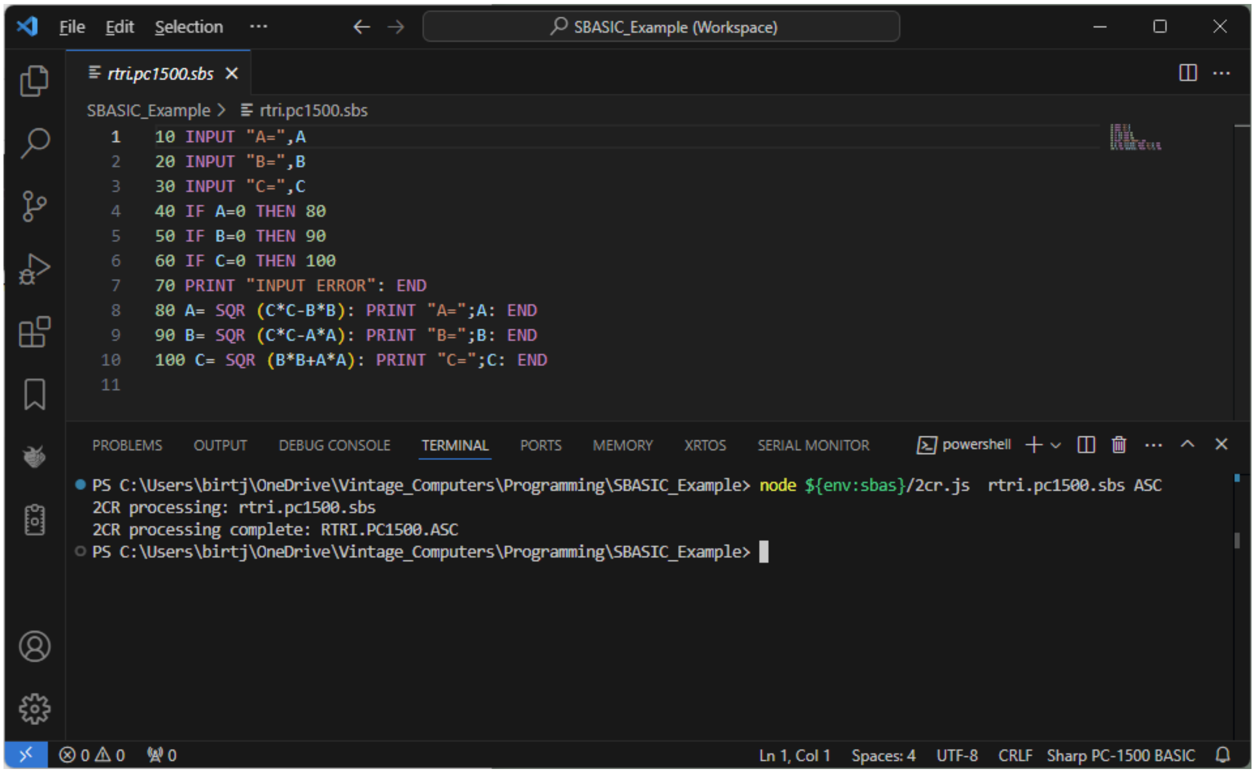
Creating a proper PC-1500 ASCII BASIC program file

The PC-1500 expects line endings (End Of Line) made up of a single CR (carriage return, 0x0D) character. Modern operating systems tend to use other EOL characters such as CRLF (0x0D 0x0A) or LF (0x0A). Some text editors can be configured as to which EOL character(s) to use but some cannot.

A descriptive file extension of '.PC1500.sbs' is used to denote a 'PC-1500 SBASIC Source' file. This is a plain text file with any EOL characters used.

To fix up the line endings to the CR that out PC-1500 expects we can use the 2cr.js script. For example, for our example program in the 'SBASIC_Example' workspace we can use the command shown below in a Powershell terminal to convert our source file to an ASC file with CR line endings. The allows us to recall the SBAS environment variable we set up previously.

```
node ${env:sbas}/2cr.js  rtri.pc1500.sbs ASC
```



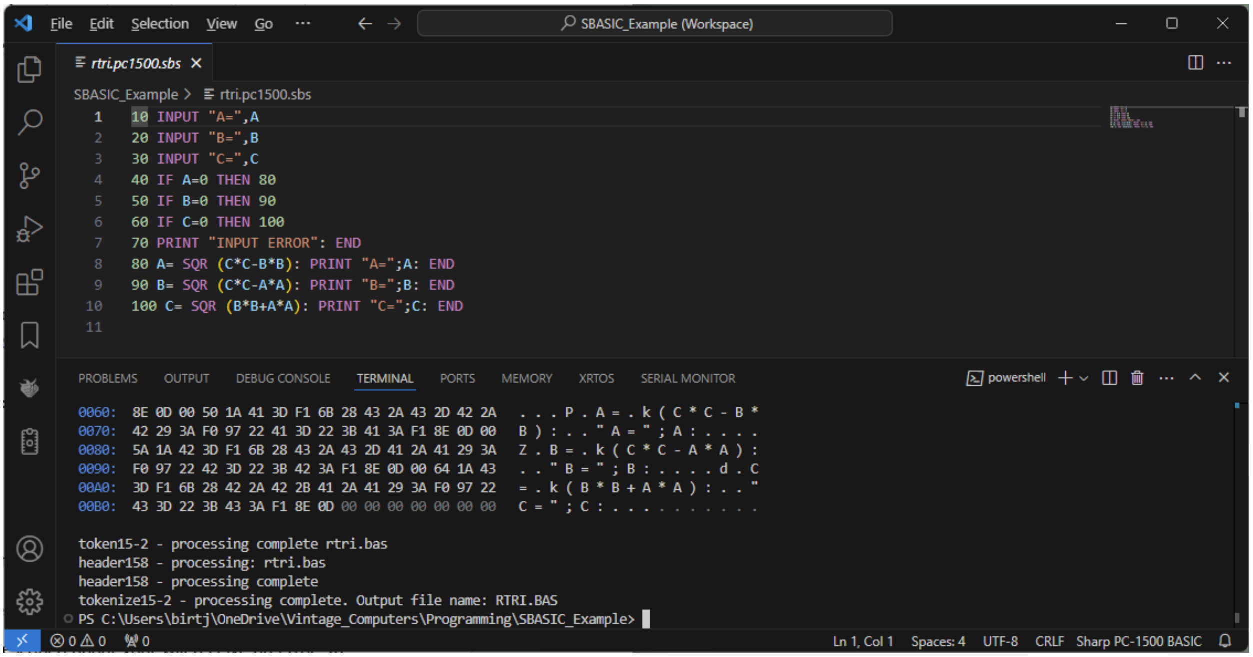
```
10 INPUT "A=",A
20 INPUT "B=",B
30 INPUT "C=",C
40 IF A=0 THEN 80
50 IF B=0 THEN 90
60 IF C=0 THEN 100
70 PRINT "INPUT ERROR": END
80 A= SQR (C*C-B*B): PRINT "A=";A: END
90 B= SQR (C*C-A*A): PRINT "B=";B: END
100 C= SQR (B*B+A*A): PRINT "C=";C: END
```

```
PS C:\Users\birtj\OneDrive\Vintage_Computers\Programming\SBASIC_Example> node ${env:sbas}/2cr.js  rtri.pc1500.sbs ASC
2CR processing: rtri.pc1500.sbs
2CR processing complete: RTRI.PC1500.ASC
PS C:\Users\birtj\OneDrive\Vintage_Computers\Programming\SBASIC_Example>
```

Creating a tokenized BASIC program (with CE-158 header)

What if we want to create a tokenized version of our BASIC program so it will load faster? We can do that with the 'tokenize15.js' helper script. This script will automatically take care of fixing up the line endings, it then tokenizes the file and adds the CE-158 header so we can load the program onto the PC-1500 using out CE-158 or CE-158X.

```
node ${env:sbas}/tokenize15.js  rtri.pc1500.sbs  rtri.bas
```



```
10 INPUT "A=",A
20 INPUT "B=",B
30 INPUT "C=",C
40 IF A=0 THEN 80
50 IF B=0 THEN 90
60 IF C=0 THEN 100
70 PRINT "INPUT ERROR": END
80 A= SQR (C*C-B*B): PRINT "A=";A: END
90 B= SQR (C*C-A*A): PRINT "B=";B: END
100 C= SQR (B*B+A*A): PRINT "C=";C: END
```

```
0060:  8E 0D 00 50 1A 41 3D F1 6B 28 43 2A 43 2D 42 2A   . . . P . A = . k ( C * C - B *
0070:  42 29 3A F0 97 22 41 3D 22 3B 41 3A F1 8E 0D 00   B ) : . . " A = " ; A : . . . .
0080:  5A 1A 42 3D F1 6B 28 43 2A 43 2D 41 2A 41 29 3A   Z . B = . k ( C * C - A * A ) :
0090:  F0 97 22 42 3D 22 3B 42 3A F1 8E 0D 00 64 1A 43   . . " B = " ; B : . . . . d . C
00A0:  3D F1 6B 28 42 2A 42 2B 41 2A 41 29 3A F0 97 22   = . k ( B * B + A * A ) : . . "
00B0:  43 3D 22 3B 43 3A F1 8E 0D 00 00 00 00 00 00 00   C = " ; C : . . . . . . . . . .

token15-2 - processing complete rtri.bas
header158 - processing: rtri.bas
header158 - processing complete
tokenize15-2 - processing complete. Output file name: RTRI.BAS
PS C:\Users\birtj\OneDrive\Vintage_Computers\Programming\SBASIC_Example>
```

The list of helper scripts

There are a number of scripts included with this extension. Most of them are directly related to Sharp SPBASIC functionality. Most are intended to be called from the command line. A few are modules which have a separate command line entry script.

A few scripts like ASCII2DEC.js, checksum.js, concat.js, reverse.js are more general purpose and were created to solve specific needs related to vintage computers. Each file starts with a brief description of how to use it.

The list of scripts:

2CR.js - Converts a text file to CR line endings
ASCIIHEX2DEC.js - Converts an ASCII encoded HEX file to decimal
bin2Bldr.js - PC-1500 BIN file to BASIC loader script
checksum.js - Generate CRC16 of binary file
concat.js - Concatenates #n binary files
detoken15.js - Command line entry to detokenize Sharp PC-1500 BASIC file
detokenize15.js – Detokenization module called by detoken15.js
hdr158.js – Module to add CE-158 header to file
reverse.js - Reverse order of a binary file
token15.js – PC1500 tokenization module called by tokenize15.js
tokenize15.js - Tokenize Sharp PC-1500 BASIC file and add CE-158 header

More to come

I'm sure many changes and additions will be made to this set of tools. For example, it might be nice to be able to choose to have a no header, CE-158 or CE-150 header added to a tokenized program. If you have suggestions just let me know.

Information and links

This is an open-source project created by Jeffrey T. Birt, a.k.a. 'Hey Birt!' Project files can be downloaded from the GitHub link below.

VS Code: https://code.visualstudio.com/
GitHub repository: https://github.com/Jeff-Birt/Sharp-SBASIC_VSCode_extension

Install Node.js: https://www.youtube.com/watch?v=eft-LJb4kW0
Typescript: https://www.youtube.com/watch?v=d56mG7DezGs

Revision history: V1.1 clarified difference between C64 PETSCII and screen codes. Thanks to lagomorph for pointing this out.