Serial Speech Synthesizer SAM

a serial port controlled text-to-speech synthesizer based on the famous software from: Don't Ask Software

User Manual

this manual is intended for firmware version:
V20211010 (October 10, 2021)

Introduction and credits

Finally it's there, a cheap (CBM userport serial (TTL-level) RS-232) based text-to-speech synthesizer. Which uses the voice of SAM (Software Automatic Mouth, the famous speech synthesizer software of the 80's). Intended mainly for the VIC-20, but there is no reason why you can not use it with your C64, C128 or Plus4. Although electrically it can be made to work with the PET series of computers, there is a problem with the kernal of the PET-series of computers and that is that they don’t support RS-232 in the same way as the VIC-20/C64/C128/Plus4. Meaning that making SSSSAM work on a PET computer does require some heavy coding, which is beyond the scope of this manual. In other words, SAM doesn’t work on a PET.

This device does not consume any RAM resources or require a SID chip. It can be controlled over a simple 2400 baud serial TTL RS-232 port (available on the userport). Making your computer speak is now a matter of sending text to this serial port, in a way that is almost similar to an ordinary PRINT statement.

If the best way to describe this device would be to compare it to existing hardware, then it perhaps should be compared to the Votrax speech synthesizer. The Votrax speech synthesizer is a serial port controlled device that can also be used in combination with the VIC-20 and the Scott Adams adventure games. And that is also the main reason for this project.

Because this device is based on the SAM speech synthesizer it's functionality is equal to that of the original SAM speech synthesizer, therefore, many parts from this manual were directly copied from the user manual of SAM, from: Don’t Ask Software. However, because this is a device and not a piece of software, there are some differences regarding its use. The main difference is the way settings need to be done. Also because some extra features and details are explained in this manual, for instance how to make SAM sing, which is a feature not very well described in the original manual.

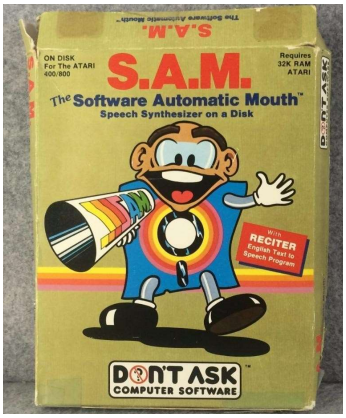
This project is a good example of how software can evolve when it reaches the open source domain. It all started with the Don't Ask Software writing the original speech synthesizer which over the years has been ported and shaped by various independent hobbyists eventually resulting in a device like this.

Although this project is named SSSSAM (which stands for: Serial Speech Synthesizer SAM), in this manual you’ll mostly find the text SAM, simply because it improves readability.

Table of Contents

1 History of SAM

1.1 What/who is SAM



SAM stands for Software Automatic Mouth. It is a speech synthesis program (no additional hardware required) developed and sold by the company Don't Ask Software. The program was released for the Apple II, Lisa, Atari 8-bit family, and Commodore 64. Released in 1982, it is the first commercial all-software voice-synthesis program. Revolutionary, considering that most speech synthesis for home computers was done using dedicated speech synthesis hardware. Of which many commercial or DIY versions were based around the famous SP0256 from General Instruments.

The Apple version of SAM prefers additional hardware that contains DACs (in other words, a soundcard), although it can instead use the computer's one-bit audio output but with the side-effect of having much distortion. The Atari makes use of the embedded POKEY audio chip. Speech playback on the Atari normally disables interrupt requests and shuts down the ANTIC chip during vocal output. The audible output is extremely distorted speech when graphic and text display is turned on. The Commodore 64 makes use of the 64's embedded SID audio chip and therefore sounds much better under all circumstances.

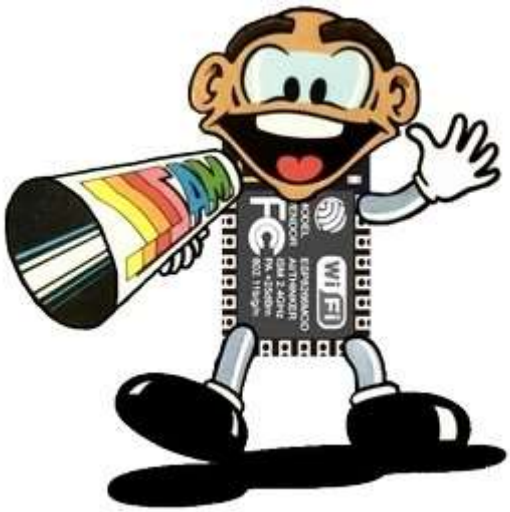
Because SAM was completely software it required some RAM from the home computer, although not very much when considering modern standards, it was sometimes too much depending on what you needed to do with the same system. If you only have 64K of RAM you must spend is wisely. But for the average BASIC programmer there was plenty of room left to use it in combination with your own BASIC programs if you wanted too. However… because it was software only, it was also copied by many people and it therefore was widely spread around the scene. Perhaps that contributed to the biggest part of it's fame. Any way, many kids had a "copy" of the program. And so did I, making my C64 speak was something amazing and fascinating and that feeling never left me. Even today when we are surrounded by perfect speech speech synthesizers, they do have lost the magic of synthesized speech. But the sound of SAM still hasn't lost its magic and instantly reminds me of the 80's.

Although for some very good reasons, it wasn't available for the VIC-20 and other Commodore models. And especially for the VIC-20, that's exactly where this project "Serial speech synthesizer SAM" can be of help. By offloading the RAM and sound requirements to "simple" and cheap dedicated hardware the VIC-20 can do speech synthesis just like the C64 but without sacrificing system resources, for a system with only 5K of RAM that's a big deal! So now your unexpanded VIC-20 can speak with the voice of SAM, just like it's big brother the C64. And the way it works makes it perfect to use it in combination with the series of Scott Adams text adventures. So if you own those text adventure cartridges you can now play then with speech synthesis which makes playing them even more fun.

This was all made possible by using the SAM software routines written by “Don't ask software”, they were ported from Assembly to C by Sebastian Macke and shaped into an ESP8266 compatible library by Earle F. Philhower who combined it with his already existing sound library. All I needed to do, was to write a wrapper around it to parse all data and settings through a TTL RS-232 serial interface. This way it allowed for easy interaction with an 8-bit computer. A small PCB was made, put into a nice box and now we can say that SAM has evolved from a home computer program to a piece of embedded software in a serial based hardware speech synthesizer. Allowing ALL 8-bit computer to speak, without expensive RAM upgrades or audio devices. All in one “simple” circuit named: SAM which stands for: Serial Speech Synthesizer SAM.

1.2 The logo

When you look at the logo of SAM (the left image), you'll instantly notice that the little guy is a just a diskette with hands, head and feet. In other words, the logo proudly shows that SAM is software. However for this project SSSSAM, that wouldn't be accurate, although it is still software, that software no longer runs on the 8-bit computer. It runs inside the modern ESP8266 micro controller. So because you now plug it into your 8-bit computer instead of loading it from disk, a logo based around a disk would be wrong. But to emphasize the fact that SSSSAM is based around the same 80's technology as SAM, the logo must remain similar. Therefore the disk-body was replaced by an ESP8266-like body. This also shows that the sound is identical, because the head of the character is identical. The feet were mirrored, to make it look more like SAM is walking in the same direction as the megaphone, just to emphasize that SAM is still alive and kicking after almost four decades.



If you look closely to the megaphone of SAM, you'll notice that the old logo shows multiple S's. Now that didn't make much sense back then, but it does so now. As the three extra S's shown now stand for Serial Speech Synthesizer, which it basically is, since it is controlled over a serial “TTL” RS-232 interface and it is a speech synthesizer.

To spice up the tiny plastic box in which SAM now houses, a logo sticker was created showing the new logo and the full name of the project. But again emphasizing the fact that this is really the sound of SAM, the speech synthesizer software of the 80's, the name SAM is shown in huge letters on the sticker. The blue lines are an ode to the style of the manuals of the Commodore product line.

2 Using SAM

2.1 VIC-20 setup

This device plugs into the user-port of your CBM computer. Below a photo of the device plugged into a VIC-20. Using the audio mixing cable SAM speaks to you through your monitor speaker.

Make sure that your computer is switched off before you plug it in, the user-port connecter requires some force to be inserted but always be careful and make sure that you insert it correctly. Considering the fact that this device connects to an old computer, always check if the contacts of the computers edge connector are clean and shiny. If they are dusty and dirty then then may require a cleanup first (for more information please refer to the trouble shooting section of this manual).

To reduce the costs and size of the device, this device does not have an integrated speaker. Therefore it can only function when connected to an external amplifier+speaker. This can be in to form of a simple battery operated walkman/discman/MP3 active speaker OR a set of heavy duty PC speakers. But the best way is perhaps as shown in the image above, using the speaker of the computer monitor by using the audio mixing cable.

The audio mixing cable it the preferred option, as it mixes the audio from the CBM computer with the audio of the speech synthesizer. Making it all sound much more “natural” because all the sound (SAM and the VIC-20) comes from the same speaker.

A great benefit of NOT having an integrated speaker (apart from the size and cost reduction) is the fact that you don't need an additional power supply to feed it. Additional power supplies could have all sorts of other issues for example different countries use different power plugs, ground potential differences, noise injection, etc. Drawing speaker power from your CBM computers power supply isn't really an option considering a speaker can draw a lot of current. It would not be wise to draw this current from the CBM computer simply because these power supplies were not designed for the purpose of feeding external devices, Commodore power supplies are running quite hot without an extra load already. But all those problems don't exist when using the audio mixing cable solution, keeping you computer cool and your desk less crowded.

More information regarding the audio mixing cable is in the “Technical info” section of this manual.

2.2 C64 setup

Regarding the C64 everything is exactly the same as it would be for the VIC-20. Please refer to that section for more details.

2.3 C128 setup

Regarding the C128 everything is exactly the same as it would be for the VIC-20. Please refer to that section for more details.

2.4 Plus4 setup

Regarding the Plus4 almost everything is exactly the same as it would be for the VIC-20. Please refer to that section for more details. However, there is one important difference and that is in the usage of the OPEN command. The OPEN command is required to properly configure the baudrate of the RS-232 serial port but the Plus4 uses different values for opening/configuring the serial port. Below an example of the difference. The 'x' in this example could be any value and is therefore not relevant. Please focus on the CHR$ part of the example below:

```
OPEN x,2,x,CHR$(10)
```

replace the CHR$(10) related text with the CHR$(26)+CHR$(5), so that the line will look like:

```
OPEN x,2,x,CHR$(26)+CHR$(5)
```

A simple two line BASIC example to make SAM speak on a Plus4 would then look like:

```
10 OPEN 1,2,3,CHR$(26)+CHR$(5):REM OPEN SERIAL PORT @ 2400Baud
20 PRINT#1,"HELLO WORLD":REM SPEAK TEXT (NO OUTPUT TO SCREEN)
```

As long you as you keep this trivial information in mind, you will have no problem in making SAM function on a Plus4. In order to prevent confusion, this information is only shown here.

2.5 PET series setup

Attention:

Because of the absence of kernal support for the RS-232 defined IO located on the userport of the PET computer, it is not possible to control SAM using a few BASIC commands. Therefore the simplicity of the concept does no longer apply to the PET series of computers. Although this could be made to work in a slightly different way, it currently is beyond the scope of this project.

Meaning that the SAM does not work on a PET computer!

For the experienced programmer(s) who does not fear in writing their own routines for handling the low-level RS-232 related signals and a way to process the data from the BASIC interface (most likely in the form of a wedge) the information below might be of use in order to connect the device to the PET computer.

In order to use SAM on a PET computer some small modifications need to be made to the PCB, this because the user-port of the PET computer does not supply +5V. This means that a 5V power supply connection must be made to the PET computer using an additional port, the cassette-port might be an option.

What needs to be done is that on the PCB of SAM, the 5V connection (which is made using a solderbased jumper connection) needs to be altered so that power can come from a set of wires (for which already two connections are reserved on the PCB) can be soldered to. The 5V connection needs to be safe and stable, do not use a cheap 5V power supply from a phone charger, as these might have some issues regarding capacitive coupling with the mains. It therefore is best to “steal” the 5V power from one of the IO-ports of the PET computer. The cassette-port is the best option.

Please refer to the chapters “schematic” and “userport connections” for more info regarding this subject. If you do not feel confident about this modification, please ask a friend with a decent amount experience regarding the modification of electronic circuitboards.

2.6 Make it speak

The SAM speech synthesizer is by default a text-to-speech synthesizer, so you can send it ASCII text directly and it will attempt to pronounce that. Just two lines of basic code can be enough to make your CBM computer speak. For example on a VIC-20, C64 or C128 you would need to type:

```
10 OPEN 1,2,3,CHR$(10):REM OPEN SERIAL PORT @ 2400Baud
20 PRINT#1,"HELLO WORLD":REM SPEAK TEXT (NO OUTPUT TO SCREEN)
```

Make sure that there is no space(s) in the `PRINT#1` command, otherwise a syntax error will be the response of your CBM computer.

Everything you want to send to the SAM speech synthesizer is done through the PRINT# command. It doesn't matter if it is text, phonemes or settings. The PRINT# statement is the most practical way to send data over the serial port connected to the SAM speech synthesizer. The example above was a bit static and may not be desired for speaking large amounts of text, in some cases it therefore may be preferred to use data statements, as shown below. As data statements have the power to give your program more structure and add all other sorts of values that may come in handy when writing a text adventure for example.

```
10 OPEN 1,2,3,CHR$(10):REM OPEN SERIAL PORT
20 READ A$:REM READ DATA FROM TABLE
30 IF A$="*" THEN 99
40 PRINT#1,A$:REM SPEAK TEXT
50 PRINT A$:REM ALSO PRINT TO SCREEN
60 FOR L=0 TO 2000:NEXT L:REM DELAY
70 GOTO 20
99 END
100 DATA "THIS IS JUST AN EXAMPLE"
110 DATA "OF USING DATA FROM A TABLE"
120 DATA "*":REM END OF DATA
```

Now this example is still very simple in what it does. Because we use a very large delay to prevent SAM from overwhelming it with text. If we could only check if SAM is ready for more data... well, we can. By checking for the OK response. Or actually for the <RETURN> (which is the CHR$(13)) that comes after it. This way we have some interaction between our basic program and SAM, instead of sending it text when we think it is ok, we will only be sending SAM text when we know it is ok. Which is exactly when SAM has finished speaking the requested text. The following example does that. SAM will recite the poem “Mary had a little lamb”.

```
10 OPEN 1,2,3,CHR$(10):GOSUB 900
20 READ A$:IF A$="*" THEN 99
21 PRINT A$
22 PRINT#1,A$:GOSUB 910
25 GOTO 20
99 END
101 REM DATA TABLE
102 REM ----------
103 DATA"MARY HAD A LITTLE LAMB."
104 DATA"IT'S FLEECE AS WHITE A SNOW."
106 DATA"EVERYWHERE THAT MARY WENT."
107 DATA"THE LAMB WAS SURE TO GO."
108 DATA"IT FOLLOWED HER TO SCHOOL ONE DAY."
109 DATA"WHICH WAS AGAINST THE RULES."
110 DATA"IT MADE THE CHIL-DREN LAUGH AND PLAY."
111 DATA"TO SEE THE LAMB AT SCHOOL."
112 DATA"AND SO THE TEACHER TURNED IT OUT."
113 DATA"BUT STILL IT LING-CURD NEAR."
114 DATA"WHY DOES THE LAMB LOVE MARY SO?"
115 DATA"THE EAGER CHIL-DREN CRY."
116 DATA"WHY, MARY LOVES THE LAMB, YOU KNOW."
117 DATA"THE TEACHER DID RE-PLY."
199 DATA"*"
900 REM CLEAR SERIAL BUFFER
901 GET#1,A$:IFA$<>""       THEN 900
902 RETURN
910 REM WAIT FOR OK<RETURN>
911 GET#1,A$:IFA$<>CHR$(13) THEN 910
912 RETURN
```

Some words are written slightly different than you might expect, for example the word “children” is split into chil-dren. And “ling-curd” instead of lingered. This is the easiest way to enforce a better pronunciation because SAM seems to struggle with some words when written normally.

The choice for this poem is based on it’s historical significance, as it was the first recorded text spoken by Thomas A. Edison on his new invention, the tinfoil phonograph, in 1877.

2.7 Configuration

SAM is a speech synthesizer with many functions, it allows you to set various settings to make SAM sound differently. Below is a table of the commands. In order to use these commands simply specify them after the keyword -CONFIG, for example:

To set the speed to a value of 100 use the following command:

```
-CONFIG SPEED 100
```

Now you can't just type this in on your CBM, it doesn't work that way, you'll need to send this command (which is nothing more then plain text) to the serial port using the following method. First open the serial port (you may leave this out if it is already open)

```
OPEN 1,2,3,CHR$(10)
```

Then when the port is opened, you may send the command:

```
PRINT#1,"-CONFIG SPEED 100"
```

If more commands or text needs to be send, just use more PRINT# statements, since the serial port is already open. You may close the port when done, but this isn't really required. Just make sure to close not to soon, otherwise the serial port get's closed before the message has been send and nothing will happen.

On the next page are many more configuration commands. If for some reason you want to restore the settings back to the default (factory) values, this this is how you can do that:

```
PRINT#1,"-CONFIG DEFAULT"
```

And in order to make these changes permanent, you must send the “save” command:

```
PRINT#1,"-CONFIG SAVE"
```

And if you have changed some settings, but haven't saved them yet, then you can undo these changes bu sending the “load” command:

```
PRINT#1,"-CONFIG LOAD"
```

Then there is the DEBUG command, this command allows you to let SAM speak regarding the changes you make, by hearing SAM say that you've changed a setting, you have confirmation about SAM having received these new settings. This is very useful when you are playing with the settings.

```
PRINT#1,"-CONFIG DEBUG 1"
```

Or if you don't want SAM to do that, you can disable that function using the value 0:

```
PRINT#1,"-CONFIG DEBUG 0"
```

Below is a list of all recognized configurations commands and their function.

Configuration commands			
Command	**Range**	**Default**	**Function**
SPEED	0-255	72	0 = very fast 255 = very slow
PITCH	0-255	164	0 = very high 255 = very low
THROAT	0-255	128	
MOUTH	0-255	128	
PHONETIC	0-1	0	0 = disabled (SAM is in TEXT-mode) 1 = enabled (SAM is in phoneme mode)
SINGMODE	0-1	0	0 = disabled (SAM is in normal speech mode) 1 = enabled (SAM is in sing mode, tone bending is disabled)
PAUSE	0-255	30	This command allows you to add an additional pause after each sentence. The pause time is the value times 10msec. Meaning that a value of 1 results in an additional pause of 10msec after each senstence. And a value of 255 adds 255*10msec = 2.55seconds of pause after each senstence.
MSG1	text	ready.	This command allows you to change the power-on message. Max. length is 80 characters.
MSG2	text	ready.	This command allows you to change the reset message. Max. length is 80 characters.
DEBUG	0-1		0 = disabled (SAM does not say the configuration values when set) 1 = enabled (SAM pronounces all entered settings)
DEFAULT	n.a.	n.a.	This command restores all settings back to the default (factory) settings but does not save them, a manual SAVE command is required to make these changes permanent.
LOAD	n.a.	n.a.	Load the settings as stored in memory (this is also done automatically after power-on or reset)
SAVE	n.a.	n.a.	Save the current settings to internal memory
SHOW	n.a.	n.a.	This returns all the settings in readable text strings
FILEBROWSER	n.a.	n.a.	<see chapter filesystem>
UPDATE	n.a.	n.a.	<see chapter updating firmware (webbrowser)>
SSID	n.a.	n.a.	<see chapter updating firmware (webbrowser)>
PASS	n.a.	n.a.	<see chapter updating firmware (webbrowser)>

2.8 Show current settings

If you want to know what all SAM settings or firmware revision are, use the following code:

```
10 PRINT"[SHIFT+CLRHOME]SETTINGS:";CHR$(14)
20 OPEN 1,2,3,CHR$(10)
30 PRINT#1,"-CONFIG SHOW"
40 GET#1,A$:IF A$="" THEN 40
50 PRINT A$;:GOTO 40
```

Line 10: will clear the screen, print the text “settings” and then switches the charset to lower case. Which is required to make displaying of uppercase characters possible.

Line 20: open the serial port and line

Line 30: will send the -CONFIG SHOW command to SAM, which will respond with the settings in clear and readable text over the serial port

Line 40-50: print the text received over the serial port to the screen of your CBM.

This program can only be stopped by pressing RUN STOP or RUN STOP + RESTORE.

2.9 OK-message

When SAM has finished processing the requested command (no matter is it has succeeded or failed processing that command), it will send an "OK" message (<CR> O K < CR>) to indicate that it is ready for a new command. This is useful because a program can use this to check if SAM is done speaking. Very important if you want SAM to speak a lot of text OR if you want SAM to sing.

Therefor if you need to detect this "OK" response in your program, then check for the following combination of 3 bytes: 79, 75, 13 (79=O, 75=K, 13=CR a.k.a. RETURN). Checking for OK can be as simple as calling a subroutine. The program remains in the loop until OK is received. An example of such a routine is shown below:

```
910 REM WAIT FOR OK
911 GET#1,A$:IF A$<>"O" THEN 911
912 GET#1,A$:IF A$<>"K" THEN 912
913 GET#1,A$:IF A$<>CHR$(13) THEN 913
914 RETURN
```

The above subroutine gets a character from the serial port, compares it with the expected character, and if it matches it will proceed. The routine checks for the letter 'O' the letter 'K' and the RETURN character. Now technically, and practically, you do not really need to check for the 'O' and the 'K'. Actually that is a bit of a waste of programming space. Your program works just as fine if you only check for the RETURN character, CHR$(13). So if you want to keep your program simple, just leave out line 911 and 912.

Always keep in mind that the OK message could be proceeded by all sorts of characters because of a previous command, or some previous OK messages might already be in the buffer but haven't been read yet. Sometimes you just don't know. Therefore you can not rely on the fact that the buffer is empty. Therefore, the best way to begin, is to clear the serial buffer completely. The subroutine attempts to do that, it simply reads all characters that are in the buffer:

```
900 GET#1,A$:IF A$<>"" THEN 900
901 RETURN
```

But another way would be to close the serial port and than opening it again. As this action will reset all pointers of the buffer. It is even faster (depending on how full the buffer actually is).

```
900 OPEN 1,2,3,CHR$(10), CLOSE 1, OPEN 1,2,3,CHR$(10)
901 RETURN
```

The OK message, is the only form of handshaking available, but in practice it is more then enough. Because SAM is perfectly capable of receiving new data while processing the current data, this because of the serial port buffer that stores all incoming data for a time when SAM is ready for it. In many cases the serial buffer is large enough to hold a few sentences. So unless you are planning on letting SAM recite poems or other forms of long uninterrupted text then you do not need to worry about checking for "OK" messages from SAM, to see if it SAM is ready for a new task.

2.10 Welcome message

SAM is a friendly little speech synthesizer and it likes to inform you that it's ready for use. It does this by speaking a welcome message. It can speak two message(s), both on a different event:

MSG1

event: POWER-ON (cold-reset) message (spoken when the computer is switched on)

default text: "Ready"

MSG2

event: Warm-reset message (spoken when the computer is reset using it's reset button)

default text: "Ready."

Now these messages aren't really interesting to listen to, but they confirm that SAM is working and that it is ready to be used. But it also adds to the computing experience. As your system now talks to you on power-up, how fun is that?

But wouldn't it be nice if you could change the welcome message. So that you have control over what SAM says when you turn your computer on, or when you press reset. This way you can really personalize your computer system, greeting you when it is powered on. You can do this using the following command:

```
OPEN 1,2,3,CHR$(10)
PRINT#1,"-CONFIG MSG1 GREETINGS PROFESSOR FALKEN. SHALL WE PLAY A GAME?"
PRINT#1,"-CONFIG MSG2 RESET PRESSED."
```

In order to make this change effective you must save it to the configuration memory with the save command:

```
PRINT#1,"-CONFIG SAVE"
```

But if you, for whatever reason, don't like SAM to talk to you for instance when you press reset. Then you must define an empty message. This keeps SAM silent during reset. For example:

```
PRINT#1,"-CONFIG MSG2"
```

In order to make this change effective you must save it to the configuration memory with the save command:

```
PRINT#1,"-CONFIG SAVE"
```

Unfortunately, many Commodore computer lack a reset button, so in many cases the computer is reset by turning the power briefly off. This isn't the best way to reset a system, because it increases the wear on your power switch and the additional surges of the power supply rush-in currents may cause for additional aging of some of the electronic parts. Also keep in mind that when using a PET computer,

this method of “resetting” also turns the monitor briefly off and on. So it is best to add or install a reset switch. How to do that is not within the scope of this manual.

Because SAM is connected to the reset signal of your computer it resets whenever your computer resets. But regarding reset, don’t press that reset button to soon after SAM has spoken it's welcome message. Because SAM get’s annoyed pretty quickly, feel free to try it out...

3 Speech synthesis (what is it?)

Many “synthesizers” found in consumer products, such as talking televisions or microwave ovens, use a "speech compression" technique of one sort or another or just playback a sample from a large piece of memory. These techniques require a person to speak the needed words or entire sentences. The speech waveform is then "compressed" using a mathematical algorithm and, as a result, can then be stored in a memory chip without taking up a lot of room. The synthesizer's job is to then take this compressed speech information and expand it back into the original waveform. Some of these systems work quite well, retaining the speaker's intonation and sometimes even his or her identity. The processes used in such synthesizers differ greatly from those used in unlimited vocabulary synthesizers like SAM.

Let's follow the evolution of an unlimited vocabulary speech synthesizer. First, we must define the task. Simply, we want to create a system that will synthesize any English utterance. One way to begin would be to record every possible utterance on tape and just play back the right one whenever we need it. This would take up more tape or computer memory than could ever exist, so this method is obviously not too practical.

The next method might be to record all the English words and play them back in a specific order to create sentences. This is certainly practical. It would take up a large amount of memory, but it would work. However, we have lost something in this process. The words now sound disjointed because we have "spliced" the sentence together. Also, the stress or inflection pattern of the sentence is either wrong or non-existent. If we wanted an accurate stress pattern, we would need to record every word in a number of different styles, at different pitches, etc.

Such a system needs too much memory. So, let's break things down even further and try to store as little as possible in memory. Instead of storing sentences or words or even syllables, we could store phonemes. Phonemes are the atoms of spoken language, the individual speech sounds. It turns out that English has a little over forty of them. Wow, this takes up practically no memory at all! We could specify the phonemes in the order we need to create words and sentences and really have ourselves a system. So, we go and record the phonemes and play them back to say the sentence, "I am a computer." Why can we barely understand it? It seems we have broken things down a bit too far. When we chop the words down to this level and then try to reassemble them, everything that blends one sound into another is lost and the results are nothing less than horrible.

But all is not lost, Our efforts are not wasted because we have the acoustic phonetician to come to our rescue. These people deal in the study of speech sounds and they can tell us just how to repair our phoneme-based system. First, instead of recording the actual speech waveform, we only store the frequency spectrums. By doing this, we save memory and pick up other advantages. Second, we learn that we need to store some data about timing. These are numbers pertaining to the duration of each phoneme under different circumstances, and also some data on transition times so we can know how to blend a phoneme into its neighbors. Third, we devise a system of rules to deal with all this data and, much to our amazement, our computer is babbling in no time.

The advantages in synthesizing speech in this way are tremendous. We use very little memory for all the data and the rules to use that data, and we also gain the ability to specify inflection, timing, and intonation. This is because we have not stored actual speech sounds, only their spectrums. (You can think of this as a printer needing only four colors of ink to reproduce all the colors in a picture.)

Now, in actuality, we do not store all the spectrums, but only those that are targets. Each phoneme has associated with it a target spectrum which can be specified with very little data. The target may be

thought of as a "frozen" speech sound, the sound you would be making if your mouth was frozen exactly in the middle of pronouncing the phoneme. The timing rules tell the synthesizer how to move from target to target in a manner that imitates the timing of a human talker.

SAM is this type of synthesizer implemented entirely in software. It has the tables of phoneme spectra and timing, together with the rules for using this data to blend the sounds together into any English utterance we may have in mind. We have traded some quality from the method using all the recorded words, but what we have gained is versatility, practicality, and the ability to do it all in real time, with very little memory usage, on an inexpensive microcomputer.

The original SAM software used a program called “reciter”, it is an English text-to-speech program that converts ordinary text into phonemes that SAM can understand. You simply supply output strings of 256 characters or less to the program. Reciter takes care of the rest. This program is already fully integrated into the speech synthesizer module and will be automatically used for all text based input.

The reciter program uses about 450 rules to convert English into SAM's phonetic language. Included among these rules are some stress markers for situations where the stress choice is unambiguous. In addition, SAM's usual punctuation rules still operate with some additional symbols ("!", ";", and ":") being considered as periods. The net result is that even directly-translated English text has a fair amount of inflection.

It also recognizes a number of special characters. Numbers are read aloud, and several others are pronounced as well. If a character is not understood by reciter, it simply isn't pronounced.

For situations where the highest quality speech with full inflection is desired, reciter isn't the best solution. For those situations we urge you to use SAM's phonetic system, which is also embedded into the device and can be enabled by selecting the phoneme mode. Though this mode is much more difficult to use as it requires you to spell out the sounds that should be pronounced. But don't be discouraged, as you'll find that “reciter” will do a better job of speaking from English text than other early 80's text-translator products.

4 The effects of punctuation

SAM understands four punctuation marks. They are the hyphen, comma, period, and question mark.

The hyphen “-” serves to mark clause boundaries by inserting a short pause in the speech. It also has other uses to be discussed later.

The comma “,” marks phrase boundaries and inserts a pause approximately double that of the hyphen.

The question-mark””? and period “.” mark the end of sentences, The period inserts a pause and also causes the pitch to fall. The question-mark also inserts a pause, but it causes the pitch to rise. Notice that not all questions should end with a question mark (rising pitch), only those that require a yes-or-no answer. ("Are we hiking today?" rises; "Why are we going to the woods?" falls at the end and should be marked with a period).

5 Phonetic input to SAM

5.1 The phonetic spelling system

SAM is equipped with a version of the easy-to-learn, very readable International Phonetic Alphabet. But “easy” is a relative term and some phoneme lookup tables, to help you find your way, are included in this manual. There are about fifty phonemes which will let you spell all the words in English. Some sounds from foreign languages are not available in the system so you need to find a way to work around that yourself. Consider this as SAM having a heavy English accent when speaking other languages.

Why use the phonetic system? Basically there are two compelling reasons.

1) In the phonetic system, all the words will be pronounced “correctly” (meaning spoken as specified)

2) You can put inflection into the speech however and wherever you want it

If you have already tried the RECITER text-to-speech program, you know that it does a fair job of pronouncing English words. However, it does make mistakes. Some words sound a little strange and others are difficult to understand. The reasons for this are not hard to understand. English is a language of exceptions rather than rules; words that are spelled alike are pronounced differently ("have" vs. "gave"). A rule system like RECITER cannot pronounce all words correctly unless it stores an enormous dictionary that takes up vast amounts of memory. But the second flaw in text-to-speech conversion is more serious. Such a rule system cannot decide where the stress belongs in what is being said. The phonetic system in SAM, on the other hand, allows you to decide where to accent syllables within a word and where to stress words within a sentence.

So it is clear that the preferred way to make SAM speak is with the phonetic alphabet. But how hard is it to use? It's really easier than writing in English because you don't have to know how to spell! You only have to know how to say the word in order to spell it phonetically.

Here is the complete list of phonemes, each presented with a sample word containing its sound. Note that there are many vowels, which is why they are all indicated by two letters rather than one.

The phonemes are classified into two categories: vowels and consonants. Among the vowels are the simple vowel sounds such as the "i" in "sit", the "o" in "slot", and the "a" in "hat". These vowels do not change their quality throughout their duration. There are also vowels called diphthongs such as the "i" in "site", the "o" in "slow", and the "a" in "hate", as well as the "oi" in "oil" and the "ow" in "how". These vowels start with one sound and end with another (e.g. "oi" glides from an "oh" sound to an "ee" sound).

The consonants are also divided into two groups: voiced and unvoiced. The voiced consonants require you to use your vocal chords to produce the sound. Such sounds as "b". "I", "n", and "z" sounds fall into this category. The unvoiced consonants, on the other hand, are produced entirely by rushing air and include such sounds as the "p", "t", "h", and "sh" sounds.

The example words have the **sound** of the phoneme, not necessarily the same letters.

VOWELS	
IY	f**ee**t
IH	p**i**n
EH	b**e**g
AE	S**a**m
AA	p**o**t
AH	b**u**dget
AO	t**a**lk
OH	c**o**ne
UH	b**oo**k
UX	l**oo**t
ER	b**ir**d
AX	g**a**llon
IX	d**i**git
DIPTHONGS	
EY	m**a**de
AY	h**igh**
OY	b**oy**
AW	h**ow**
OW	s**low**
UW	cr**ew**

VOICED CONSONANTS	
R	**r**ed
L	**all**ow
W	**a**way
WH	**wh**ale
Y	**y**ou
M	Sa**m**
N	ma**n**
NX	so**ng**
B	**b**ad
D	**d**og
G	a**g**ain
J	**j**u**dg**e
Z	**z**oo
ZH	plea**s**ure
V	se**v**en
DH	**th**en
UNVOICED CONSONANTS	
S	**S**am
SH	fi**sh**
F	**f**ish
TH	**th**in
P	**p**oke
T	**t**alk
K	**c**a**k**e
CH	spee**ch**
/H	a**h**ead

SPECIAL PHONEMES	
UL	settl**e** (= AXL)
UM	astron**om**y (= AXM)
UN	functi**on** (= ASN)
Q	kitt-en (glottal stop)

The following symbols are used internally by some of S.A.M.'s rules, but they are also available to the user.

YX	diphthong ending
WX	diphthong ending
RX	R after a vowel
LX	L after a vowel
/X	H before a non-front vowel or consonant
DX	"flap" as in pi**t**y

Note: The symbol for the "H" sound is /H. A glottal stop is a forced stoppage of sound.

SELDOM-USED PHONEME COMBINATIONS

Phoneme Combination	You probably want:	Unless it splits syllables like:
GS	GZ e.g. ba**gs**	bu**gs**pray
BS	BZ e.g. slo**bs**	o**bs**cene
DS	DZ e.g. su**ds**	Hu**ds**on
PZ	PS e.g. sla**ps**	--
TZ	TS e.g. cur**ts**y	--
KZ	KS e.g. fi**x**	--
NC	NXG e.g. s**inging**	**ing**rate
NK	NXK e.g. ba**nk**	Su**nk**ist

The table "Alphabetically sorted list of SAM's phonemes" on the next page, is a very practical table. (source: https://sites.google.com/site/h2obsession/CBM/C128/SAM-128/alpha-long)

Alphabetically sorted list of SAM's phonemes									
Letter(s)	**Phoneme**	**Example(s)**		**Letter(s)**	**Phoneme**	**Example(s)**		**Letter(s)**	**Phoneme**	**Example(s)**
ə (ambiguous)	AX	fall**e**n, gall**o**n		**igh**	AY	m**igh**t		**ti**	SH	mo**ti**on
a	AE	S**a**m		**io**	AYAA	**io**n		**u**	AH	b**u**g
	EY	m**a**de		**io**	AYAX	l**io**n, b**io**logy			YUW	h**u**ge
ae	EY	sund**ae**			AYOH	b**io**mass, d**io**de		**ua**	AA	g**ua**rd
	IY	p**ae**an, c**ae**sium		**ir**	ER	b**ir**d			UWAX	d**ua**l
	AY	m**ae**stro		**j**	J	**J**ew			UWAE	d**ua**lity
ai	EY	m**ai**n		**k**	K	**k**itchen		**ua**	UWAA	n**ua**nce
	EH	h**ai**r			KX	ne**ck**lace			WOH	q**ua**rt
al	AO	t**al**k		**l**	L	**l**ong			WEY	q**ua**ke
ao	EYAA	ch**ao**s			LX	c**all**		**uai**	WEY	q**uai**nt
au	AO	c**au**se		**m**	M	**m**ay		**ue**	(Y)UW	d**ue**, h**ue**
augh	AO	c**augh**t		**n**	N	**n**ot			WEH	q**ue**st
ay	EY	m**ay**		**ng**	NX	so**ng**, runni**ng**			WIX	q**u**ery
b	B	**b**ad		**o**	AA	p**o**t		**uee**	WIY	q**uee**n
c	K	**c**ome			OH	c**o**ne		**ui**	WIH	q**ui**ck
	S	re**c**eive, **c**ede			OW	n**o**			WAY	q**ui**te
ch	CH	**ch**ew			UW	d**o**, wh**o**			UWIH	t**ui**tion
	K	**ch**aos		**oa**	OH	f**oa**m		**uie**	WAYAX	q**uie**t
	K/X	lo**ch**		**oe**	OW	f**oe**		**uo**	WOW	q**uo**te
d	D	**d**og			OWEH	p**oe**t, c**oe**xist		**uy**	AY	b**uy**
dg	J	fu**dge**			AH	d**oe**s		**v**	V	**v**ote, se**v**en
e	EH	b**e**g		**oi**	OY	c**oi**n		**w**	W	**w**in, **w**eather
	IY	h**e**		**oo**	UH	b**oo**k			WX	sa**w**
ea	IY	m**ea**t			UX	l**oo**t		**wh**	WH	**wh**en, **wh**ether
	EH	d**ea**d			UW	b**oo**			/X	**wh**o
	IYAE	**rea**ct			AH	bl**oo**d		**x**	KS	e**x**act, jin**x**
eau	YUW	b**eau**ty		**ou**	AW	c**ou**ch			EHKS	**X**-Ray
	IYAO	**reau**thor			AH	t**ou**ch			Z	**x**ylophone
ee	IY	fr**ee**		**ough**	AO	th**ough**t		**y**	Y	**y**ou
	IYEH	r**ee**nter			OW	d**ough**			YX	sa**y**
ei	IY	rec**ei**ve			UW	thr**ough**			AY	m**y**
	IYIH	r**ei**mage		**ow**	AW	h**ow**			IY	carr**y**
eigh	EY	w**eigh**			OW	sl**ow**		**z**	Z	**z**oo
er	ER	h**er**d		**oy**	OY	b**oy**			ZH	a**z**ure
ew	UW	n**ew**, cr**ew**		**p**	P	**p**oke				
ey	EY	th**ey**		**q**	K	**q**ueue				
	IY	k**ey**		**r**	R	**r**ed				
f	F	**f**ish			RX	ba**r**				
g	G	**g**o		**s**	S	**S**am				
	GX	pro**g**ress			Z	bit**s**				
	J	hu**g**e			ZH	mea**s**ure				
h	/H	**h**ead		**sch**	SK	**sch**ool				
i	IH	f**i**t		**sh**	SH	**sh**e				
	IX	digit			*	**sh**tik				
	AY	**i**ce, b**i**cycle, h**i**		**si**	ZH	vi**si**on				
ia	AYAX	b**ia**s		**ssi**	SH	pa**ssi**on				
	AY	d**ia**mond		**t**	T	**t**alk				
ie	AY	d**ie**			DX	pi**t**y				
	IY	ch**ie**f		**th**	DH	**th**en, **th**e				
	AYAX	d**ie**t			TH	**th**in, pa**th**				

On the phoneme chart, you will notice six phonemes: YX, WX, RX, LX, /X, and DX

Which are described as being used by SAM's rule system. However, they have been provided with letter codes so that you may experiment with these special sounds directly. YX and WX are weaker versions of Y and W. RX and LX are smooth gliding versions of R and L. /X is the "h" sound in "who", and DX is the quick flap of the tongue on the upper palate as in the word "pity".

We are now ready to transcribe ordinary speech into its phonetic representation. Let's use the following sentence as an example: “I do my calculations on the computer.”

The first step is to say each word aloud and decide how many syllables are in the word. a syllable has one vowel phoneme and its associated consonants (if any). We then identify the proper vowel phoneme by comparing its sound to lhe sounds listed in the table, and do the same for the consonants. The resultant combination of phonemes is the phonetic representation of the syllable. We do this for each syllable in a word.

In our example. the first word "I" is a single phoneme, the diphthong "AY". The next word "do" is a single syllable comprised of the diphthong "UW" preceded by the voiced consonant "D". The phonetic spelling is therefore "DUW". Similarly. the third word "my" again uses the "AY" sound, this time preceded by an "M", resulting in "MAY".

The word "calculations" has four syllables. The first syllable transcribes as "KAEL". The "c" sound is pronounced as "k". unlike the "s" pronunciation in a word like "cell" (notice there is no "C" in the phoneme table). The next syllable "cu" transcribes as "KYUW". Note here that the "Y" sound prevents this syllable from being pronounced as "coo". The third syllable comes out as "LEY", and the fourth becomes "SHAXNZ". This word ends with a voiced sound "7" and notthe hissy"S" sound as in "list". You will rapidly discover that many words contain the phonetic combinations "AXL". "AXM". and "AXN". To enhance the readability of the phonetic spelling, the special symbols "UL". "UM", and "UN" can be substituted for these combinations. The "tions" syllable is now written as "SHUNZ". So, "calculations" becomes "KAELKYUWLEYSHUNZ".

The next word "on" becomes "AAN", and "the" becomes "DHAX". By the way. if the word "the" precedes a word beginning with a vowel, it gets pronounced "thee" and is spelled "DHIY". You should also notice that the "th" letter combination has two phonetic representations: unvoiced (TH) as in "thin", or voiced (DH) as in "the".

Once you get used to the phonetic system, it will seem very easy and obvious. Initially, there will be some spellings that seem tricky (did you know that "adventure" has a "CH" in it?). However, the rule is always to write the word the way you say it, not the way you spell it.

To help you learn the system fast, an English-to-phonetic spelling dictionary of almost 1500 words. Many common words are in the dictionary, some unusual ones are in it as well. If you are really stuck on how to spell a word that isn't in the dictionary, think of another word that sounds like it and that one may be listed. Remember that it is all about how things sound and not how they are written.

In any case, don't hesitate to experiment with the phonetic spelling system. Let your ears be your guide. This system is relatively easy to learn, use and read, and you will be amazed at what you can do with it.

5.2 Adding stress to SAM's speech

In the phonetic mode, SAM is capable of speaking with a great deal of inflection and emphasis. This gives a much more natural and understandable quality to the speech than is otherwise possible.

The stress system for SAM is particulary easy to use. There are eight stress markers that can be used simply by inserting a number (1-8) after the vowel to be stressed. For example. the monotonic pronunciation of the word "hello" produced by the phonetic spelling "/HEHLOW" becomes a much friendlier sounding greeting when spelled "/HEH3LOW".

Why do you have to put in the stress markers? Simply because they can go anywhere and SAM has no way of knowing where you want them to go. The following simple example will demonstrate this. We will have SAM say "Why should I walk to the store?" in a number of different ways.

1. WAY2 SHUH7D AY WAO5K TUX DHAH STOH5R.	(You want a reason to do it.)
2. WAY7 SHUH2D AY WAO7K TUX DHAH STOH5R.	(You are reluctant to go.)
3. WAY5 SHUH7D AY2 WAO7K TUX DHAH STOHR.	(You want someone else to do it.)
4. WAY5 SHUHD AY7 WAO2K TUX7 DHAH STOHR.	(You'd rather drive.)
5. WAY5 SHUHD AY WAO5K TUX DHAH STOH2OH7R.	(You want to walk somewhere else.)

Each of these stress examples has a slightly different meaning, even though the words are all the same. Stress markers give you the ability to let SAM be expressive. What do the stress markers do? The number you type tells SAMto raise (or lower) his pitch and elongate the associated vowel sound. The number system works like this:

1 = very emotional stress

2 = very emphatic stress

3 = rather strong stress

4 = ordinary stress

5 = light stress

6 = neutral (no pitch change) stress

7 = pitch-dropping stress

8 = extreme pitch-dropping stress

When should you use each of these? It all depends on how you want SAM to sound. Say the words to yourself as expressively as you can and see where your voice rises and falls. Remember, the smaller the number, the more extreme the emphasis will be. Also, the stress markers will help get difficult words pronounced correctly. If some syllable is not enunciated sufficiently, put in a neutral stress marker.

A general rule is that the most important word or words in a sentence get the most stress and the rest of the words get little or no stress. However, words of more than one syllable should have stress marked on their accented syllables (most dictionaries show which these are if you are uncertain).

We will now assign stresses to our first example sentence about doing calculations on the computer. The first word "AY" is usually an important word (can you think of anyone more important?). We will write it as "AY4", assigning ordinary stress. "DUW", the only verb, is also important. We'll try "DUW4". "MAY" isn't very strong (unless you want to draw attention to it) and it is a single syllable. so we will leave it alone. "KAELKYUWLEYSHUNZ" is polysyllabic so we must identify the accented syllables. It is also the most important word in the sentence so it will have the strongest stress. "LEY" has the primary stress and "KAEL" receives the secondary stress. so we will write "KAE4LKYUWLEY3SHUNZ". "AAN" and "DHAX" are short, unstressed words. "KUMPYUWTER" has a single accent on "PYUW" and gets written "KUMPYUW4TER". So. our original sentence gets written:

AY4 DUW4 MAY KAE4LKYUWLEY3SHUNZ AAN DHAH KUMPYUW4TER.

How about really unusual stress? When you place extraordinary emphasis on a word, you do so by elongating its vowel sounds. SAM can do the same thing. For example, a call for help can become "/HEH5EH4EH3EH2EH2EH3EH4EH5EHLP." You can always do this with the ordinary vowel sounds, but be careful with the diphthongs. They are complex sounds and if you repeat them, they will not do what you want (e.g. "OYOYOYOYOYOY" sounds just like it reads in English). To extend the diphthong sounds, you need to break them into component parts. So "OY" can be extended with "OHOHIYIYIY", and "AY" can be extended with "AAAAIYIYIY". You should experiment to find out just what you can do.

Unlike many other speech synthesis systems, SAM allows you to control consonant stresses directly. This is usually done to produce a special tonal pattern in a word. Sometimes you might want a pitch rise on the final phoneme occurring just before a comma. For example, try typing: "AY4 YUWZ SAE5M3, AE4ND RIYSAY4TER." Notice how the pitch rises on the "M". It is never necessary to specify stress for a consonant occurring immediately before a stressed vowel. This is handled automatically.

Try to become familiar with the stress marker system. It makes all the difference between an ordinary speech synthesizer and the very expressive SAM.

5.3 Final notes on phonetic input

SAM is capable of speaking only 2.5 seconds of speech without a break (this is the size of his "breath"). If the string to be spoken exceeds this, SAM will insert short breaks every 2.5 seconds. SAM always breaks at punctuation marks in anticipation of the following phrase. So, if you don't like where SAM broke up a phrase, you can specify your own breaks with hypens. An example of this is: "I use the telephone - to call out of town".

SAM uses the spaces between words to makes his sentence-breaking decisions. If a single word requires more than 2.5 seconds to say, SAM will not be able to insert his own breaks and will therefore be unable to say the word.

In summary, the procedures outlined above may seem complex, but this is because they were presented in fine detail. In reality, the steps become automatic and you will soon be able to type in phonetics almost as fast as you can type English text.

6 The use of pitch and speed controls

SAM is capable of speaking in a wide range of tones and at many different rates. Both pitch and speed controls can be configured using the PITCH and SPEED commands. The effects of the values for pitch and speed are shown below, but for a good idea of how it sounds, you should just play with the values.

PITCH	(default = 64)
00-20	impractical
20-30	very high
30-40	high
40-50	high normal
50-70	normal
70-80	low normal
80-90	low
90-255	very low

SPEED	(default = 72)
00-20	impractical
20-40	very fast
40-60	fast
60-70	fast conversational
70-75	normal conversational
75-90	narrative
90-100	slow
100-225	very slow

There are also THROAT (default=128) and MOUTH (default=128) settings. These settings are very difficult to describe and it is questionable if any value other than default improves the sound. So, just play with them and see for yourself. Some examples that might be of use as a sort of guide to choose a voice are in the table below.

Voice suggestions				
DESCRIPTION	**SPEED**	**PITCH**	**THROAT**	**MOUTH**
SAM (default)	72	64	128	128
Elf	72	64	110	160
Little Robot	92	60	190	190
Stuffy Guy	82	72	110	105
Little Old Lady	82	32	145	145
Extra-Terrestrial	100	64	150	200

Below, is an example program that demonstrates how you can change these values of SAM's voice to create a different sound. This way you could simulate different voices that you may require if you are to use SAM for an adventure game. Or perhaps you want to personalize SAM's voice to something that you find more pleasant to listen to. Keep in mind that if you change these settings and then use the

command “-CONFIG SAVE”, that these settings are saved. Meaning that after a reset SAM’s voice will sound just like the way you configured it.

```
10 open 1,2,3,chr$(10):rem open serial
11 print#1,"-config debug0"
12 print#1,"-config speed72"
13 print#1,"-config pitch64"
14 print#1,"-config throat128"
15 print#1,"-config mouth128"
16 print#1,"i can speak normal"
17 rem -------------
22 print#1,"-config speed200"
26 print#1,"or very slow"
27 rem -------------
32 print#1,"-config speed30"
36 print#1,"or very fast"
37 rem -------------
42 print#1,"-config speed72"
43 print#1,"-config pitch40"
46 print#1,"or in a very high pitch"
47 rem -------------
53 print#1,"-config pitch200"
56 print#1,"or just very low"
57 rem -------------
62 print#1,"-config speed92"
63 print#1,"-config pitch60"
64 print#1,"-config throat190"
65 print#1,"-config mouth190"
66 print#1,"or like a little robot"
67 rem -------------
72 print#1,"-config speed100"
73 print#1,"-config pitch150"
74 print#1,"-config throat150"
75 print#1,"-config mouth128"
76 print#1,"or like a big robot"
77 rem -------------
82 print#1,"-config speed72"
83 print#1,"-config pitch64"
84 print#1,"-config throat128"
85 print#1,"-config mouth128"
86 print#1,"so, want do you want me to say now?"
```

The example on the previous page, was written with readability in mind. Therefor every configuration is written on it's own line. However, you can make the code much more efficient by combining multiple settings on a single line. For instance:

```
42 print#1,"-config speed72"
43 print#1,"-config pitch40"
```

Could also be written as:

```
42 print#1,"-config speed72 pitch40"
```

This makes your listing shorter and saves you precious space in your CBM's memory.

7 Making SAM sing

It is possible to make SAM sing. But for a speech synthesizer to be able to sing some settings need to be changed. Both singmode and phonetic have to be activated (which can be done by setting their value to 1).

```
-CONFIG SINGMODE1 PHONETIC1
```

When you're done playing with the singmode, you can disable this mode again by setting both values to 0 using the command: `-CONFIG SINGMODE0 PHONETIC0`

Then is "just" a matter of feeding SAM with the proper speech (in the form of phoneme's as it will allow you to stretch a word as you would normally do when singing) and we need to set the proper pitch. In order to get the proper pitch values there is a small table of notes below that make it a bit easier to find the correct value. Please be aware that this pitch produced by SAM is an approximation of the desired frequency, it is not perfect but it is sufficient.

Musical notes and their pitch values

Note	Pitch	Freq.		Note	Pitch	Freq.		Note	Pitch	Freq.
C	115	130.82		C	58	261.63		C	29	523.25
C#	108	138.59		C#	55	277.18		C#	28	554.37
D	103	146.83		D	52	293.66		D	26	587.33
D#	98	155.56		D#	49	311.13		D#	25	622.25
E	94	164.81		E	46	329.63		E	23	659.26
F	88	174.61		F	44	349.23		F	22	698.46
F#	82	185.00		F#	42	369.99		F#	21	739.99
G	78	196.00		G	39	392.00		G	20	783.99
G#	74	207.65		G#	37	415.30		G#	19	830.61
A	70	220.00		A	35	440.00		A	18	880.00
A#	66	233.08		A#	33	466.16		A#	17	932.33
B	62	246.94		B	31	493.88		B	16	987.77

Making SAM (a speech synthesizer) sing is nothing new. The song “Daisy Bell” was performed by the IBM 704 a very long time ago. This highly professional machine used mainly vacuum tube technology and was very sophisticated for its time. It made the speech sounds by being connected to a vocoder. The IBM 704 could do 12000 floating point operations per second (flops). For comparison, the C64 does approximately 2200 flops.

In 1962 Arthur C. Clarke (who wrote the novel and co-wrote the screenplay for the 1968 movie – “2001: A Space Odyssey”) visited Bell Labs. There, he was treated to a performance of the song ‘Daisy Bell’ by the IBM 704 computer. This evidently inspired him to have the HAL-9000 computer sing the song during it's final scene. This as an homage to the programmers of the 704 at Bell Labs.

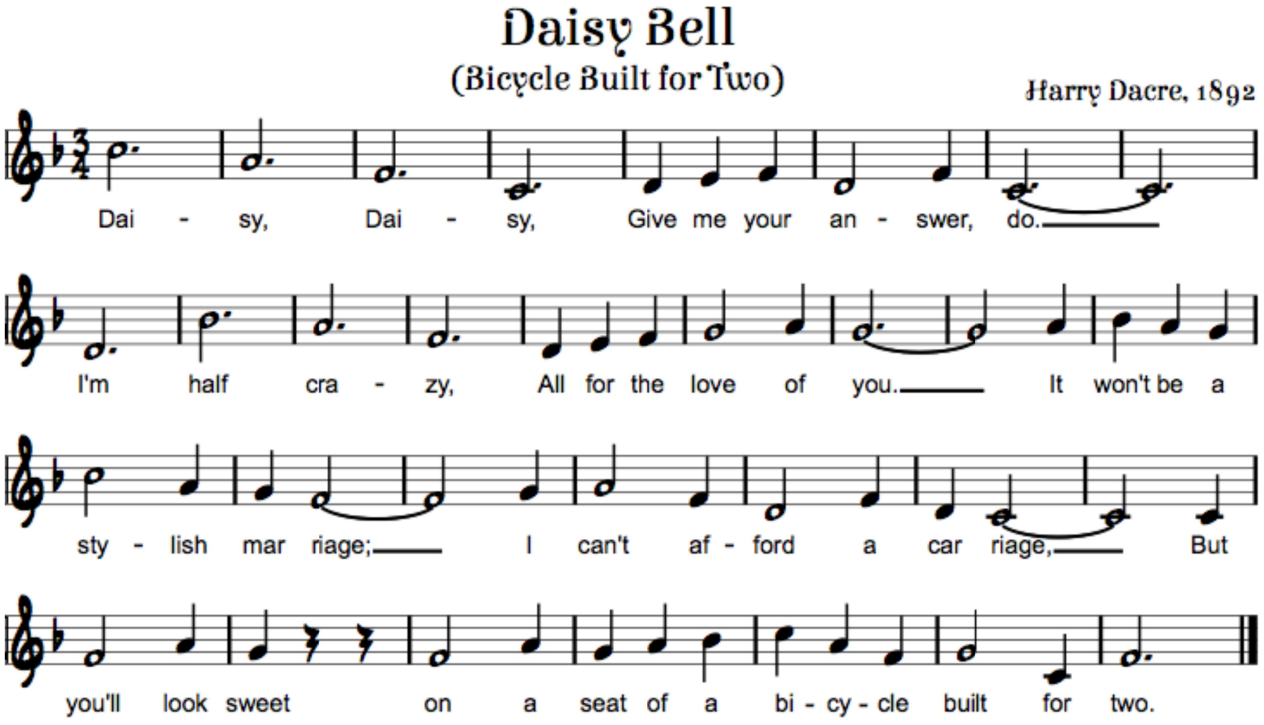
In the movie you'll see the astronaut disabling the HAL-9000 computer by removing all the memory modules in order to save its own life. As the computer gradually starts loosing its memory it refers to its earliest memories and tells the astronaut about a song it's being taught a long time ago. The nervous astronaut agrees in the song being sung by the computer and HAL-9000 sings the song to which it refers to as “Daisy”. The computer sings the song in a very slowed down version suggesting the regression and failure of the HAL-9000 computer. This reference in the movie was perhaps one of the reasons why we all still know this song and know it as “Daisy”. The song has been used or referred to (sometimes in very subtle ways) in many other films, television shows and even video games. Most likely because this was such an iconic scene in a movie praised for its impressive special effects.

Therefore over the years, it appears that this little song has become more and more historically significant. And if it regards singing speech synthesizers, then this song should not be left out. Therefore a small part of this song (as explained on the next page) has been implemented into the demo-mode of this module. So with a single command `-DEMO1` you can make SAM sing this song. Example:

```
10 OPEN 1,2,3,CHR$(10)
20 PRINT#1,"-DEMO1"
```

On the next page is an explanation of how this song was programmed. It makes a nice example for making SAM sing your own songs. But, keep in mind that not every song is suited to be sung by SAM.

Below is the song “Daisy Bell” or “Bicycle build for two” (or just “Daisy” for short) has the following notes and lyrics:



So to make SAM sing “*Daisy, Daisy, give me your answer do.*”, we need to send the correct phonemes at at the correct pitch as shown below:

Pitch = 26 “DEYYYYYYYYY”
Pitch = 31 “ZIYIYIYIYIYIYIY”
Pitch = 39 “DEYYYYYYYYY”
Pitch = 52 “ZIYIYIYIYIYIYIY”
Pitch = 46 “GIXV”
Pitch = 44 “MIYIY”
Pitch = 39 “YAOW”
Pitch = 46 “AEAEAEN”
Pitch = 39 “SERER”
Pitch = 52 “DUXUXUXUXUXUXUXUXUXUXUXUX”

Tip: if you want SAM to sing uninterrupted, it might be a good idea to set debug-mode to 0

Now the demo of SAM singing (which you can start by sending the command “demo1”) wasn't written in CBM BASIC, because it runs inside the micro controller running the SAM software. But you still may be curious on how that code, to make SAM actually sing, looks. Below the C-code used for making SAM sing the daisy song. You'll notice the setting of the frequency of SAMs voice, a simple constant/define is used to write the value 26 as N2D. But this was done purely to improve readabillity, considering 26 isn't as intuitive on a note-scale as N2D is. Where N stands for Note, 2 stands for octave and D stands for the actual note. After the pitch is set, the text can be spoken. Again with phonetic sounds. So it's the same as you would do it in BASIC only with slightly different notation.

```
/*daisy, daisy, give me your answer do*/
sam->SetPitch(N2D);  sam->Say(out,"DEYYYYYYYY");               /*dai*/
sam->SetPitch(N1B);  sam->Say(out,"ZIYIYIYIYIYIYIY");          /*sy*/
sam->SetPitch(N1G);  sam->Say(out,"DEYYYYYYYY");               /*dai*/
sam->SetPitch(N1D);  sam->Say(out,"ZIYIYIYIYIYIYIY");          /*sy*/
sam->SetPitch(N1E);  sam->Say(out,"GIXV");                     /*give*/
sam->SetPitch(N1F);  sam->Say(out,"MIYIY");                    /*me*/
sam->SetPitch(N1G);  sam->Say(out,"YAOW");                     /*your*/
sam->SetPitch(N1E);  sam->Say(out,"AEAEAEN");                  /*an*/
sam->SetPitch(N1G);  sam->Say(out,"SERER");                    /*ser*/
sam->SetPitch(N1D);  sam->Say(out,"DUXUXUXUXUXUXUXUXUXUXUX"); /*do*/
delay(250);

/*I'm halve crazy all for the love of you*/
sam->SetPitch(N1A);  sam->Say(out,"AYYYYYYMM");
sam->SetPitch(N2D);  sam->Say(out,"/HAEAEAEAEAEAEF");
sam->SetPitch(N1B);  sam->Say(out,"KREYYYYYYY");
sam->SetPitch(N1G);  sam->Say(out,"ZIYIYIYIYIYIYIY");
sam->SetPitch(N1E);  sam->Say(out,"AXLL");
sam->SetPitch(N1F_); sam->Say(out,"FAOR");
sam->SetPitch(N1G);  sam->Say(out,"DHER");
sam->SetPitch(N1A);  sam->Say(out,"LUHUHUHV");
sam->SetPitch(N1B);  sam->Say(out,"AXAXV");
sam->SetPitch(N1A);  sam->Say(out,"YUXUXUXUXUXUXUXUXUX");
delay(250);

/*it won't be a stylish marriage I can't afford a carriage*/
sam->SetPitch(N1B);  sam->Say(out,"IHT");
sam->SetPitch(N2C);  sam->Say(out,"WOWNT");
sam->SetPitch(N1B);  sam->Say(out,"BIY");
sam->SetPitch(N1A);  sam->Say(out,"ER");
sam->SetPitch(N2D);  sam->Say(out,"STAYYYYY");
sam->SetPitch(N1B);  sam->Say(out,"LIHSH");
sam->SetPitch(N1A);  sam->Say(out,"MAE");
sam->SetPitch(N1G);  sam->Say(out,"RIXIXIXIXIXIXIXIXIXIXIXJ");
```

```
sam->SetPitch(N1A);  sam->Say(out,"AYY");
sam->SetPitch(N1B);  sam->Say(out,"KAEAEAEAENT");
sam->SetPitch(N1G);  sam->Say(out,"ER");
sam->SetPitch(N1E);  sam->Say(out,"FAOAOAORD");
sam->SetPitch(N1G);  sam->Say(out,"ER");
sam->SetPitch(N1E);  sam->Say(out,"KAA");
sam->SetPitch(N1D);  sam->Say(out,"RIXIXIXIXIXIXIXIXIXIXIXIXJ");
delay(250);

/*but you look sweet upon the seat of a bicycle made for two*/
sam->SetPitch(N1D);  sam->Say(out,"BUHT");
sam->SetPitch(N1G);  sam->Say(out,"YUXUXL");
sam->SetPitch(N1B);  sam->Say(out,"LUXK");
sam->SetPitch(N1A);  sam->Say(out,"SWIYIYIYIYT");
sam->SetPitch(N1D);  sam->Say(out,"ER");
sam->SetPitch(N1G);  sam->Say(out,"PAAAAAAN");
sam->SetPitch(N1B);  sam->Say(out,"ER");
sam->SetPitch(N1A);  sam->Say(out,"SIYIYIYT");
sam->SetPitch(N1B);  sam->Say(out,"UHV");
sam->SetPitch(N2C);  sam->Say(out,"ER");
sam->SetPitch(N2D);  sam->Say(out,"BAY");
sam->SetPitch(N1B);  sam->Say(out,"SIH");
sam->SetPitch(N1G);  sam->Say(out,"KUXL");
sam->SetPitch(N1A);  sam->Say(out,"MEYYYYD");
sam->SetPitch(N1D);  sam->Say(out,"FER");
sam->SetPitch(N1G);  sam→Say(out,"TUXUXUXUXUXUXUXUXUXUX");
```

The table below shows the values required to approximate the perfect note. The speech synthesizer isn't very accurate regarding pitch, but the result is good enough for a simple song.

```
#define N1C   58        //261.63                  #define N2C   29        //523.25
#define N1C_  55        //277.18                  #define N2C_  28        //554.37
#define N1D   52        //293.66                  #define N2D   26        //587.33
#define N1D_  49        //311.13                  #define N2D_  25        //622.25
#define N1E   46        //329.63                  #define N2E   23        //659.26
#define N1F   44        //349.23                  #define N2F   22        //698.46
#define N1F_  42        //369.99                  #define N2F_  21        //739.99
#define N1G   39        //392                     #define N2G   20        //783.99
#define N1G_  37        //415.3                   #define N2G_  19        //830.61
#define N1A   35        //440                     #define N2A   18        //880
#define N1A_  33        //466.16                  #define N2A_  17        //932.33
#define N1B   31        //493.88                  #define N2B   16        //987.77
```

Making SAM sing is fun, but a lot of work to do it right. Making SAM sing from BASIC is slightly different, for some a slight disappointment maybe. The reason for that is that there always will be a small delay between the given command and the spoken text. In other words, there will be tiny, slightly awkward, pauses. These are caused by the time it takes to send the data over the serial port, which only operates at a speed of 2400 bits/second. However, this problem is not an issues on the real SAM (that ran on purely software in the memory of a C64)? So If you really want SAM, to sing the latest Rick Astley song, then perhaps you can combine the information from this manual, with a copy of the original SAM software and then play along with that.

Below is a small example of the same song DAISY, but now completely controlled from your Commodore computer from BASIC. Please notice the DEBUG0 setting, which prevents SAM from confirming every pitch value change.

```
8 REM -= PREPARE SAM =-
10 OPEN 1,2,3,CHR$(10):GOSUB 900
11 PRINT#1,"-CONFIG DEBUG0":GOSUB910
13 PRINT#1,"-CONFIG SINGMODE1":GOSUB910
15 PRINT#1,"-CONFIG PHONETIC1":GOSUB910
16 PRINT#1,"-CONFIG PAUSE0":GOSUB910
18 REM -= READ TABLE =-
20 READ A$,B$:IF A$="*" THEN 99
21 REM:PRINT "-CONFIG PITCH";A$
22 REM:PRINT B$
23 PRINT#1,"-CONFIG PITCH";A$:GOSUB910
24 PRINT#1,B$:GOSUB910
25 GOTO 20
99 END
101 REM -= DATA TABLE =-
103 DATA"26","DEYYYYYYYYY"
104 DATA"31","ZIYIYIYIYIYIYIY"
106 DATA"39","DEYYYYYYYYY"
107 DATA"52","ZIYIYIYIYIYIYIY"
108 DATA"46","GIXV"
109 DATA"44","MIYIY"
110 DATA"39","YAOW"
111 DATA"46","AEAEAEN"
112 DATA"39","SERER"
113 DATA"52","DUXUXUXUXUXUXUXUXUXUXUX"
900 GET#1,A$:IFA$<>"" THEN 900
902 RETURN
910 GET#1,A$:IFA$<>CHR$(13) THEN 910
913 RETURN
999 DATA"*","*"
```

The song is in the data statements, each line consists of a pitch value and a phonetic text. At the beginning of the program SAM is put into singing mode, when the song is over SAM remains in singing mode. You may give the command to undo that or you can simply reset your system.

8 Dictionary

SAM has an algorithm that can determine how to pronounce a word for the English language. A system that is able to convert plain text into spoken speech is mostly referred to as text-to-speech. Many speech synthesizer of the 80's did not have this functionality! Which is slightly strange as it does make a speech synthesizer much more easy to operate while keeping it versatile.

A text-to-speech system prevents the need for spelling out the exact pronunciation (list of individual phonemes required for speaking a word). Originally this piece of SAM code is called the "reciter" and it is used for every word given to SAM. The only way to bypass it is to set SAM into the phonetic speech mode. The reciter routines do a pretty good job and are able to pronounce most of the word with sufficient accuracy. Although in some rare cases a word needs to be tweaked in order to make SAM speak exactly it the way you want. In those cases a dictionary could be of help. This would allow to search for these exceptions and speak them in a predefined manor. The definition of the pronunciation of a word in consisting out of multiple phoneme is defined in a file called "dictionary".

Dictionary functionality is a function that can be enabled/disabled. And the way it works (if the dictionary function is enabled) is relatively simple. Every word send to SAM is searched for in the dictionary. And if found, the list of phonemes as defined in the dictionary for that word are used to speak that word. If it cannot be found, SAM uses the reciter routines and therefore determines for itself how a word should be pronounced, which in many cases will be just perfect. So using a dictionary you can have best of both worlds. You can speak with the ease of direct text-to-speech and have the accuracy of a dictionary for those special exceptions that do require some extra attention with their pronunciation.

8.1 How it works

Upon power-on reset, the system will search through the file-system to see if a dictionary is present. If found an index will be made of every letter of the alphabet. This way when a word starting with (for example) the letter 'H' is searched for, SAM already knows the offset in the file and can jump directly to that location. Starting it's search from the first word in the dictionary starting with that letter. And because all words are stored in the dictionary in alphabetical order SAM can also detected pretty quickly if a word is present or not. This is VERY important, because everything needs to happen in real-time, if the search through the dictionary was to be slow, SAM would speak with awkward and varying pauses. Therefore it is important to prevent the dictionary from being too large. A dictionary with a size of a thousand words should not be a problem for SAM to handle. This is something you may want to experiment with.

SAM supports up to two dictionaries and scanning these might take a few seconds depending on their size. This means that it takes the same amount of time longer before SAM pronounces it startup or reset message. In order to reduce this delay, reduce the size of the dictionary. Or perhaps remove one or both of the possible dictionaries.

8.2 Dictionary rules

In order to write a dictionary there are some syntax rules to obey, otherwise it will not work properly.

- The first line of the dictionary must contain a definition of the content. This is basically a commented line and comments are indicated by a simple '*' character. Everything on the line that starts with a '*' is ignored.
- All words need to be stored in an alphabetical order. If this is not the case, the dictionary will not function properly.
- Words may not contain capitol letters
- Phonemes consist entirely of capitol letters
- A line with a word may not contain any space
- Word and phonemes are separated by the '=' character
- Refer to the chapter in this manual about the phonetic spelling system, as that goes into more detail on how to creates spoken words from phonemes. Keep in mind that you use the phonemes as defined in this manual, making a mistake (syntax error) might result in a word being not spoken at all. But assuming that you try out every word before you enter it in the dictionary this will not be a big issue.

An example of very simple dictionary containing only a few simple Dutch words is shown below. Now Dutch is not the language of the average SAM user. But it does demonstrate what is possible with the power of a dictionary and how simple it can be to define a dictionary. The main problem with a dictionary is that, writing it can be a lot of work. But you should never attempt to realize the perfect dictionary, as most words will never be spoken. Keep it simple, define the words you think you need for your current project and nothing more. And on the next project add some new words, etc. This way the dictionary grows with your needs and the task of writing the perfect dictionary feels less cumbersome as it is spread over a longer period of time.

```
* DUTCH Dictionary for SSSSAM
* ---------------------------
***A***
appel=AHPPAXLX
***B***
beest=BEYST
***C***
cirkel=SIH4KUL
***G***
goed=JUHT
***H***
hallo=/HAHLXLXOW
```

8.3 Multiple dictionaries

A dictionary is a powerful thing, if you know what a single dictionary can, just imagine what two dictionaries can do. Because this would allow you to speak two different languages. SAM supports up to two dictionaries. These are named the “primary dictionary” and the “secondary dictionary”. When SAM needs to speak a sentence and the dictionary function is enabled SAM first checks the primary dictionary, if the word is present there, that word will be spoken. But if the primary dictionary does not contain the word, the secondary dictionary is searched and if that search doesn’t find anything then SAM will speak using it’s own interpretation (as defined by the “reciter” algorithm) of the word.

This means that if you want SAM to speak Dutch, you define a dictionary of Dutch words and set that to the primary dictionary. Then SAM will always try to pronounce a word according to that dictionary and if the word is not present it will refer to the other dictionary etc.

8.4 Alternative usage

A dictionary is a powerful thing as it can do more things than define the way a word is spoken. It can also be used for preventing SAM of saying nasty things. Meaning that if you define a word that you do not want SAM to say then you simply define another word or leave out the definition completely, so SAM will remain silent when the word is requested to be spoken. Now the latter isn’t very userfriendly. Because it will make SAM look bad, because it makes it seem that it doesn’t work. So instead of keeping SAM silent, it would be better to make SAM say something innocent, like “BIYIYIYP” (which sounds like BEEEEP). This does allows some fun to, because the user can hunt for BEEPed out words. Search the swear word, could be a fun game.

Although this would make it possible to prevent SAM from speaking dirty or foul language, keep in mind that it is completely normal for kids to experiment with language. I’m pretty sure most kids of the 1980’s who are parent today did exactly the same.

Anyway, feel free to use your imagination. Because you can also make a prank dictionary for Aprils fools day. And change every word into the opposite. As an example to alternative speech below are a few examples of how this would look like in a censor or prank dictionary.

cock=BIYIYIYP	← censoring a word (beep it out)
cock=	← censoring a word (ignore it)
cock=RUHSTER	← censoring a word (alternative word)
hate = LAH4V	← fun word (pronounce the opposite)

8.5 Dictionary trouble shooting

No matter how much time and care you put into a dictionary errors can always slip in. Therefore it may occur that you think that a word is perfectly fine in the dictionary, but SAM doesn't seem to find it. In those cases there is something wrong with the dictionary and it needs to be fixed. Here are some things you might need to examine in order to solve it. Now we assume that you double checked the spelling of the word in both the dictionary AND the request made by your CBM computer.

SAM really likes the alphabet. Therefore all words in the dictionary must be stored in alphabetical order. This way SAM can make an index of all the letters of the alphabet and jump straight to the correct section of the dictionary when searching for a word. If the words aren't stored in this order, then SAM is confused. Also upper case or capitol letters and spaces confuse SAM.

Check if words are in alphabetical order. If SAM find a word that is beyond the alphabetical order of the word SAM is searching for, SAM will abort. For example if section C contains the following entries:

correction=KOHREH4KSHUN

count=KAW4NT

country=KAH4NTRIY

create=KRIYEY4T ← incorrect, alphabetical order is incorrect

cousin=KAH4ZIXN

critical=KRIH4TIXKUL

SAM will not be able to find the word "cousin", because SAM finds the word "create" first and because the letter 'r' comes after the letter 'o' in the alphabet, SAM will assume that the word is not in the dictionary. This is easily fixed by moving the word "create" down in the dictionary directly above "critical".

Check is words are witten in lower case, if SAM searches the dictionary but finds a word with a capitol letter SAM stops the search immediately, failing to find all the other words after that. For example if you have in the section for the letter D the following entries:

degree=DIXGRIY4

degrees=DAXGRIY4Z

Dakota=DAHKOW4TAH ← incorrect, no capitol letter in word allowed!!!

delaware=DEH4LAXWEH6R

delay=DIXLEY4

demonstrate=DEH4MUNSTREYT

SAM will not be able to find the word delay, simply because SAM stops the search when finding the D from Dakota. This is easily fixed by changing "Dakota" into "dakota".

Check for spaces, these should not be in the library. For example, if section E contains the following words:

electricity=ULEHKTRIH4SIXTIY

electronic=ULEHKTRAA4NIXK

elementary=EH4LUMEH4NTRIY

eleven=IXLEH4VIXN

emphasis = EH4MFAXSIHS ← incorrect, spaces are not allowed

encyclopedia=EHNSAY5KLAXPIY4DIYAH

energy=EH4NERJIY

engineering=EH5NJUNIY4RIHNX

enter=EH4NTER

SAM will not be able to find the word “emphasis” because SAM can only find “emphasis<space>“ which isn’t the same and therefore will be ignored.

And finally when SAM stays completely silent when a word should be spoken then there is something wrong with the list of phoneme. For example, if section L contains the following words:

liters=LIY4TERZ

little=LIH4TU ← incorrect, phoneme list is incomplete

load=LOW4D

SAM does find the word, but isn’t able to pronounce it because the list of phoneme is wrong. Just changes “LIH4TU” into “LIH4TUL”.

Now you may wonder why SAM is so critical about the dictionary. Couldn’t SAM be made less critical about the content of the dictionary? Well technically spoken anything is possible, the letter could be converted into the proper case and spaces could be simply ignored. But do not forget that SAM needs to function in real time. Every action that SAM needs to do requires processing time, resulting in awkward pauses if processing takes too long. Therefore it would make no sense to do these actions every single time for every single character in the dictionary considering that in 99.99999% of the time they aren’t required. This would only waste valuable processing time and only reduces the performance of SAM. Keep in mind that dictionaries can be large and can take up a lot of time to process! And let’s be honest, is it too much to ask to define a word according the rules of the dictionary? Nope it isn’t. So let’s help SAM out a little and obey his dictionary rules. Then everything will be just fine and will function as efficiently as possible.

9 Filesystem

SAM is a device that can store settings in non volatile memory. For example, you can define the welcome message and no matter how long your computer is switched off, SAM does not forget. So when it is switched on again things work the same way as the last time you used it. Now these settings need to be stored somewhere and according to some structure. The location is inside the flash memory of the processor and the structure is a filesystem with a JSON file for the settings and additional files (like the dictionary).

9.1 Filesystem

Although there is no real need for the common user to modify the content of the filesystem, there is a way to explore the contents and do some minor modifications. This may be desired in case of retrieving things from the filesystem or to do minor upgrades of the content. In order to do so a filebrowser that can be accessed through a webbrowser is to be used. In order to start the filebrowser SAM needs to be instructed to set up this functionality.

ATTENTION:

By default your VIC20 and C64 are in capital letter mode, for all commands this is perfectly fine. However, this is a problem when entering a case sensitive item like a networkname or password. Therefore you must switch your CBM computer to the lowercase mode by pressing <shift> + <CBM> on your CBM keyboard. This is the only way that your can edit/see the networkname and password properly. Unfortunately, there are some restrictions in the characters you may use, this because your CBM can only generate the following characters:

```
!"#$%&'()*+,-./0123456789:;<=>?@[]
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
```

The <space> and <return> “characters” are supported for normal usages, but cannot be used in a combination with a networkname or password.

If your networkname or password contains characters that are not supported by SAM then you cannot setup a connection, simply because the CBM computer does not allow you to enter the characters, as these characters are not on the keyboard, so you can’t type them in. The only solution would be to change your wifi network configuration in the router. Although you might consider to visit a friend with a less complicated wifi network name and password setting.

In order to connect the module to the wifi network, the user must enter the network SSID and Password, so that these settings can be used by SAM to connect to the network. Once connected, the user can go to a webpage hosted by SAM and upload the new firmware image using a simple filebrowser. In order to enable the webserver in SAM that hosts this little firmware updating webpage, a small program needs to be entered. This program sends the networkname (SSID) and password (also referred to as “key”) and starts the update mode. Type in the following program. Make sure that your system is in lower case mode in order to properly enter your network name and password (see screenshot):

```
10 open 1,2,3,chr$(10)
20 print#1,"-config ssid WiFinetwork"
30 print#1,"-config pass WiFipassword"
40 print#1,"-config filebrowser"
50 get#1,a$:if a$="" then 50
60 print a$;:goto 50
```

Line 10: open serial port

Line 20-30: send the wifi's SSID and password information (enter your personal values here, replace the text “WiFinetwork” with the name of the wifi network you want to connect with. And replace the text “WiFipassword with the password of the wifi network you want to connect with.)

Line 40: instruct SAM to start the filebrowser server (this blocks all other functionality)

Line 50-60: get all data from SAM over the serial port and display it on the screen of your CBM. This way you'll get some feedback regarding the wifi connection state and update progress. Also on the screen you'll see the IP-address to which you should go to with your browser in order to do the update.

If this programs doesn't work the first time, reset everything (computer off for 10 seconds) and try again (please double check your SSID and password settings before retrying). Sometimes wifi networks can be a bit stubborn and don't want to connect the 1st time, so no reason to panic. This can also be because of range limitations or a crowded network. Therefore it is preferred to do this action at home and close to your wifi router. Do not try this at a public network as these are very slow and unreliable.

Regarding security, SAM is normally not connected to the wifi network, so you do not need to worry about other people accessing it and updating your firmware without your knowledge. The wifi connection is only established when you give SAM the -CONFIG FILEBROWSER command.

If all goes well, the basic program will print an IP-address that you can enter in your webbrowser. It will look like the screenshot below.

<screenshot van filebrowser>

10 Technical info

10.1 Serial specifications

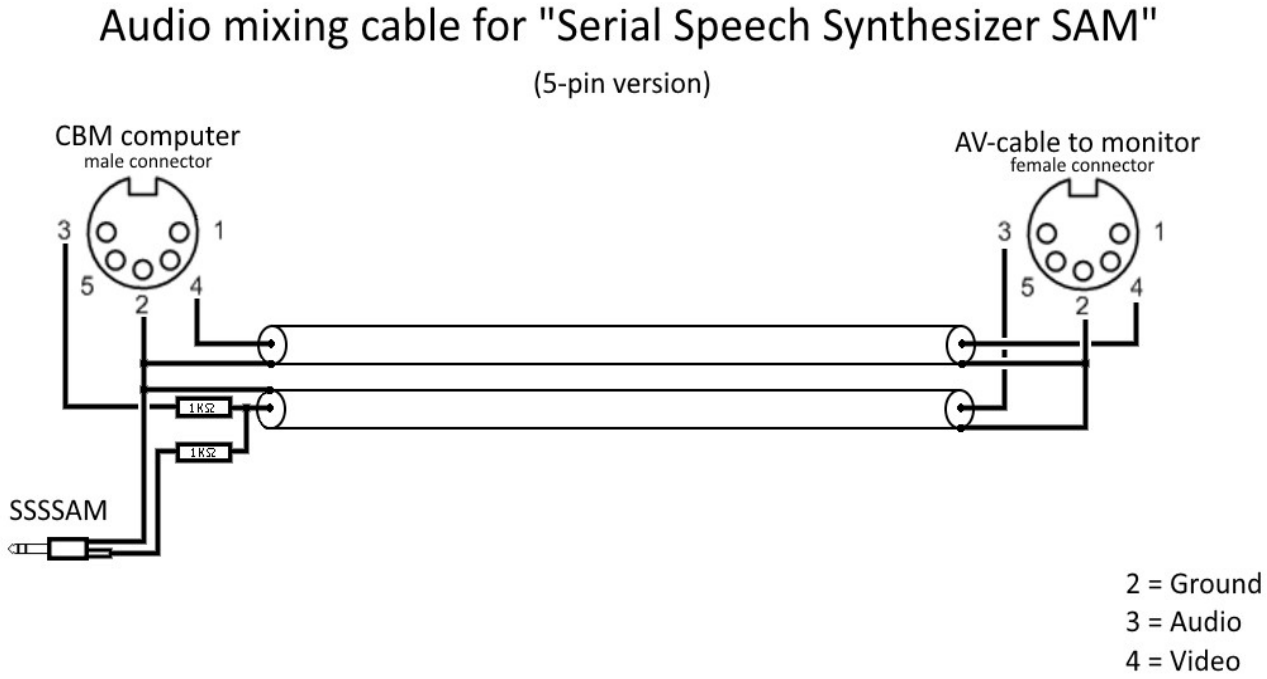
The device operates over the serial port, although the levels aren't RS-232, the timing and bit definitions are. Therefore we may state that this is TTL RS-232.

Baudrate	:	2400 bits/second
Number of bits	:	8
Parity	:	No
Number of stop bits	:	1
Flow control	:	No

SAM does not echo everything you send it, simply because this would only complicate the usage in combination with old computers, as they would need to process that data. However, SAM send an “OK” every time is has finished a command or speech request. So if many text needs to be spoken, then it might be useful to wait for the OK from SAM before sending a new text string.

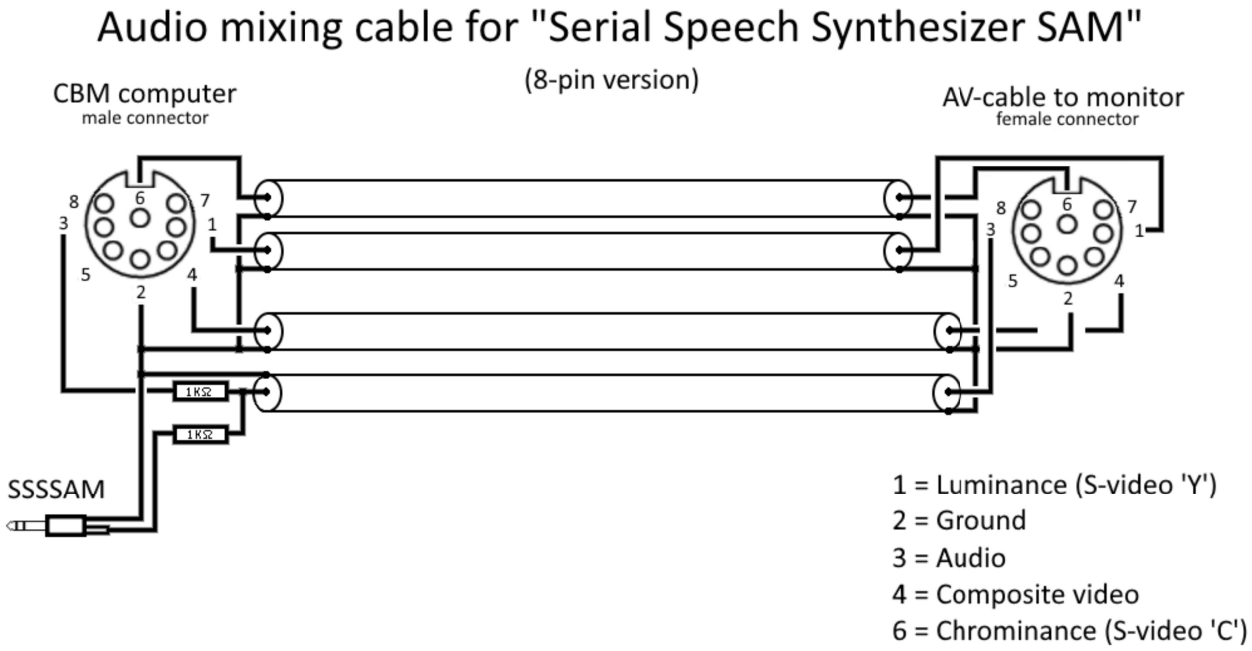
10.2 Audio mixing cable

This device is best to be used with the 5-pin composite video audio mixing cable. In case this cable get's lost or damaged you can repair or rebuild the cable using the schematic as shown below. The 5-pin cable can be used on all CBM computers that can connect to an external monitor.

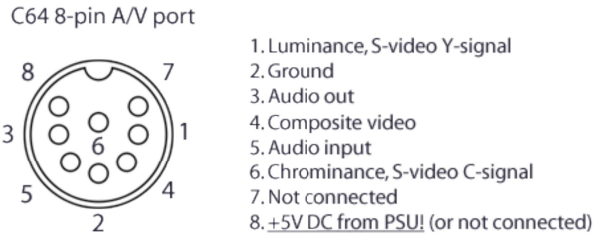


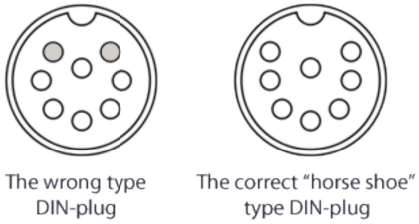
The schematic above shows the 5-pin AV cable (a.k.a. composite video cable) for carrying the audio and video signals using a coaxial cable. This connector fit's on all CBM computers that allow connection to an external monitor. The VIC-20 can only accept a 5-pin connector. The C64 and other models cab accept both the 5-pin and 8-pin connector. The 8-pin connector has 3 extra pins of which pin-1 and pin-6 carry the extra signals for the Y/C video system also known as S-video. This can produce slightly clearer image on your monitor/television (if your monitor/television supports it). Because the SSSSAM device has been developed for the VIC-20, it makes no sense in supplying a cable that doesn't fit into the VIC-20, therefore the 8-pin cable is not supplied with the SSSSAM device. Another major factor is that the 8-pin connector is expensive and difficult to obtain and would significantly increase the total cost of the device for functionality not used by the main public.

If you really desire the 8-pin video cable then the only option would be by making such a cable yourself.



Keep in mind that the 8-pin DIN plugs come in two variations, Commodore uses the “horse shoe” or “262 degrees” type, which isn’t very common. There is another 8-pin DIN connector that is of the “270 degrees” type, DO NOT USE THAT ONE, as it does not fit without damaging the connector in your C64. Unfortunatley, that 270 degrees connector is very common and this makes it very confusing. So keep this in mind if you want to make your own cables. The pin-out for the 8-pin connector and the layout of the 270 and 262 degrees connector types are shown on the next page.



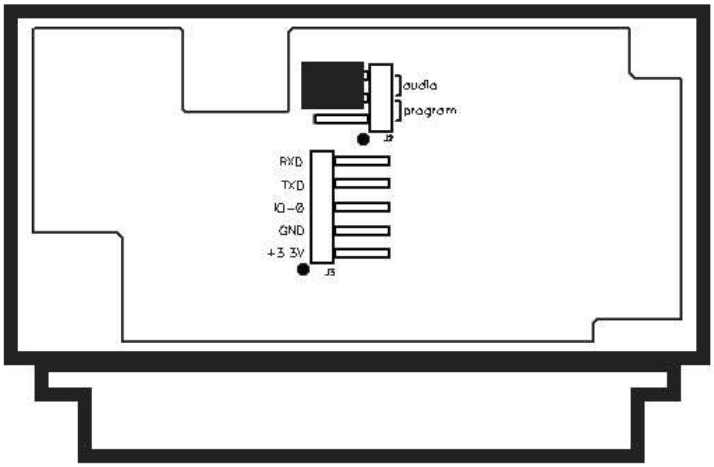


10.3 Updating firmware (method 1: using a cable)

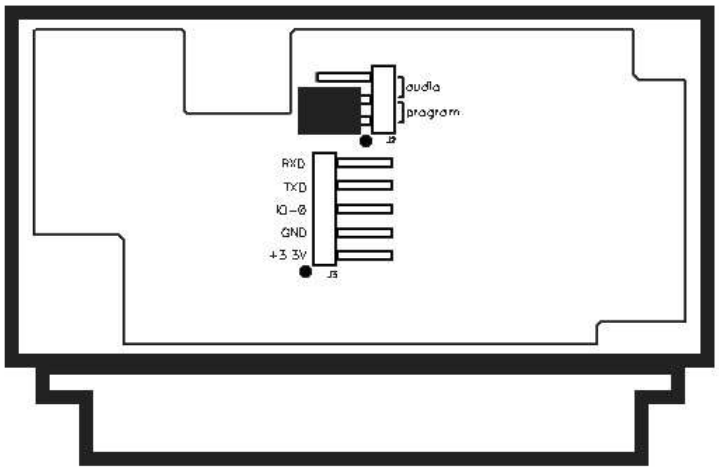
This device contains a micro controller that uses firmware programmed into the flash memory of the device. This firmware may be changed/upgraded but that procedure is not intended to be executed by the “average” user. This function is merely intended for use by developers using the Arduino IDE in combination with the ESP8266 core.

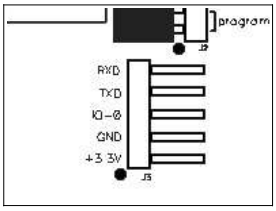
The audio is generated using the I2S functionality of the microcontroller (ESP8266) unfortunately this functionality is shared with the RXD line of the serial port required for firmware updates. Therefore in order to play audio or program firmware the circuitry has to be adjusted for that specific task. The jumper allows the audio filter circuit or the RXD signal to be (dis)connected.

Below is the position of the audio/program jumper shown for normal use. Place the jumper in this location (see the image below) if you want audio output on the audio jack socket.



In order to be able to program the device the jumper must be placed into the program location as shown below.





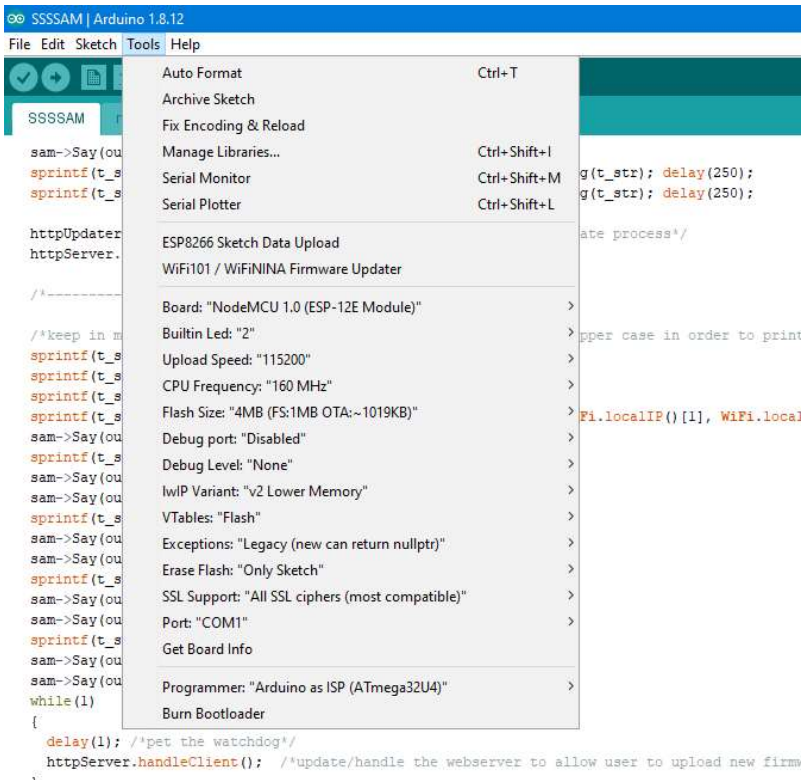
The 5-pin header allows for the RXD and TXD signals to be connected to your PC, the levels on these pins are designed to be used for 3.3V levels (although by changing the values of R15 and R16 on the PCB is possible to allows the use of 5V levels).

The 3.3V is an output, do not apply a voltage to this pin! This pin is intended to supply power to your RS-232 to TTL level shifter. However when a "modern" USB to TTL-RS-232 converter is used, it will require no power at all and all you need to do is connect the RXD, TXD and GND to the appropriate pins of that converter.

The IO-0 line is to be held low during power-on (or reset) to force the device into the bootloader mode. You must leave this pin floating during normal use.

10.3.1 *Developing firmware*

Tip: if you are developing firmware for this device, then changes are that you are constantly swapping the location of the audio/program jumper. For those situations it is highly recommended to place a simple toggle-switch via some 10cm long wires to the 3-pin jumper. And a push button to ground for the IO-0 signal. This way you can quickly switch between audio-output and programming mode and by holding down the push button during power-on (reset) you invoke the bootloader mode. Releasing the button after 3 seconds after power-on (reset) keeps the device in bootloader mode but allows it to start immediately upon finishing of the uploading of the new firmware (manual reset may be required). Regarding the IDE settings, make sure that your IDE is configured as the shown in the screenshot:



You might ask yourself “Where do I find the .bin files after compilation?”
After compilation (or ESP8266 sketch data upload (which generates the SPIFFS filesystem .bin file)) you may find the firmware and filesystem .bin files in the folder:

C:\Users\???\AppData\Local\Temp\arduino_build_???
The first set of question marks should be replaced with the username of your Windows system and the second set of question marks should hold the name of the arduino_build folder with the latest date.

You can copy these files from here in order to redistribute them among users so they can upload them through the webinterface bootloader.

10.4 Updating firmware (method 2: using a webbrowser)

If you want to update the firmware of SAM then you can use the webbrowser based bootloader. This method is very convenient for regular users because it doesn't require cables and you don't need to open the device. The only requirement is that SAM can connect to a wifi network.

ATTENTION:

By default your VIC20 and C64 are in capital letter mode, for all commands this is perfectly fine. However, this is a problem when entering a case sensitive item like a networkname or password. Therefore you must switch your CBM computer to the lowercase mode by pressing <shift> + <CBM> on your CBM keyboard. This is the only way that your can edit/see the networkname and password properly. Unfortunately, there are some restrictions in the characters you may use, this because your CBM can only generate the following characters:

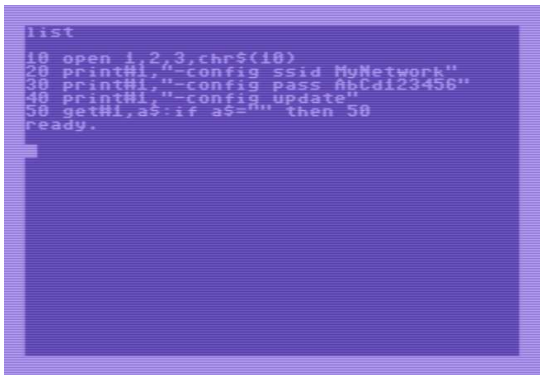
```
!"#$%&'()*+,-./0123456789:;<=>?@[]
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
```

The <space> and <return> "characters" are supported for normal usages, but cannot be used in a combination with a networkname or password.

If your networkname or password contains characters that are not supported by SAM then you cannot setup a connection, simply because the CBM computer does not allow you to enter the characters, as these characters are not on the keyboard, so you can't type them in. The only solution would be to change your wifi network configuration in the router. Although you might consider to visit a friend with a less complicated wifi network name and password setting.

In order to connect the module to the wifi network, the user must enter the network SSID and Password, so that these settings can be used by SAM to connect to the network. Once connected, the user can go to a webpage hosted by SAM and upload the new firmware image using a simple filebrowser. In order to enable the webserver in SAM that hosts this little firmware updating webpage, a small program needs to be entered. This program sends the networkname (SSID) and password (also referred to as "key") and starts the update mode. Type in the following program. Make sure that your system is in lower case mode in order to properly enter your network name and password (see screenshot):

```
10 open 1,2,3,chr$(10)
20 print#1,"-config ssid WiFinetwork"
30 print#1,"-config pass WiFipassword"
40 print#1,"-config update"
50 get#1,a$:if a$="" then 50
60 print a$;:goto 50
```



Line 10: open serial port

Line 20-30: send the wifi's SSID and password information (enter your personal values here, replace the text “WiFinetwork” with the name of the wifi network you want to connect with. And replace the text “WiFipassword with the password of the wifi network you want to connect with.)

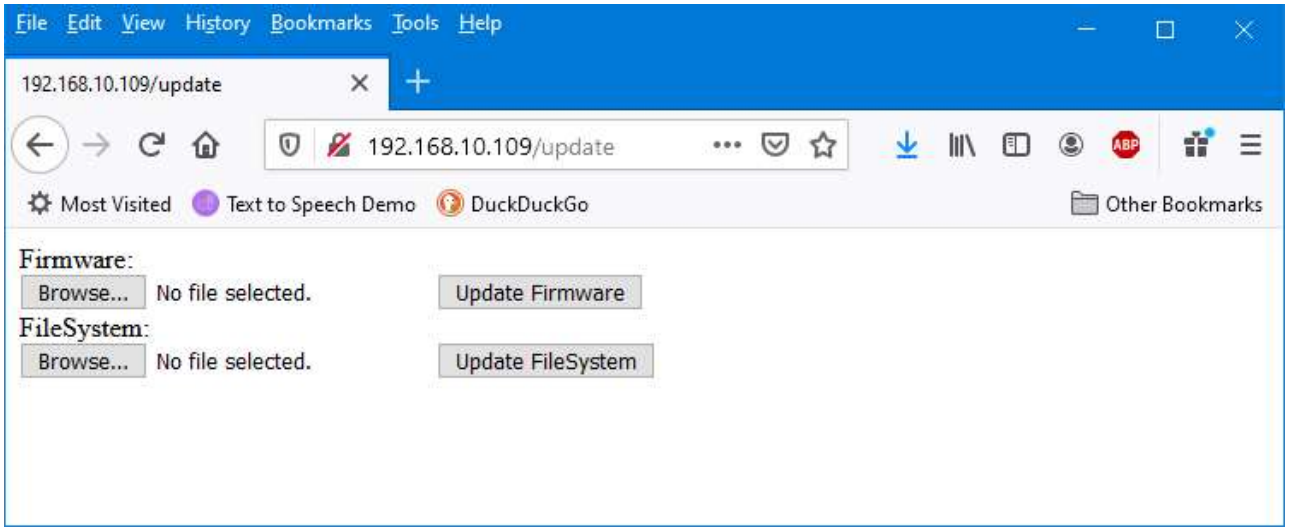
Line 40: instruct SAM to start the update server (this blocks all other functionality)

Line 50-60: get all data from SAM over the serial port and display it on the screen of your CBM. This way you'll get some feedback regarding the wifi connection state and update progress. Also on the screen you'll see the IP-address to which you should go to with your browser in order to do the update.

If this programs doesn't work the first time, reset everything (computer off for 10 seconds) and try again (please double check your SSID and password settings before retrying). Sometimes wifi networks can be a bit stubborn and don't want to connect the 1st time, so no reason to panic. This can also be because of range limitations or a crowded network. Therefore it is preferred to do this action at home and close to your wifi router. Do not try this at a public network as these are very slow and unreliable.

Regarding security, SAM is normally not connected to the wifi network, so you do not need to worry about other people accessing it and updating your firmware without your knowledge. The wifi connection is only established when you give SAM the -CONFIG UPDATE command.

If all goes well, the basic program will print an IP-address that you can enter in your webbrowser. It will look like the screenshot below.



The screen offers you two options, you can update the firmware of SAM. And if you’d need to, you can also update the filesystem of SAM. Now the firmware is pretty obvious, it is the code inside SAM that defines it’s functionality. And therefore this is the most likely thing to update in case there is a new firmware available in case of a bugfix.

The filesystem update button should normally not be used, the filesystem is a very small solid-state-drive inside SAM, it holds the configuration file. In the future it may hold many more files for extra functionality of SAM… or it may not, who will tell?

Both file-types have the same extension, so make sure to upload the correct file. After pressing the "update" button, it may appear that nothing happens, however, that is not the case, the file is transferred and after approximately a minute the system will reset itself and the update is finished. Because SAM has reset itself, the webserver will be closed and the webpage cannot refresh itself anymore, so your browser will notify you that the connection has been lost. This is completely normal. You may use SAM as if you have just switched it on as the new update is now finished.

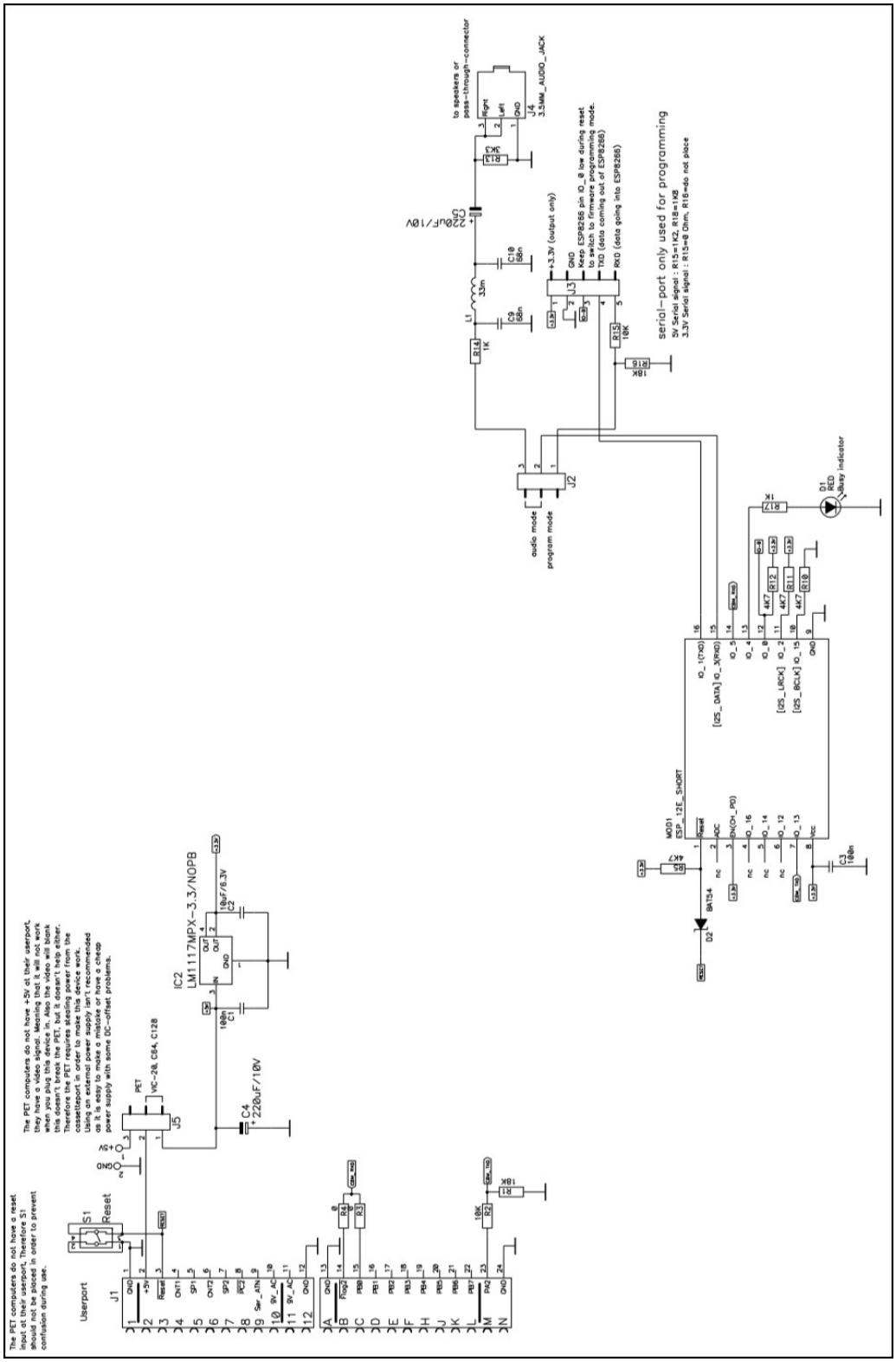
Attention:

If you accidentally upload filesystem.bin into the firmware area, then the file is transferred but rejected before the actual program starts, so nothing happens.

If you upload firmware.bin into the filesystem area, the filesystem is ruined, however when the system starts, it detects the error situation and recreates a new filesystem with the default values, restoring basic functionality, which means that it will create a new setting file with default values. So essentially, all your custom settings are lost as they will be restored to factory default values.

This means that it is highly unlikely that you "brick" your device by pressing the wrong buttons. However, this doesn't mean that you should not think about what you are doing, updating the firmware or the filesystem is a potentially dangerous procedure. If you are uncertain about your abilities of performing this procedure then always ask for help.

10.5 Schematic

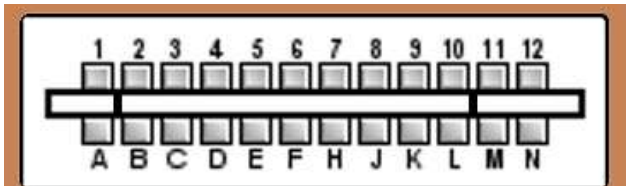


10.6 User port connections

All 8-bit Commodore computers have many similarities but also lot's of (tiny) differences, so beware. Because things that are called and look the same, are not always the equal. For example the user-port, below the pinout of user-ports for various Commodore computer models. The image below shows what you would see if you'd stand behind the computer and look at it's rear side. Please note the notches in the PCB material of the connector, located between pin A-B en L-M. These are keying notches, the idea is that the mating connector should have "keys" inserted at these locations and will prevent the user from inserting the connector upside down. Unfortunately, in practice these "keys" were almost always left out of the mating connector. As a result, many users, damaged their equipment by accidentally inserting connectors upside down into the user-port.

But even if used, these keying pins aren't perfect themselves. Although edge connectors are relatively easy to be bought, the "keys" aren't, even if you manage to find them it is no guarantee that they'll fit properly. Too loose simply mean that they will fall out when least expected, but to tight and they will not fit at all or completely prevent you from inserting the connector into the computer. And even when you think they fit just fine, don't be surprised if the keying pin(s) get caught in the computer and stay there without you even noticing. Therefore, considering all this combined with the fact that the device should be self explanatory on how it should be inserted (with it's label on top, clearly visible during use) this device has no keying pins.

But, this does mean that it is possible to insert the device into the user-port upside down! Please do not do this! You will cause damage, as this device will short-out the 9V AC connections on the user port. Eventually resulting in a blown fuse inside your CBM computer.



pinout as seen when looking directly at the rear end of the CBM computer

(please notice the keying pin slots, which can be used as extra visual reference)

User-port pinout of Commodore computers					
Pin	**VIC-20**	**C64 / C128**	**Plus 4** (C16 has no userport)	**PET**	**Used by SSSAM**
1	GND	GND	GND	GND	**GND**
2	+5V	+5V	+5V	TV Video	**+5V**
3	\Reset	\Reset	\Reset	IEEE-SRQ	**\Reset**
4	Joy-0	CNT1	P2	IEEE-EOI	
5	Joy-1	SP1	P3	Diagnostic sense	
6	Joy-2	CNT2	P4	Tape read-1	
7	Lightpen	SP2	P5	Tape read-2	
8	Tape switch	\PC2	RxC	Tape Write	
9	Serial ATN in	Serial ATN in	Serial ATN in	TV Vertical sync.	
10	9V AC (+ phase)	9V AC (+ phase)	9V AC (+ phase)	TV Horizontal sync.	
11	9V AC (- phase)	9V AC (- phase)	9V AC (- phase)	GND	
12	GND	GND	GND	GND	**GND**
A	GND	GND	GND	GND	**GND**
B	CB1	\Flag2	P0	CA1	**CBM RXD**
C	PB0	PB0	RxD	PB0	**CBM RXD**
D	PB1	PB1	RTS	PB1	
E	PB2	PB2	DSR	PB2	
F	PB3	PB3	P7	PB3	
H	PB4	PB4	DCD	PB4	
J	PB5	PB5	P6	PB5	
K	PB6	PB6	P1	PB6	
L	PB7	PB7	DSR	PB7	
M	CB2	PA2	TXD	CB2	**CBM TXD**
N	GND	GND	GND	GND	**GND**

10.7 Design considerations

During the design of this device some things just didn't make it and were decided not to be placed at the very last moment. But on close inspection of the PCB, you may notice these empty places. Below a short explanation of the components:

LED

For visual indication of SAM being busy (which is the case when SAM speaks) an LED was designed into the system but that never made it into the final product for various reasons. One of those reasons was that you hardly see it, since it is at the back of the computer and the front of the tiny case is almost completely occupied by the logo sticker, making the sticker smaller wasn’t an option considering the sticker is already very small for the information it needs to contain.

Reset button

Another thought was to add a reset button to the system, this way some extra functionality is added that may come in handy for systems that don't have a reset button. However, this didn't make it into the final product for various reasons. One of the main reasons is that the reset-pin isn't available on the user-port of all Commodore computers, for example the PET series have a different function on that pin. So to prevent confusion, it was better to leave out this functionality in order to prevent different variations of the case (with or without reset button). A secondary reason was the very limited available space on the PCB inside the case and the size and shape of the case itself.

Case desgin

A major contributor for leaving out the LED and reset button is the fact that both items need holes. Considering that holes can be very complicated, due to the required tolerances to make everything fit snugly and look nice. Keep in mind that this is a hobby project, every hole is drilled manually. This device would be far more complicated if it had 3 holes. Because the holes have thing sticking through (LED, reset button, Audio jack) all holes or components should be placed exactly at the right location. If a hole or component was to be placed 0.1mm to the left or right, than it might cause in a reset button getting stuck. And making the hole larger to allow for greater error would only result in making things optically worse. Just imagine, an audio jack/button/LED with a large gap around it, or even worse, a gap on the left but a tight fit on the right. This would make the entire device look very cheap and ugly. Therefore a design that only has one hole makes things much easier to fabricate and assemble.

11 Trouble shooting

Sometimes things don't just work the way you expect it to work. Below are some situations you may encounter and how you should respond to them.

11.1 Safety first

Regarding retro computer safety, never plug something in or remove something from a computer that is switched on. Always switch the computer power off before making/breaking any connection.

During maintenance or repair, always be careful with possible static discharges, therefore you should take your precautions and properly ground yourself, the best way to do this is by using a wrist strap. Connect this strap to the bare metal parts of the chassis of the computer you are working on or to the earth of your electrical system. The wrist strap has a 1MOhm resistor that prevents damage to your computer due to Electro Static Discharge (ESD). The strap safely brings you and the computer you are working on, to the same potential, making discharges due to handling impossible. Please consult an expert if you are not familiar with ESD.

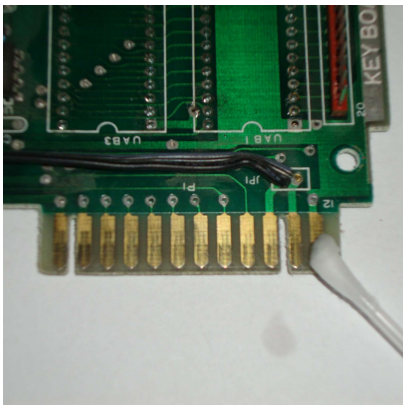
11.2 Bad contacts and how to clean

Many times old computers suffer from bad connections, bad connections between the computer and it's peripheral. These bad connections can be caused by a connector not placed properly or by contacts being dirty. The latter is not surprising considering the age of these machines.

Cleaning contacts with alcohol and cotton swaps

Dirt can be removed in various ways, but we must always be careful how to handle these machines, therefore follow the ESD safety precautions and if you are not familiar with these consult an expert.

Contact cleaning can be done using some cleaning alcohol and a cotton swap, make sure that after cleaning all contacts have dried completely before switching on your computer.



Cleaning contacts with a pencil eraser

A more thorough way of cleaning contacts is by using a pencil eraser. These are slightly abrasive and allow you to remove the most stubborn dirt or oxides. The eraser allows you to gently "grind" the contacts. Do not press hard on the eraser when you touch the pads, use very little force and constantly check for the effect your action has. Sometimes even one gently swipe of the eraser can be enough to clean a pad, very dirty pads may need multiple swipes. Then again if your pads are really dirty, it is best to clean them with a little bit of alcohol and some cotton swaps first. A pencil eraser may leave some gummy residue onto the contacts, which is easily removed with a cotton swap.

Attention: be extremely careful and make absolutely sure not to overdo it because it is very easy to remove too much. Removing too much of the gold plating of the contacts, resulting in contacts corroding even faster then before, making it very unreliable and require you to clean it even more often. Although (technically) not all is lost when this happens as it is possible to re-apply the gold plating using electroplating and some very dangerous chemicals, but that process way beyond the scope of this manual and should be avoided at all times. Keep in mind that in most cases re-applying the gold plating may cost you even more then your computer is worth.

Cleaning contacts with very fine sandpaper

At first this may sound like a reasonable idea, but believe me IT IS NOT. This is in fact the most destructive method possible. The gold plating on the contacts is so very thin that damage is done almost instantly. So… **DO NOT** USE SANDPAPER TO CLEAN CONTACTS!!!

11.3 Compatibility issues

If you use SAM on a C64 in combination with a Final Cartridge III (FC-III), then you may experience problems with SAM not recognizing your commands. For some reason the cartridge messes up the serial port data making it impossible to make SAM do things as long as the cartridge is active. You do not need to remove the FC-III completely, you can disable the cartridge by using a BASIC command named "KILL". Just type KILL <enter> and the cartridge is disabled, now SAM can be controlled normally.

11.4 Error. Serial buffer overflow

When you send SAM too much data the serial buffer WILL overflow. SAM does not like this and will speak the message "Error. Serial buffer overflow" and SAM will discard all the data that has been received up to that point. Meaning that you will loose data. But there is a simple solution, don't send SAM too much data. Give SAM time to pronounce the text you request SAM to speak. When SAM is ready for more text SAM will send the OK-message, so in your code you could be polling for this message before sending more text.

11.5 Why double spaces act like CR

When you send SAM a sentence that has two spaces in between words (or at the end of a sentence), SAM will process these two spaces as if it was a single CR (carriage return). Now this may seems a little strange perhaps, but then again, why would you put extra spaces in a sentence? Because if you need a pause, why not use a comma or for a longer pause use a CR?

But strange or not, this functionality is a direct result of the fact that the VIC-20 Scott Adams adventure games don’t send the CR in a proper manor. Meaning that these games send double spaces where a CR would be required. Which in turn would make SAM sound very awkward if not handled properly. By detecting the “double spaces” situation and processing them as if they were a single CR, everything is perfectly fine. And the Scott Adams adventure games work perfectly in combination with SAM.

11.6 Properly entering commands

In this manual you'll see some examples being given, for example the speed command:

```
-CONFIG SPEED10
```

Now you can't just type this in on your CBM, it doesn't work that way, you'll need to send this command (which is nothing more then plain text) to the serial port using the following method. First you must open the serial port channel at 2400 baud using the BASIC commands:

```
OPEN 1,2,3,CHR$(10)
```

Then when the port is opened, you may send the command through the use of the print statement:

```
PRINT#1,"-CONFIG SPEED10"
```

If more commands or text needs to be send, just use more PRINT# statements, since the serial port is already open. You may close the port when done, but this isn't really required. More details about the OPEN, PRINT# and CLOSE# command is shown below.

11.7 Startup message seems to be delayed

If you experience the startup message to be delayed, then this is most likely caused by the presence of a dictionary. Upon startup the system scans the file-system for dictionaries and makes an index in order to be able to scan the dictionary realtime during speech. The longer the dictionary, the longer the scanning, the longer the perceived delay upon startup. Remove or rename the dictionary files, so that these are no longer used in order to solve this problem completely. By reducing the size of the dictionary the perceived delay will also be reduced. Always keep in mind if you really need a dictionary.

11.8 The OPEN statement

The Serial Speech Synthesizer SAM device is connected to the TTL RS-232 serial port of the computer, in order to use this from the BASIC command line (or from within a program) all you need to do is to open the serial port at the correct baudrate.

```
OPEN 1,2,3,CHR$(10)
```

1 = the channel used for communication, CHR$(10) = 2400 baud

When the serial port is opened it remains open until closed by the CLOSE command. If you do not want to use channel 1, you are free to use any other channel within the limits of the CBM's functionality, as long as you use that same number for the PRINT and CLOSE statement further on.

11.9 The PRINT# statement

The Serial Speech Synthesizer SAM device is connected to the TTL RS-232 serial port of the computer, in order to use this from the BASIC commandline (of from within a program) a simple PRINT# statement is all you need. Although, you must make sure that the port is opened and that the same channel number is used as when the port was opened.

`PRINT#1,"HELLO WORLD"`	1 = the channel used for communication
	"HELLO WORLD" = the text that will be spoken.
`PRINT#1,"-CONFIG SPEED10"`	1 = the channel used for communication
	"-CONFIG SPEED10" = the CONFIG command
`PRINT#1,"-DEMO"`	1 = the channel used for communication
	"-DEMO" = this DEMO command (starts the DEMO)

11.10 The CLOSE# statement:

You can open the serial port for use at with the OPEN command (see above) and when this functionality is no longer required, you may close it with the command:

`CLOSE#1`	1 = the channel that needs to be closed

Attention: when opening ports on your CBM, for example for communication with your printer or disk drive or datasette, it is very normal to close it when you are done. So most programs have near it's end a `CLOSE#X` instruction, to close the opened port (multiple CLOSE#X commands are required when multiple ports are used).

Unfortunately, using the serial port things work slightly different. Because if you give the CLOSE command BEFORE all data is completely send over the serial port, the CLOSE command will instantly abort the transmission of data. This may result in a SAM speaking only a part of a sentence or in no speech at all. Therefore wait until all data is send before closing OR in most cases (which is the easiest) just leave out the CLOSE command entirely (the close command related to the serial port channel that is). As you may have noticed, all the examples in this manual do not use any close statement at all.

11.11 The CMD statement (redirecting output)

Commodore BASIC has a command that allows for easy redirecting of the standard output. This way it is possible to output the text generated by a print statement (which normally goes to the screen) to the disk drive, printer or serial port (RS-232). The serial (RS-232) port is the port to which SAM is connected. So we can output the data generated by print, list, etc. to SAM using the following commands

`OPEN 1,2,3,CHR$(10)`	1 = channel, 2 = serial port, CHR$(10) = 2400 baud
`CMD 1`	redirect standard output to channel 1
`PRINT"hello"`	send text to the (redirected) standard out

You can stop the redirectioning of the output using an empty print command or by generating an error (just type something that causes a syntax error). Because error printing forces the output back to the default output (which is the screen).

```
PRINT#1
```

The above few lines cause the text “hello” to be send to the serial port. Now unfortunately… this is only a partial redicrection of outputted data, which is a bit confusing. Because everything you type is still visible on the screen and not send to the redirected port. In other words, if you are blind and want to use redirectioning of text output to the serial port, in order to hear what you are typiong. Then this form of output redirectioning is completely useless. For those situations you’ll need a special kernal that performs that task. At the moment of writing the name of such a kernal is not known by the author, but it is known that they exist. Please refer to, the user manual or programmers reference guide of your CBM computer, for more details about CMD.

12 Experimental functionality

Because this device is build around an ESP8266, it means that it can access the local WiFi network. This functionality is already in use as a method for easily updating the firmware. But wouldn't it be nice if it could be used for more functions? Well, experiments have been done and a Telnet client has been implemented… but to be honest it isn't really fast. Which is one of the reasons that these telnet related commands aren't mentioned in the command overview. Technically these commands are not supported, if you use them… good luck.

There are various reason for this. One of them is the serial port used to send data back to the CBM computer. This serial port is a bit-banged serial port. Because the normal serial port is not usable because it's IO-pin is already being used to generate the analog audio signal. Now although the bit-banged serial port is very accurate, it also means that is is very time consuming meaning that it will lock up SAM during the processing of that data.
Now it doesn't help that the CBM can only process data rates of up to 2400 bits/sec (faster is possible using assembly, but for a simple BASIC program 2400 is the absolute limit). And even then it is a bit on the fast side, because if a BASIC program is processing that data (printing it to the screen) then this will also take time and the buffer on the CBM is only 255 bytes. Meaning that if this buffer is fed more data than it can handle it will simply overflow and data will get lost. All of this combined with the fact that some BBS (Bulletin BoardS) can respond with large amounts of data, buffer overflows are bound to happen and data will be lost. Because there is no hardware handshaking between SAM and the CBM although some other form of handshaking could be realized, this has not been done yet. Therefore the only way to make this works at this moment, is to slow down that data towards the CBM. This can be done using the SERIALDELAY command, which considerably can slow communications down, but allowing the CBM to keep up.

However, if you want to experiment these are the commands that can be used. They are shown in the form of an example. Keep in mind that using this commands are at your own risk, there are no guarantees that it will work and success varies depending on the BBS visitted. The example below tries to connect to the nice BBS named "borderlinebbs.dyndns.org" at port 6400. In order to stop the program press run/stop on your CBM, but in order to gain full control over SAM you must reset your system.

```
10 open 1,2,3,chr$(10)
15 print#1,"-config debug0"
20 print#1,"-config ssid WiFinetwork"
30 print#1,"-config pass WiFipassword"
40 print#1,"-config serialdelay10"
50 print#1,"-config telnethost borderlinebbs.dyndns.org"
51 print#1,"-config telnetport 6400"
52 print#1,"-config telnetstart"
70 get#1,a$:if a$="" then 72
```

```
71 print a$;
72 get b$:if b$="" then 74
73 print#1b$;
74 goto 70
```

Line 10: open serial port.

Line 15: disables SAM debug mode, meaning that the -config related commands will not be pronounced by SAM. This speeds up the execution of the program but also prevent user annoyances/frustration. This because SAM will most likely not be able to pronounce the name of your wifi network, password, the URL of the BBS, etc.

Line 20-30: send the wifi's SSID and password information (enter your personal values here, replace the text "WiFinetwork" with the name of the wifi network you want to connect with. And replace the text "WiFipassword with the password of the wifi network you want to connect with. Make sure that if your password has capital letters, that you enters these values in capital form, press shift+CBM to switch to lower case mode in order to visualize that text in normal form (upper and lower case).

Line 40: slow down the serial traffic in order to prevent an overflow of data on the CBM side, a value of 10 will add a delay time with an equivalent of 10 characters. Effectively this will make the communication 11 times slower than normal. It will allow the CBM a bit more time to process the received characters.

Line 50-52: parse the telnet host information to SAM and start a connection

Line 70-71: get all data from SAM over the serial port and display it on the screen of your CBM. This way you'll the data send by the telnet host will be displayed on your screen. If the telnet host is a supports the PETSCII charset then you'll automatically have color and control characters as these are implemented in the PETSCII characterset. This means that the BBS can send you simple but colorful welcome screen PETSCII images which you can display with the simple BASIC program as shown here. As the true magic lies within the PETSCII character set combined with the "print a$" command.

Line 72-73: get all data from the user (keyboard) and send it to the SAM over the serial port so it will be send to the telnet host.

Line 74: keep looping in order to continuously scan fro data from the telnet host or the user.

13 English-to-phonetic spelling dictionary

- NUMBERS -

one = WAH4N
two = TUW4
three = THRIY4
four = FOH4R
five = FAY4V
six = SIH4KS
seven = SEH4VIXN
eight = EY4T
nine = NAY4N
ten = TEH4N
eleven = IXLEH4VIXN
twelve = TWEH4LV
thirteen = THER4TIY6N
twenty = TWEH4NTIY
thirty = THER4TIY
hundred = /HAH4NDRIXD
thousand = THAW4ZUND
million = MIH4LYUN

- UNITS -

units = YUW4NIXTS
inches = IH4NCHIXZ
feet = FIY4T
yards = YAA4RDZ
miles = MAY4LZ
centimeters = SEH4NTIXMIY6TERZ
kilometers = KIXLAA4MIXTERZ
acres = EY4KERZ
ounces = AW4NSIXZ
pounds = PAW4NDZ
tons = TAH4NZ
grams = GRAE4MZ
teaspoons = TIY4SPUWNZ
cups = KAH4PS
pints = PAY4NTS
quarts = KWOH4RTS
gallons = GAE4LUNZ
liters = LIY4TERZ
degrees = DAXGRIY4Z

- DAYS OF THE WEEK -

Monday = MAH4NDEY

Tuesday = TUW4ZDEY

Wednesday = WEH4NZDEY

Thursday = THER4ZDEY

Friday = FRAY4DEY

Saturday = SAE4TERDEY

Sunday = SAH4NDEY

- MONTHS OF THE YEAR -

January = JAE4NYUXEHRIY

February = FEH4BRUXEH6RIY

March = MAA4RCH

April = EY4PRIXL

May= MEY4

June = JUW4N

July = JUHLAY4

August = AO4GAXST

September = SEHPTEH4MBER

October = AAKTOW4BER

November = NOHVEH4MBER

December = DIHSEH4MBER

- STATES AND PROVINCES -

United States = YUWNAY4TIXD STEY4TS
Alabama = AE4LAXBAE6MAX
Alaska = AHLAE4SKAH
Arizona = EH4RAXZOW5NAH
Arkansas = AA4RKUNSAO
California = KAE5LAXFOH4RNYAH
Colorado = KAA5LAXRAA4DOW
Connecticut = KAHNEH4TIXKAHT
Delaware = DEH4LAXWEH6R
Florida = FLOH4RIXDAH
Georgia = JOH4RJAH
Hawaii = /HAHWAY4IY
Idaho = AY4DAH/HOW
Illinois = IHLUNOY4
Indiana = IH5NDIYAE4NAH
Iowa = AY4AHWAH
Kansas = KAE4NZIXS
Kentucky = KEHNTAH4KIY
Louisiana = LUXIY4ZIYAE5NAH
Maine = MEY4N
Maryland = MEH4RULIXND
Massachusetts = MAE5SAXCHUW4SIXTS
Michigan = MIH4SAXGUN
Minnesota = MIH5NAXSOW4TAH
Mississippi = MIH5SIXSIH4PIY
Missouri = MIHZUH4RIY
Montana = MAANTAE4NAH
Nebraska = NAXBRAE4SKAH
Nevada = NAXVAE4DAH
New Hampshire= NUW6/HAE4MPSHER
New Jersey = NUWJER4ZIY
New Mexico = NUWMEH4KSIXKOW
New York = NUWYOH4RK
North Carolina = NOH4RTH KEH5RULAY4NAH
North Dakota= NOH4RTH DAHKOW4TAH
Ohio = OW/HAY4OW
Oklahoma = OWKLAX6/HOW4MAH
Oregon = OH4RIXGUN
Pennsylvania = PEH5NSULVEY4NYAH
Rhode Island = ROW5D AY4LUND
South Carolina = SAW4TH KEH5RULAY4NAH
South Dakota = SAW4TH DAXKOW4TAH
Tennessee = TEH5NAXSIY4
Texas = TEH4KSAXS
Utah = YUW4TAO6
Vermont = VERMAA4NT
Virginia = VERJIH4NYAH
Washington = WAA4SHIHNXTAHN
West Virginia = WEH5ST VERJIH4NYAH
Wisconsin = WIHSKAA4NSUN
Wyoming = WAYOW4MIHNX
Provinces of Canada =
PRAA4VIXNSIXZ AHV KAE4NAXDAH
Alberta = AELBER4TAH
Brit. Col.= BRIH4TIXSH KAHLAH4MBIYAH
Manitoba = MAE5NIXTOW4BAH
New Brunswich = NUWBRAH4NZWIXK
Newfoundland = NUW4FIXNLIXND
Nova Scotia = NOH4VAX5KOW4SHAH
Ontario = AANTEH4RIYOW
Pr. Edward Isl = PRIH5NS EH4DWERD AY4LUND
Quebec = KUHBEH4K
Saskatchewan = SAESKAE4CHAXWAAN

- A -

abandon = AHBAE4NDUN
ability = AHBIH4LIXTIY
able = EY4BUL
abort = AHBOH4RT
about = AHBAW4T
above = AHBAH4V
absolute = AE5BSOHLUW4T
abuse = AHBYUW4S
accelerate = EHKSEH4LEREYT
accent = AE4KSEHNT
accept = AEKSEH4PT
access = AE4KSEHS
accident = AE4KSIXDEHNT
account = AHKAW4NT
acknowledge = EHKNAW4LIHJ
action = AE4KSHUN
active = AE4KTIHV
address = AE4DREHS
adjust = AHJAH4ST
adult = AHDAH4LT
advance = EHDVAE4NS
adventure = AEDVEH4NCHER
affair = AHFEY4R
afford = AHFOH4RD
after = AE4FTER
age = EY4J
agree = AHGRIY4
air = EH4R
airplane = EH4RPLEYN
alarm = AHLAA4RM
algebra = AE4LJAXBRAH
alien = EY4LIYIXN
allow = AHLAW4
alone = AHLOW4N
along = AHLAO4NX
alphabet = AE4LFAXBEHT
alternate = AO4LTERNIXT
America = AHMEH4RIXKAH
among = AHMAH4NX
analysis = AHNAE4LIXSIXS
anger = AE4NXGER
announce = AHNAW4NS
answer = AE4NSER
antenna = AENTEH4NAH
anticipate = AENTIH4SIXPEYT
apology = AHPAA4LAXJIY
appear = AHPIY4R
apple = AE4PUL
appropriate = AHPROH4PRIYIXT
approve = AHPRUW4V
area = EH4RIYAH
arm = AA4RM
arrive = AHRAY4V
ask = AE4SK
assumption = AHSAH4MPSHUN
astronomy = AHSTRAA4NUMIY
Atari = AHTAA4RIY
atom = AE4TUM
attack = AHTAE4K
audio = AO4DIYOW
authority = AHTHOH4RIXTIY
automatic = AO5TUMAE4TIXK
auxiliary = AOKZIH4LYERIY
available = AHVEH4LAXBUL

- B -

baby = BEY4BIY
back = BAE4K
bad = BAE4D
balance = BAE4LIXNS
bank = BAE4NXK
bargain = BAA4RGUN
base = BEY4S
basic = BEY4SIHK
battle = BAE4TUL
beam = BIY4M
beautiful = BYUW4TIXFUHL
behave = BIY/HEY4V
belief = BIXLIY4F
beneficial = BEH4NAXFIH4SHUL
betray = BIYTREY4
better = BEH4TER
bible = BAY4BUL
bibliography = BIH5BLIYAA4GRAXFIY
bicycle = BAY4SIXKUL
billion = BIH4LYUN
binary = BAY4NEHRIY
bite = BAY4T
black = BAE4K
blast = BLAE4ST
block = BLAA4K
blood = BLAH4D
board = BOH4RD
bomb = BAA4M
book = BUH4K
boot = BUW4T
boss = BAO4S
bottle = BAA4TUL
bottom = BAA4TUM
box = BAA4KS
boy = BOY4
brain = BREY4N
branch = BRAE4NCH
break = BREY4K
brief = BRIY4F
bring = BRIH4NX
broken = BROW4KIXN
brother = BRAH4DHER
budget = BAH4JIXT
buffer = BAH4FER
bug = BAH4G
bureau = BYER4OW
burglar = BER4GULER
bus = BAH4S
business = BIH4ZNIXS
busy = BIH4ZIY
by = BAY4
byfe = BAY4T

- C -

cabinet = KAE4BUNIXT
cable KEY4BUL
calculate = KAE4LKYAXLEYT
calendar = KAE4LUNDER
call = KAO4L
calorie = KAE4LERIY
cancel = KAE4NSUL
candy = KAE4NDIY
cant = KAE4NT
capacity = KAXPAE4SIXTIY
captain = KAE4PTIXN

capture = KAE4PCHER
card = KAA4RD
careful = KEH4RFUHL
carry = KEH4RIY
cartridge = KAA4RTRIXJ
case = KEY4S
cashier = KAE4SHIY4R
cassette = KAXSEH4T
catalog = KAE4TULAOG
celebrate = SEH4LAXBREYT
celestial = SULEH4SCHIYUL
Celsius = SEH4LSIYAXS
center = SEH4NTER
certain = SER4TQN
challenge = CHAE4LIXNJ
change = CHEY4NJ
channel = CHAE4NUL
chapter = CHAE4PTER
charge = CHAA4RJ
chauvenism = SHOH4VIXNIHZUM
Cheese = CHIY4Z
child = CHAY4LD
children = CHIH4LDRIXN
chocolate = CHAO4KLIXT
choreography = KOH5RIYAA4GRAXFIY
Christmas = KRIH4SMAXS
church = CHER4CH
cinema = SIH4NUMAH
circle = SER4KUL
circuit = SER4KIXT
circumstance = SER4KUMSTAENS
citizen = SIH4TIXSUN
city = SIH4TIY
classify = KLAE4SIXFAY
clear = KLIY4R
close = KLOW4Z
coaxial = KOHAE4KSIYUL
coffee = KAO4FIY
coherent = KOW/HEH4RIXNT
cold = KOW4LD
college = KAA4LIXJ
color = KAH4LER
comfortable = KAH4MFTERBUL
command = KUMAE4ND
common = KAA4MUN
company = KAHM4PUNIY
complain = KUMPLEY4N
complex = KUMPLEH4KS
component = KAHMPOH4NUNT
computer = KUMPYUW4TER
condition = KUNDIH4SHUN
conscience = KAA4NSHUNTS
console = KAA4NSOHL
control = KUNTROH4L
conversation = KAA5NVERSEY4SHUN
coordinate = KOHWOH4DUNIXT
corporation = KOH5RPEREY4SHUN
correction = KOHREH4KSHUN
count = KAW4NT
country = KAH4NTRIY
cousin = KAH4ZIXN
create = KRIYEY4T
critical = KRIH4TIXKUL
culture = KAH4LCHER
curious = KYUH4RIYAXS

- D -

danger = DEY4NJER
data = DEY4TAH
decay = DIXKEY4
decibel = DEH4SIXBUL
decrease = DIYKRIY4S
definition = DEH5FUNIH4SHUN
degree = DIXGRIY4
delay = DIXLEY4
demonstrate = DEH4MUNSTREYT
department = DIYPAA4RTMIXNT
desire = DIXZAY4ER
develop = DIXVEH4LAHP
dictionary = DIH4KSHUNEHRIY
different = DIH4FRIXNT
discount = DIH4SKAWNT
distance = DIH4STIXNS
distribution = DIH5STRAXBYUW4SHUN
division = DIXVIH4ZHUN
doctor = DAA4KTER
double = DAH4BUL
down = DAW4N
drive = DRAY4V
dungeon = DAH4NJUN

- E -

earth = ER4TH
easy = IY4ZIY
economics = IY5KUNAA4MIXKS
education = EH5JUWKEY4SHUN
either = IY4DHER
eject = IXJEH4KT
electricity = ULEHKTRIH4SIXTIY
electronic = ULEHKTRAA4NIXK
elementary = EH4LUMEH4NTRIY
emphasis = EH4MFAXSIHS
encyclopedia=EHNSAY5KLAXPIY4DIYAH
energy = EH4NERJIY
engineering = EH5NJUNIY4RIHNX
enter = EH4NTER
enunciate = IYNAH4NSIYEYT
equal = IY4KWUL
erase = IXREY4S
error = EH4ROHR
escape = EHSKEY4P
estimate = EH4STUMIXT
Europe = YUH4RAXP
evil = IY4VUL
exciting = EHKSAY4TIHNX
explain = EHKSPLEY4N
expression EHKSPREH4SHUN
extra = EH4KSTRAH

- F -

face = FEY4S
fail = FEY4L
Fahrenheit = FEH4RIXN/HAYT
false = FAO4LS
family = FAE4MULIY
fast = FAE4ST
fatal = FEY4TUL
father= FAA4DHER
fault = FAO4LT
female = FIY4MEYL
fight = FAY4T
figure = FIH4GYER

file = FAY4L
filter= FIH4LTER6
finance = FAY4NAENS
find = FAY4ND
finger = FIH4NXGER
finish = FIH4NIXSH
fire = FAY4ER
first = FER4ST
flavor = FLEY4VER
flight = FLAY4T
flow chart = FLOW4CHAART
flower = FLAW4ER
fluorescent = FLUHREH4SIXNT
focus = FOW4KAXS
follow = FAA4LOW
foot = FUH5T
force = FOH4RS
formula = FOH4RMYUXLAH
forward = FOH4RWERD
fraction = FRAE4KSHUN
fragile = FRAE4JUL
freedom = FRIY4DUM
frequency = FRIY4KWUNSIY
from = FRAH4M
fuel = FYUW4L
full = FUH4L
function = FAH4NXKSHUN
fundamental = FAH5NDUMEH4NTUL
fuse = FYUW4Z
fusion = FYUWSZHUN
future = FYUW4CHER

- G -

gain = GEY4N
galaxy = GAE4LAXKSIY
game = GEY4M
garbage = GAA4RBIXJ
gasoline = GAE4SULIYN
gate = GEY4T
general = JEH4NERUL
generate = JEH4NEREYT
genius = JIY4NYAXS
gentle = JEH4NTUL
genuine = JEH4NUYXIXN
geometry = JIYAA4MIXTRIY
get = GEH4T
giant = JAY4IXNT
gift = GIH4FT
glass = GLAE4S
gnome = NOW4M
go = GOW4
gold = GOH4LD
good = GUH4D
gourmet = GUHRMEY4
government = GAH4VERNMEHNT
grand = GRAE4ND
graphic = GRAE4FIXK
gravity = GRAE4VIXTIY
ground = GRAW4ND
guarantee = GAE4RIXNTIY4
guide = GAY4D
gun = GAH4N
gyroscope = JAY4RAXSKOWP

- H -

habit = /HAE4BIXT
hacker = /HAE4KER
hair = /HEH4R
half = /HAE4F
hallucination = /HULUW4SIXNEY5SHUN
hand = /HAE4ND
happy = /HAE4PIY
hardware = /HAA4RDWEHR
harmony = /HAA4RMUNIY
have = /HAE4V
head = /HEH4D
heart = /HAA4RT
helicopter = /HEH4LIXKAAPTER
hello = /HEH4LOW
here = /HIY4R
hero = /HIY4ROW
herta = /HER4TS
hesitate = /HEH4ZIXTEY6T
hexadecimal = !HEH5KSIXDEH4SUMUL
high = /HAY4
history = /HIH4STERIY
hobby = /HAA4BIY
hold = /HOW4LD
home = /HOW4M
honest = AA4NIXST
horoscope = /HOH4RAXSKOWP
hospital = /HAA4SPIXTUL
hour = AW4ER
house = /HAW4S
however = /HAWEH4VER
huge /HYUW4J
human = /HYUW4MUN
humor = /HUYW4MER
husband = /HAH4ZBUND
hyper = /HAY4PER
hypothesis = /HAYPAA4THAXSIHS

- I -

I = AY4
ice = AY4S
idea = AYDIY4AX
identical = AYDEH4NTIXKUL
identity = AYDEH4N11XTIY
illusion = IHLUX4ZHUN
image = IH4MIXJ
imagination = IHMAE4JIXNEY5SHUN
immobilize = IXMOH4BULAYZ
important = IHMPOH4RTUNT
in = IH4N
inch = IHN4CH
included = IHNKLUX4DIXD
income = IH4NKUM
inconvenient = IHN5KUNVIY4NYUNT
increase = IHNKRIY4S
indeed = IHNDIY4D
index = IH4NDEHKS
indicate = IH4NDIXKEYT
indirect = IH5NDEREH4KT
individual = IH5NDIXVIH4JUWUL
industry = IH4NDAHSTRIY
inferior = IHNFIH4RIYER
inflation = IHNFLEY4SHUN
influence = IH4NFLUWIXNS
information = IH5NFERMEY4SHUN
-ing = IHNX

inject = IHNJEH4KT
injure = IH4NJER
initial = IXNIH4SHUL
inside = IHNSAY4D
inspect = IHNSPEH4KT
insulator = IH4NSULEYTER
integer = IH4NTIXJER
intelligent = IHNTEH4LIXJIXNT
interest = IH4NTREHST
interference = IH4NTERFIY4RIXNS
intermittent = IH4NTERMIH4TNNT
invader = IHNVEY4DER
invent = IHNVEH4NT
inverse = IH4NVERS
involve = IHNVAA4LV
iron = AY4ERN
irrational = IHRAE4SHUNUL
isolate = AY4SULEYT
issue = IH4SHUW
item = AY4TUM

- J -

jacket = JAE4KIXT
jam = JAE4M
jargon = JAA4RGUN
jazz = JAE4Z
jiffy = JIH4FIY
job = JAA4B
join = JOY4N
joke = JOW4K
judge = JAH4J
jump = JAH4MP
junction = JAH4NXKSHUN
junior = JUW4NYER
just = JAH4ST
jail = JEY4L
jewelry = JUW4LRIY
journey = JER4NIY
jungle JAH4NXGUL
junk = JAH4NXK

- K -

keep = KIY4P
key = KIY4
keyboard = KIY4BOHRD
kilobyte = KIH4LAXBAYT
kind = KAY4ND
kingdom = KIH4NXGDUM
knight = NAY4T
knowledge = NAA4LIXJ

- L -

label = LEY4BUL
lady = LEY4DIY
language = LAE4NXGWIXJ
large = LAA4RJ
laser = LEY4ZER
last = LAE4ST
late = LEY4T
laugh = LAE4F
launch = LAO4NCH
law = LAO4
layer = LEY4ER
lead = LIY4D
lease = LIY4S
lecture = LEH4KCHER

left = LEH4FT
legal = LIY4GUL
legend = LEH4JIXND
leisure = LIY4ZHER
length = LEH4NTH
letter = LEH4TER
level = LEH4VUL
liberal = LIH4BERUL
life = LAY4F
lift = LIH4FT
light = LAY4T
like = LAY4K
limit = LIH4MIXT
linear = LIH4NIYER
liquid = LIH4KWIXD
list = LIH4ST
listen = LIH4SIXN
literature = LIH4TERIXCHER
little = LIH4TU
load = LOW4D
local = LOW4KUL
location = LOWKEY4SHUN
lock = LAA4K
logarithm = LAO4GERIH5DHUM
logical = LAA4JIHKUL
long = LAO4NX
look = LUH4K
loop = LUW4P
lose = LOW4Z
love = LAH4V
low = LOW4
loyal = LOY4UL
luminescence = LUW4MIXNEH5SIXNS
lunatic = LUW4NAXTIH6K
luxury = LAH4GZHERIY

- M -

machine = MAXSHIY4N
madam = MAE4DUM
made = MEY4D
magazine = MAEGAXZIY4N
magic = MAE4JIHK
magnet = MAE4GNIXT
magnitude = MAE4GNIHTUX5D
mail = MEY4L
main = MEY4N
major = MEY4JER
make = MEY4K
malfunction = MAE5LFAH4NXKSHUN
man = MAE4N
manager = MAE4NIXJER
maneuver = MUNUW4VER
manipulate = MUNIH4PYUHLEYT
manual = MAE4NYUWUL
manufacture = MAE5NUYXFAE4KCHER
many = MEH4NIY
marginal = MAA4RJIXNUL
market = MM4RKIXT
marriage = MEH4RIXJ
mass = MAE4S
master = MAE4STER
mate = MEY4T
material = MAXTIH4RIYUL
mathematics = MAE4THUMAE5TIXKS
mature = MAXCHUX4R
maximum = MAE4KSIXMUM

may = MEY4
meaning = MUY4NIHNX
measure = MEH4ZHER
mechanical = MIXKAE4NIHKUL
mechanism = MEH4KUNIHZUM
media = MIY4DIYAH
medical = MEH4DIXKUL
medium = MIY4DIYUM
member = MEH4MBER
memory = MEH4MERIY
mental = MEH4NTUL
menu = MEH4NYUW
merchandise = MER4CHUNDAY5S
merge = MER4J
metal = MEH4TUL
meter = MIY4TER
method = MEH4THIXD
micro = MAY4KROW6
middle = MIH4DUL
might = MAY4T
mile = MAY4L
military = MIH4LIXTEH6RIY
million = MIH4LYUN
mind = MAY4ND
mineral = MIH4NERUL
miniature = MIH4NIYAXCHER
minimum = MIH4NIXMUM
minus = MAY4NIXS
miracle = MIH4RIXKUL
miscellaneous = MIH5SULEY4NIYAXS
missile = MIH4SUL
mister = MIH4STER
mixture = MIH4KSCHER
mnemonic = NIXMAA4NIXK
model = MAA4DUL
modulation = MAA4JULEY5SHUN
molecule = MAA4LIXKYUWL
moment = MOH4MIXNT
money = MAH4NIY
monitor = MAA4NIXTER
monolithic = MAANULIH4THIXK
monotone = MAA4NAXTOW6N
month = MAH4NTH
moon = MUW4N
morning = MOH4RNIHNX
most = MOW4ST
mother = MAH4DHER
motion = MOW4SHUN
motor = MOW4TER
mouth = MAW4TH
move = MUW4V
much = MAH4CH
multiply = MAH4LTIX6PLAY
murder = MER4DER
muscle = MAH4SUL
music = MYUW4ZIXK
must = MAH4ST
my = MAY4
myself = MAYSEH4LF
mystery = MIH4STERIY

- N -

naive = NAY5IY4V
name = NEY4M
narrate = NAE4REYT
narrow = NAE4ROW

natural = NAE4CHERUL
nature = NEY4CHER
navigate = NAE4VIXGEYT
near= NIY4R
need = NIY4D
negative = NEH5GAXTIH6V
negotiate NIXGOW4SHIYEYT
neighborhood = NEY4BER/HUH6D
nerve = NER4V
neutral = NUX4TRUL
news = NUW4Z
nice = NAY4S
night = NAY4T
noise = NOY4Z
nomenclature = NOH4MIXNKLEY6CHER
none = NAH4N
normal = NOH4RMUL
north = NOH4RTH
nose = NOW4Z
notation = NOHTEY4SHUN
notice = NOW4TIXS
nothing = NAH4THIHNX
now = NAW4
nuclear = NUX4KLIYER
number= NAH4MBER

- O -

object = AA4BJEHKT
obligation = AA5BLIXGEY4SHUN
observe = AXBZER4V
obvious = AA4BVIYAXS
occational = AHKEY4ZHUNUL
occupation = AA5KYUXPEY4SHUN
ocean = OW4SHUN
odd = AA4D
of = AH4V
off = AO4F
offer = AO4FER
office = AO4FIXS
official = AHFIH4SHUL
ogre = OW4GER
ohm = OW4M
oil = OY4L
O.K. = OW4KEY
old = OW4LD
omen = OW4MUN
on = AA4N
open = OW4PUN
operate = AA4PEREYT
opinion = AHPIH4NYUN
oppose = AHPOW4Z
opposite = AA4PAXSIHT
option = AA4PSHUN
orbit = OH4RBIHT
orchestra = OH4RKEHSTRAH
order = OH4RDER
ordinary = OH4RDIXNEHRIY
organize = OH4GUNAYZ
origin = OH4RIXJIXN
oscillation = AA5SULEY4SHUN
other = AH4DHER
ought = AO4T
out = AW4T
outlet = AW4TLEHT
output = AW4TPUHT
outside = AWTSAY4D

over = OW4VER
own = OW4N
oxygen = AA4KSAXJIXN

- P -

pack = PAEPAE4K
package = PAE4KIXJ
page = PEY4J
paint = PEY4NT
pair = PEH4R
palace = PAE4LIXS
panel = PAE4NUL
paper = PEY4PER
parabola = PERAE4BULAH
paradox = PAE4RAXDAA6KS
parallel = PAE4RULEH6L
paragraph = PAE4RAXGRAEF
pardon = PAA4RDUN
parent = PEH4RUNT
parity = PAE4RIXTIY
park = PAA4RK
part = PAA4RT
particle = PAA4RTIXKUL
particular = PAARTIH4KYUHLER
pass = PAE4S
patch = PAE4TCH
pathetic = PAHTHEH4TIXK
pattern = PAE4TERN
pause = PAO4Z
pay = PEY4
payroll = PEY4ROW6L
peculiar = PIXKYUW4LYER
penalty = PEH4NULTIY4
penetrate = PEH4NAXTREY6T
perception = PERSEH4PSHUN
perfect = PER4FIXKT
period = PIH4RIYIXD
permanent = PER4MUNIXNT
permission = PERMIH4SHUN
person = PER4SUN
personality = PER4SUNAE5LIX1
perspective = PERSPEH4KTIXV
pet = PEH4T
phantom = FAE4NTUM
phase = FEY4Z
phenomenon = FUNAA4MIXNU
philosophy = FULAA4SAHFIY
phoneme = FOW4NIYM
photo = FOW4TOW
physical = FIH4ZIXKUL
physics = FIH4ZIXKS
piano = PYAE4NOW
pick = PIH4K
picture = PIH4KCHER
pilot = PAY4LIXT
pin = PIH4N
pirate = PAY4RIXT
pistol = PIH4STUL
pitch = PIH4TCH
pity = PIH4TIY
place = PLEY4S
plan = PLAE4N
planet = PLAE4NIXT
plastic = PLAE4STIxK
plausible = PLAO4ZAXBUL
play = PLEY4

please = PLIY4Z
pleasure = PLEH4ZHER
plectrum = PLEH4KTRUM
plenty = PLEH4NTIY
plot = PLAA4T
plug = PLAH4G
plus = PLAH4S
poetry = POW4IXTRIY
point = POY4NT
poke = POW4K
police = PULIY4S
policy = PAA4LIXSIY
polynomial = PAA5LIXNOH4MIYUL
pop = PAA4P
popular = PAA4PYULER
population = PAA4PYULEY4SHUN
port = POH4RT
portable = POH4RTAXBUL
positive = PAA4ZIXTIX6V
position = PAXZIH4SHUN
power= PAW4ER
practice = PRAE4KTIHS
precise = PRIXSAY4S
prefer = PRIXFER4
preliminary = PREIXLIH4MIXNEHRIY
prepare = PRIXPEH4R
present = PREH4ZIXNT
press = PREH4S
pressure = PREH4SHER
prevent = PRIXVEH4NT
primary = PRAY4MEHRIY
primitive = PRIH4MIXTIX6V
prince = PRIH4NS
princess = PRIH4NSEHS
print = PRIH4NT
private = PRAY4VIXT
probably = PRAA4BAXBLIY
problem = PRAA4BLUM
proceed = PROHSIY4D
process = PRAA4SEHS
produce = PRAXDUW4S
professional = PRAXFEH4SHUNUL
professor = PRAHFEH4SER
profit = PRAA4FIXT
program = PROW4GRAEM
project = PRAA4JEHKT
promise = PRAA4MIHS
pronounce = PRUNAW4NS
proper = PRAA4PER
proportional = PRAXPOH4RSHUNUL
protect = PRAXTEH4KT
proud = PRAW4D
psychiatrist = SAYKAY4AXTRIX6ST
public = PAH4BLIXK
publish = PAH4BLIHSH
pull = PUH4L
pulse = PAH4LS
pure = PYUW4R
push = PUH4SH
put = PUH4T

- Q -

quality = KWAA4LIXTIY
quantity = KWAA4NTIXTIY
question = KWEH4SCHUN
quick= KWIH4K

quiet = KWAY4IXT
quit = KWIH4T
quiz = KWIH4Z
quote = KWOW4T
quotient = KWOW4SHUNT

- R -

race = REY4S
radar = REY4DAAR
radiation = REY5DIYEY4SHUN
radio = REY4DIYOW
radius = REY4DIYAHS
rain = REY4N
random = RAE4NDUM
range = REY4NJ
rare = REH4R
rate = REY4T
rather = RAE4DHER
ratio = REY4SHIYOW
reach = RIY4CH
reaction = RIYAE4KSHUN
read = RIY4D
realistic = RIY5LIH4STIXK
reason = RIY4ZUN
receive = RIXSIY4V
reciter = RIXSAY4TER
recognize = REH4KAXGNAYZ
recommend = REH5KUMEH4ND
record = REH4KERD
recover = RIYKAH4VER
rectangle = REH4KTAENXGUL
reduce = RIXDUW4S
refer = RIYFER4
reference = REH4FERIXNS
reflection = RIXFLEH4KSHUN
refrigerator = RIXFRIH4JEREYTER
region = RIY4JUN
register = REH4JIXSTER
regular = REH4GYUXLER
reject = RIXJEH4KT
relativity = REH5LAXTIH4VIXTIY
relax = RIXLAE4KS
relay= RIY4LEY
release = RIXLIY4S
relief = RIYLIY4F
religion = RIXLUH4JUN
remain = RIYMEY4N
remember = RIXMEH4MBER
remove = RIYMUX4V
rent = REH4NT
repeat = RIXPIY4T
replace = RIXPLEY4S
reply = RIXPLAY4
report = RIXPOH4RT
represent = REHPRIXZEH4NT
reproduction = RIY5PRAXDAH4KSHUN
republic = RIXPAH4BLIXK
rescue = REH4SKYUW
research = RIY4SERCH
reserve = RIXZER4V
resistance = RIXZIH4STUNS
respect = RIXSPEH4KT
response = RIXSPAA4NS
rest = REH4ST
restore = RIXSTOH4R
retail = RIY4TEY6L

return = RIXTER4N
reverse = RIXVER4S
review = RIXVYUW4
revolution = REH5VULUXWSHUN
rhapsody = RAE4PSAXDIY
rhythm = RIH4DHUM
rich = RIH4CH
ride = RAY4D
ridiculous = RIXDIH4KYULAXS
right = RAY4T
rigid = RIH4JIXD
ring = RIH4NX
rise = RAY4Z
river = RIH4VER
road = ROW4D
rocket = RAA4KIXT
roll = ROH4L
room = RUW4M
rough = RAH4F
round = RAW4ND
rubber= RAH4BER
rule = RUW4L
run = RAH4N
rush = RAH4SH

- S -

sabotage = SAE5BAXTAA6ZH
sacrifice = SAE4KRIXFAYS
sad = SAE4D
safe = SEY4F
safety = SEY4FTIY
saint = SEY4NT
sale = SEY4L
SAM = SAE4M
same = SEY4M
sample = SAE4MPUL
sanctuary = SAE4NXKCHUWEH6RIY
sandwich = SAE4NWIXCH
sarcasm = SAA4IRKAEZUM
satisfaction = SAE4TIXSFAE4KSHUN
savage = SAE4VIXJ
save = SEY4V
say = SEY4
scale = SKEY4L
scandal = SKAE4NDUL
scarce = SKEY4RS
scatter = SKAE4TER
scenic = SIY4NIXK
schedule = SKEH4JYUWL
scheme = SKIY4M
scholar = SKAA4LER
school = SKUW4L
science = SAY4IHNS
scientific = SAY4UNTIH5FIXK
scientific = SAY4AXNTIH5FIXK
scissors = SIH4ZERZ
score = SKOH4R
scramble = SKRAE4MBUL
scratch = SKRAE4CH
scream = SKRIY4M
screw = SKRUW4
script = SKRIH4PT
scroll = SKROW4L
seal = SIY4L
search = SER4CH
season = SIY4ZUN

second = SEH4KUND
secret = SIY4KRIXT
secretary = SEH4KRIXTEH5RIY
section = SEH4KSHUN
security = SIXKYUH4RIXTIY
see = SIY4
seek = SIY4K
segment = SEH4GMIXNT
self = SEH4LF
sell = SEH4L
semi- = SEH4MIY
send = SEH4ND
sensation = SEHNSEY4SHUN
senior = SIY4NYER
sense = SEH4NS
sensible = SEH4NSIXBUL
sensitive = SEH4NSIXTIX6V
sentence = SEH4NTIXNS
separate = SEH4PERIXT
sequence = SIY4KWEHNS
serial = SIH4RIYUL
serious = SIH4RIYAHS
serve = SER4V
service = SER4VIXS
session = SEH4SHUN
set = SEH4T
settle = SEH4TUL
several = SEH4VERUL
sex = SEH4KS
shadow = SHAE4DOW
shake = SHEY4K
shame = SHEY4M
shape = SHEY4P
share = SHEY4R
sharp = SHAA4RP
she = SHIY4
sheet = SHIY4T
shield = SHIY4LD
shift = SHIH4FT
shock = SHAA4K
shoot = SHUW4T
shop = SHAA4P
short = SHOH4RT
should = SHUH4D
show = SHOW4
shy = SHAY4
sick = SIH4K
side = SAY4D
sight = SAY4T
sign = SAY4N
signal = SIH4GNUL
silent = SAY4LIXNT
silver = SIH4LVER
similar = SIH4MULER
simple = SIH4MPUL
simplicity = SIHMPLIH4SIXTIY
simulator = SIH4MYULEYTER
sin = SIH4N
single = SIH4NXGUL
sinister = SIH4NIXSTER
sir = SER4
siren = SAY4RIXN
sit = SIH4T
situation = SIH5CHUWEY4SHUN
skeptical = SKEH4PTIXKUL
sketch = SKEH4TCH

skill = SKIH4L
skip = SKIH4P
slang = SLAE4NX
sleep = SLIY4P
sleeve = SLIY4V
slip = SLIH4P
slot = SLAA4T
slow = SLOW4
small = SMAO4L
smart = SMAA4RT
smell = SMEH4L
smooth = SMUW4DH
snap = SNAE4P
so = SOW4
social = SOW4SHUL
society = SAXSAY4IXTIY
soft = SAO4FT
solar = SOW4LER
soldier = SOH4LJER
solemn = SAA4LUM
solid = SAA4LIXD
solitude = SAA4LIXTUW6D
solution = SULUW4SHUN
some = SAH4M
somebody = SAH4MBAADIY
song = SAO4NX
soon = SUW4N
sophisticated = SAXFIH4STIXKEYTIXD
sorry = SAA4RIY
sort = SOH4RT
sound = SAW4ND
south = SAW4TH
space = SPEY4S
spare = SPEY4R
spatial = SPEY4SHUL
speak = SPIY4K
special = SPEH4SHUL
specific = SPAXSIH4FIXK
speculate = SPEH4KYULEYT
speech = SPIY4CH
speed = SPIY4D
spell = SPEH4L
spend = SPEH4ND
sphere = SFIY4R
spin = SPIH4N
spiral = SPAY4RUL
spirit = SPIH4RIXT
splendid = SPLEH4NDIXD
split = SPLIH4T
spoil = SPOY4L
spontaneous = SPAANTEY4NIYAHS
sports = SPOH4RTS
spot = SPAA4T
spread = SPREH4D
spring = SPRIH4NX
spy = SPAY4
square = SKWEH4R
squeeze = SKWIY4Z
stability = STAXBIH4LIXTIY
staff = STAE4F
stand = STAE4ND
standard = STAE4NDERD
star = STAA4R
start = STAA4RT
state = STEY4T
static = STAE4TIXK

station = STEY4SHUN
stay = STEY4
steady = STEH4DIY
steer = STIY4R
step = STEH4P
stereo = STEH4RIYOW
stick = STIH4K
stimualte = STIH4MYULEYT
stock = STAA4K
stone = STOW4N
stop = STAA4P
store = STOH4R
story = STOH4RIY
straight = STREY4T
strange = STREY4NJ
strategy = STRAE4TIXJIY
street = STRIY4T
strength = STREY4NTH
strike = STRAY4K
strong = STRAO4NX
structure = STRAH4KCHER
stubborn = STAH4BERN
student = STUW4DIXNT
study = STAH4DIY
stuff = STAH4F
stupid = STUX4PIXD
style = STAY4L
subject = SAH4BJEHKT
substance = SAH4BSTIXNS
subtle = SAH4TUL
succession = SAHKSEH4SHUN
succeed = SAHKSIY4D
such = SAH4CH
sudden = SAH4DIXN
suggest = SAHGJEH4ST
sum = SAH4M
summer = SAH4MER
sun = SAH4N
super = SUX4PER
superb = SUXPER4B
superior = SUXPIH4RIYER
supply = SAXPLAY4
support = SAXPOH4RT
sure = SHUX4R
surprise = SERPRAY4Z
surroundings = SERAW4NDIHNXGZ
suspend = SAHSPEH4ND
swear = SWEH4R
sweep = SWIY4P
swell = SWEH4L
swing = SWIH4NX
syllable = SIH4LAXBUL
symbol = SIH4MBUL
symbolic = SIHMBAA4LIXK
symmetric = SIHMEH4TRIXK
sympathy = SIH4MPAXTHIY
synchronize = SIH4NXKRAX5NAYZ
synonym = SIH4NUNIXM
system = SIH4STUM
synthesizer = SIH4NTHAXSAYZER

- T -

tab = TAE4B
table = TEY4BUL
tactical = TAE4KTIXKUL
tail = TEY4L

take = TEY4K
talent = TAE4LIX6NT
tall = TAO4L
talk = TAO4K
tap = TAE4P
tape = TEY4P
target = TAA4RGIXT
task = TEY4ST
tax = TAE4KS
teach = TIY4CH
team = TIY4M
technical = TEH4KNIXKUL
technology = TEHKNAA4LAXJIY
telephone = TEH4LAX6FOWN
television = TEH4LAX6VIXZHUN
temper = TEH4MPER
tender = TEH4NDER
tense = TEH4NS
tension = TEH4NSHUN
term = TER4M
terminal = TER4MIXNUL
terrestrial = TER6EH4STRIY6UL
terrible = TEH4RAXBUL
territory = TEH4RAXTOH6RIY
terror = TEH4RER6
test = TEH4ST
testimony = TEH4STUMOHNIY
text = TEH4KST
than = DHAE4N
than = DHAE4N
thank = THAE4NXK
that = DHAE4T
the = DHAH4
theater = THIY4AHTER
then = DHEH4N
theorem = THIY4RUM
theory = THIY4RIY
thermometer = THERMAA4MIXTER
thesis = THIY4SIXS
they = DHEY4
thin = THIH4N
thing = THIH4NX
think = THIH4NXK
this = DHIH4S
thought = THAO4T
threshold = THREH4SH/HOWLD
through = THRUW4
ticket = TIH4KIXT
tight = TAY4T
time = TAY4M
tiny = TAY4NIY
tired = TAY4ERD
title = TAY4TUL
together = TUXGEH4DHER
tolerance = TAA4LERIXNS
tone = TOW4N
tool = TUW4L
top = TAA4P
toss = TAO4S
touch = TAH4CH
tough = TAH4F
tournament = TER4NUMIXNT
toward = TOH4RD
toward = TOW4RD
town = TAW4N
toy = TOY4

trace = TREY4S
track = TRAE4K
trade = TREY4D
tradition = TRAXDIH4SHUN
traffic = TRAE4FIXK
trail = TREY4L
trajectory = TRAXJEH4KTERY
transaction = TRAENZAE4KSHUN
transfer = TRAE4NSFER
transform = TRAENSFOH4RM
transistor = TRAENZIH4STER
translate = TRAE4NZLEYT
transmit = TRAE4NZMIXT
transparent = TRAE5NSPEH4RIXNT
transportation = TRAE5NZPOHRTEY4SHUN
trap = TRAE4P
treasury = TREH4ZHERIY
tree = TRIY4
trek = TREH4K
tremendous = TRIXMEH4NDAXS
trespass = TREH4SPAES
trial = TRAY4UL
trangle = TRAY4AENXGUL
trick = TRIH4K
trgger = TRIH4GER
trim = TRIH4M
trip = TRIH4P
triple = TRIH4PUL
triumph = TRAY4AHMF
troll = TROW4L
trophy = TROW4FIY
trouble = TRAH4BUL
truck = TRAH4K
true = TRUW4
truth = TRUW4TH
trj = TRAY4
tune = TUW4N
tunnel = TAH4NUL
turn = TER4N
tutor = TUW4TER
twist = TWIH4ST
type = TAY4P
typewriter = TAY4PRAYTER

- U -

ugly = AH4GLIY
ultimate = AH4LTAX6MIXT
uncle = AH4NKUL
under = AH4NDER
understand = AH5NDERSTAE4ND
uniform = YUW4NIXFOHRM
union = YUW4NYUN
unit = YUW4NIXT
universal = YUW5NIXVER4SUL
unless = AHNLEH4S
up = AH4P
upset = AHPSEH4T
urge = EH4RJ
use = YUW4S
utility = YUWTIH4LIXTIY

- V -

vacation = VEYKEY4SHUN
vacuum = VAE4KYUWM
vague = VEY4G
valid = VAE4LIXD

value = VAE4LYUW
valve = VAE4LV
vanadium = VUNEY4DIYUM
vapor = VEY4PER
variation = VEH5RIYEY4SHUN
various = VEH4RIYAHS
vary = VEH4RIY
veal = VIY4L
vector = VEH4KTER
vegetable = VEH4JTAXBUL
vehicle = VIY4IX6KUL
ventilate = VEH4NTULEYT
verb = VER4B
versatile = VER4SAXTUL
verse = VER4S
version = VER4ZHUN
vertical = VER4TIXKUL
very = VEH4RIY
veto = VIY4TOW
vibration = VAYBREY4SHUN
vicinity = VAXSIH4NIXTIY
victory = VIH4KTERIY
video = VIH4DIYOW
village = VIH4LIXJ
vinyl = VAY4NUL
violation = VAY4AXLEY5SHUN
virtue = VER4CHUW
visible = VIH4ZIXBUL
visit = VIH4ZIXT
vital = VAY4TUL
vocabulary = VOHKAE4BYULEHRIY
vocal = VOW4KUL
voice = VOY4S
volt = VOW4LT
volume = VAA4LYUWM
voluntary = VAA4LUNTEH5RIY
vote = VOW4T
vowel = VAW4UL
voyage = VOY4IXJ
video = VIH4DIYOW

- W -

wafer = WEY4FER
wage = WEY4J
wait = WEY4T
wake = WEY4K
walk = WAO4K
wall = WAO4L
war = WOH4R
warm = WOH4RM
warp = WOH4RP
warranty = WOH5RIXNTIY4
wash = WAA4SH
waste = WEY4ST
watch = WAA4CH
water = WAO4TER
watt = WAA4T
wave = WEY4V
way = WEY4
weak = WIY4K
wealth = WEH4LTH
wear = WEH4R
wedding = WEH4DIHNX
week = WIY4K
weight = WEY4
welcome = WEH4LKUM

well = WEH4L
were = WER4
what = WHAH4T
wheel = WHIY4L
when = WHEH4N
which = WHIH4CH
while = WHAY4L
whisper = WHIH4SPER
white = WHAY4T
who = /HUW4
whole = /HOW4L
wide = WAY4D
wild = WAY4LD
will = WIH4L
win = WIH4N
window = WIH4NDOW
wing = WIH4NX
winter = WIH4NTER
wise = WAY4Z
wish = WIH4SH
with = WIH4TH
wizard = WIH4ZERD
woman = WUH4MUN
women = WIH4MIXN
wonder = WAH4NDER
word = WER4D
Wordrace = WER2DREYS
work = WER4K
world = WUH4RLD
worry = WER4IY
would = WUH4D
wrap = RAE4P
write = RAY4T
wrong = RAO4NX

- X -

Zerox = ZIH4RAAKS
X-ray = EH4KSREY
xylophone = ZAY4LAXFOWN

- Y -

yacht = YAA4T
yard = YAA4RD
yawn = YAO4N
year = YIH4R
yellow = YEH4LOW
yes = YEH4S
you = YUW4
your = YOH4R
youth = YUX4TH

- Z -

zany = ZEY4NIY
zero = ZIY4ROW
zig-zag = ZIH3GZAEG
zip = ZIH4P
zodiac = ZOW4DIY6AEK
zone = ZOW4N