Votrax SC-01 To SpeakJet Translator

The SC-01 speech chip was one of the most popular speech chips in use during the '80s ...

by Robert Doerr

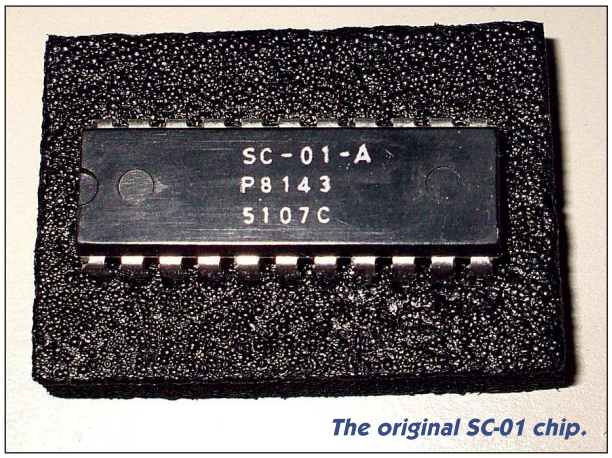
It is the voice for the Heathkit HERO 1 and HERO Jr robots, the RB5X robot, some arcade games, and several other devices. Many will always remember hearing this chip asking if "Dr. Falken" would like to play a game of chess in the classic "War Games" movie. The SC-01 had a long run, but these days they are getting hard to find. I was concerned about this and wanted to ensure there was some sort of replacement option that would be available for the future.

Finding a suitable replacement for this chip proved to be an interesting project. It highlights the many different problems that come up in the robotics hobby. What seemed at first to be a very straightforward endeavor ended up covering a lot of ground. I'll try to review all the ups and downs and share some knowledge along the way.

Currently, there is no direct drop-in replacement for the SC-01 speech synthesizer, so I decided to go about creating one. At least a hybrid one for now!

The Language Barrier

First, let me start with how the SC-01 generates its speech. Some speech chips (or modules) accept regular ASCII text strings and others act like a sound recorder which play back

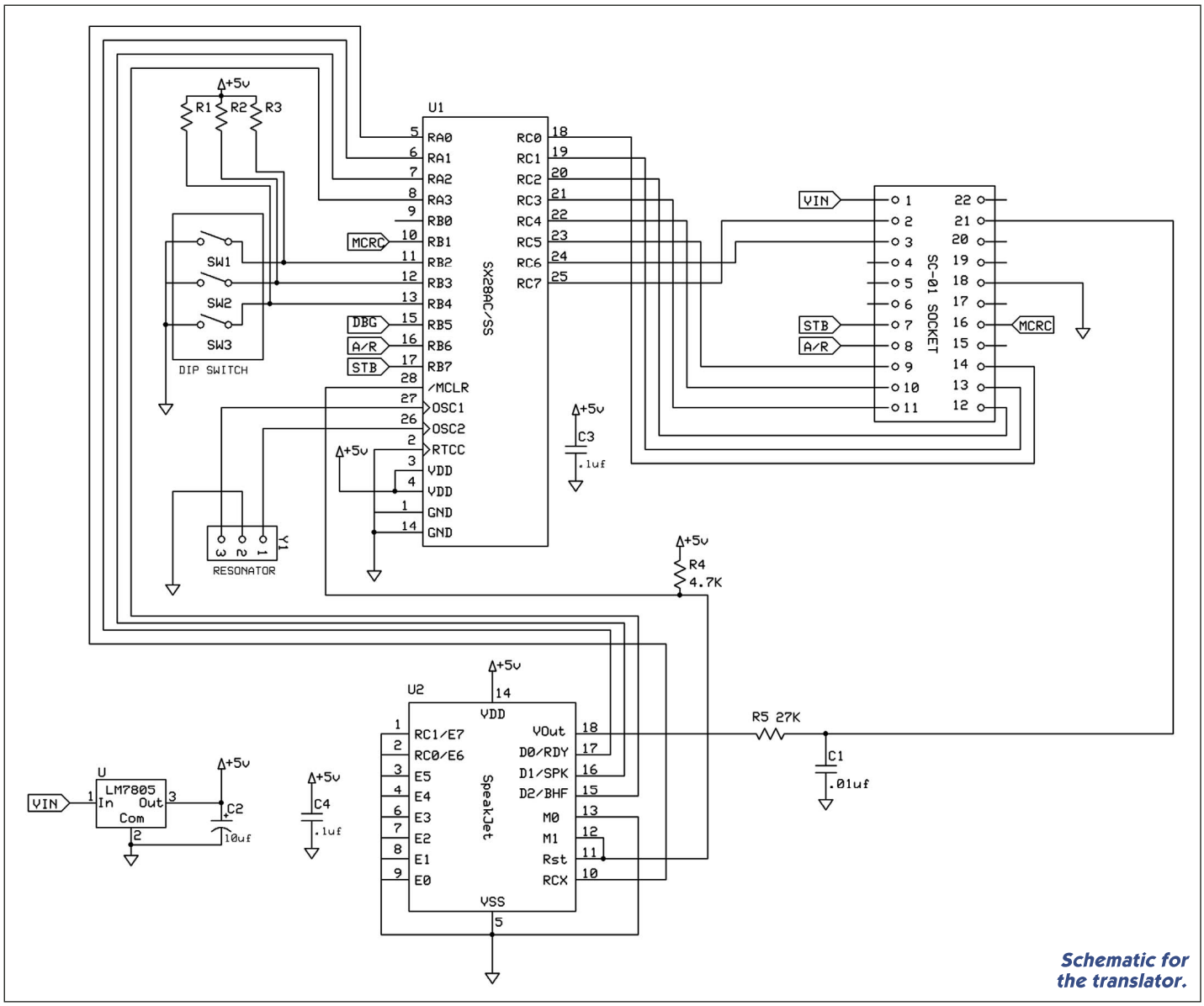


The original SC-01 chip.

stored phrases. The SC-01, however, is a phoneme based synthesizer which builds words from small sound fragments called phonemes. With this in mind, I looked to see what other phoneme-based speech chips are out there. After looking at the few available chips, it seemed that the closest match would be the SpeakJet (SpeakGin) chip. Although it too is phoneme based, that is about all it has in common with the SC-01 chip (except that they are both in DIP packages).

In the SC-01, there are 64 of these phonemes defined. The SpeakJet has 72 (allophones) plus a variety of sound effects. The first part of the project was to see if this idea had merit and was possible. All of the codes for each phoneme are different for each chip. I went through and made a lookup table for what I thought would be a good mapping of each SC-01 phoneme to SpeakJet allophone. With this conversion table in hand, I had some speech strings from the HERO 1 that I ran through the table.

Initially, I had a SpeakJet wired up to an old Handyboard for testing. I then took the translated string of phoneme codes and with an Interactive 'C' program, sent them all to the SpeakJet. The results of that first test were inspiring and showed that this could be a viable option. Some of the words sounded exactly the same while others needed work. (More to follow ...)



Schematic for the translator.

The Protocol Barrier

Now that the SpeakJet could sound like the good old SC-01 (provided the right codes were fed into it), the next step was handling the protocol it uses to talk to the host. These days, a lot of peripheral devices can be told what to do using a single serial line with perhaps a handshaking line or two.

The SC-01 and many earlier devices are from when most peripheral devices were parallel based. The SC-01 accepts six parallel bits of phoneme data, two bits of inflection data, and has a couple control lines to latch the data and acknowledge (busy) that it was received. This added one more thing to deal with for a translator. The SpeakJet, on the other hand, expects to receive all the allophones sent as a serial data stream.

Too Much Power

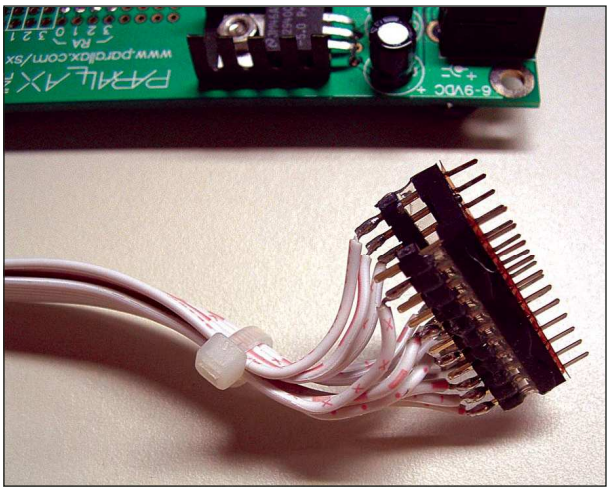
Another oddity about the SC-01 is its source of power. Instead of just +5V that most devices seem happy with, this chip was commonly run at +12V. Even so, it had a nice feature in that the data lines had 5V compatible inputs to make it easy to interface to standard 5V systems. A hybrid module would also need an onboard 5V regulator to bring the supply down to a safe level.

Enter the Translator

To fit in with the idea of drop-in replacement for the SC-01, the whole thing had to plug into the odd 22-pin DIP socket and act just like an SC-01. Lately, I've been working with the Parallax SX series of microcontrollers and found that the SX28 was ideal for this project. The translator program was written in SX/B (BASIC compiler) to make it easy for everyone reading this article to follow the code. The SX28 acts as the hardware protocol translator, phoneme translator, and handles all the handshaking signals. In order to do this, it must:

- Accept the parallel phoneme and pitch data meant for the SC-01.
- Acknowledge to the host that it was received.
- Perform a lookup to determine what the equivalent SpeakJet phoneme should be.
- Send any special codes to the SpeakJet.
- Send the new phoneme to the SpeakJet (if buffer is not too full).
- Set the acknowledge line high to signal that host can send another phoneme.

The custom DIP adapter.



The Hardware

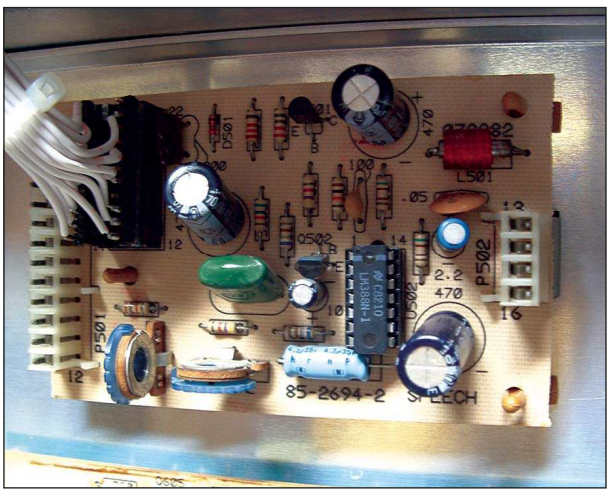
To jumpstart the project, this whole prototype was built upon an SX28 protoboard that Parallax offers. It contains a SX28AC/SS-G surface mount chip, voltage regulator, prototype area, and a header for the programming adapter. Programming the SX series chips also requires the use of an SX-Key or SX-Blitz. The ability to quickly Flash the SX28 processor with new versions of the translator code really helped speed the development process along.

The SX28 is available in both a 28-pin SSOP package and a 28-pin DIP package. An important point to note is that pins 1 through 14 are not the same on both package styles. Make sure to note which package is used in any schematic that uses the SX28 chip! Care has to be taken when switching package styles to ensure the wiring is correct. In the example schematic, a surface mount SX28 SSOP package was used.

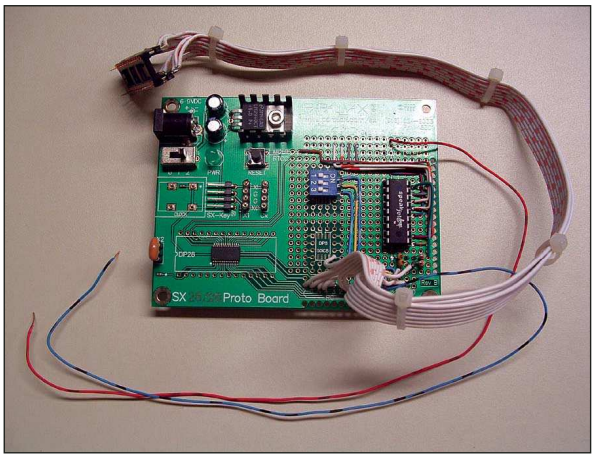
The SX28 chip sits between the 22 pin SC-01 socket and the SpeakJet chip to translate all the signals. The only exception is the voice out signal which the SpeakJet handles and goes out through pin 21 of the 22 pin socket. Port A of the SX28 handles the serial data to and also gets the status back from the SpeakJet. Port B is used to get the strobe from the host since that port can generate interrupts. This will allow for an alternate version of the translator to be written as interrupt driven.

A portion of port B can also act as an analog converter and that pin is wired to pin 16 (MCRC) of the 22 pin SC-01 socket. It can eventually look at the riginal SC-01 timing signal and adjust the translation speed accordingly. The remaining pins on port B are used to get configuration information from a DIP switch. Port C is used to get the phoneme and inflection data from the host.

To ensure the serial timing to the SpeakJet would be accurate, a 4 MHz resonator is used. Although the internal RC clock of the SX28 is fine for many projects, an external resonator or crystal should be used when timing is critical.



Original SC-01 amp board with the DIP adapter replacing the SC-01 microchip.



SpeakJet-based replacement board.

Issues That Came Up *(and were overcome)*

As a real world test, I pulled out the genuine SC-01 chip from the Speech board of my HERO 1 robot and plugged in my translator gadget.

The original power source for this prototype originated from the supply pin of the SC-01 socket. The HERO 1 can shut down parts of itself to save power and as a result the power on the speech board would cycle on and off this 12V supply whenever the robot tried to talk. It also made downloading new translator program code a chore since the board would normally be off.

As a temporary solution, I supplied power to the prototype board from the +5V connection on the HERO 1 breadboard. The amplifier section on the HERO 1 speech board would still power up and down to save power. Later, the 5V regulator will take the 12V from pin 1 on the 22 pin socket for power so that all the connections are directly to the SC-01 socket.

The default behavior of the SpeakJet is to announce 'READY.' It is also what the HERO 1 says when you first power him up. When the power was first applied, it would say READY but it was misleading. I knew it wasn't going to be that easy! It only did it the first time it was powered up and following that, all it made was a sort of sick 'Ehhh' sound repeating a bit before going silent.

Well, that isn't supposed to happen! It was pretty obvious what was going on with the READY announcement, so I took another look at the source code. I found a typo for the variable name used for the index to look up the SpeakJet allophone in the table. As a result, it was always pointing to the first phoneme in the table (that happens when your index is always 0) so that explained it. I fixed that and started to hear a few new phonemes.

I could make out some of the phonemes and portions of words but it was way off. I knew that the lookup table would need work but thought "I can't be *that* far off!" A little troubleshooting work quickly uncovered what had happened. I had used a 22 pin DIP socket to make the plug-in adapter to go into the original SC-01 socket. The leads are fairly thin as it was a standard dual leaf socket. You may have already guessed what had happened.

Two of the six data leads used to send the phoneme data to the SC-01 on my custom connector had folded under instead of going into their appropriate pins in the socket below. That left two of the six data bits used to select a phoneme open and in a floating state. Usually when something is open it floats high, so it meant that some phonemes would never be used and other incorrect ones would be selected in their place! Once that was fixed, it started to sound a lot better.

The robot would speak a portion of what it was supposed to but would be truncated before it could finish. The SC-01 would accept a single phoneme at a time and would be ready to accept the next while speaking so the speech would be continuous. This ended up being another issue.

The SpeakJet is nice enough to offer a 64 byte buffer for incoming commands/allophones. The robot would send along all its phonemes which were being buffered by the SpeakJet. Once transferred, the robot would assume the speech was done and shut down power to the speech board, shutting off the sound amplifier. (Hmm, that's a problem!)

To confirm that was the case, I threw in a small delay after each phoneme was received and sent over to the SpeakJet. It definitely showed this was it and then brought up the issue of how to deal with it. The SpeakJet provides a few handshaking signals. It can signal if it's ready, it can tell you if it is actively speaking, and it can tell you if the 64 byte buffer is half full. Unfortunately, it has no easy way of letting you know when there is only one byte left in the buffer.

This is something that would have been extremely useful in its role of impersonating an SC-01. Instead of sending all it could take and using the buffer half full as a handshaking signal, I wanted to spoon-feed the chip and provide it the allophone codes one at a time so I would know about when it would be done. I did try using the speaking line as the handshake, but the problem was there ended up being a pause between each phoneme which was unacceptable. It seemed that it

References

www.robotworkshop.com
Author's website, home for the HERO robots and vintage robot Guru (Pre-programmed SX28 chips with resonator available here).

www.parallax.com
Provider of the SX series processors. Offers free software development tools like SX/B.

http://forums.parallax.com/forums/
Online user forum for SX series microcontrollers.

www.speechchips.com
Provider of SpeakJet chips.

www.redcedar.com
Great historical reference and data on SC-01.

groups.yahoo.com/speakjet
Online user group for SpeakJet chip.

www.speakjet.com
Website for the SpeakJet chip.

www.soundgin.com
Website for the SpeakGin and SoundGin chips.

www.rbrobotics.com
Home of the RB5X robot.

would either take too much or too little. Unfortunately, none were just right ...

Luckily, an elegant little solution hit me. Why not just go ahead and send codes to the SpeakJet using the buffer half full as a throttle. Then as a way to sync up the timing, I could use the Speaking handshake line whenever the SC-01 was sent a pause or a STOP. It just so happens that the convention used in the HERO 1 is such that all the speech sent to the speech board ends with a STOP to ensure the board would finish speaking before it was powered off. This was PERFECT! An audible pause between phonemes might normally be a problem, but if the phoneme was a silent one, then no one would notice. This is just what I needed to make it work on HERO 1 and it got around the power issue.

After that, some more work went into the translation table for the SC-01 phonemes to SpeakJet allophones. Extra code was added to consider the two inflection bits. If they changed state from the last phoneme, then the program will send out a code to the SpeakJet to change its inflection to improve the emulation. It's still not perfect, but keeps getting better with each revision.

One of the last minute additions into the code was to send a small pause phoneme to the SpeakJet when everything was first powered up. Without this, the first phoneme that was translated and sent to the SpeakJet was garbled. Adding that delay cleared up the problem and now everything sounds just as expected.

The Extras

Finally — just for fun — I wanted to use some of the extra features of the SpeakJet and put a few of the extra unused pins on the SX28 chip to good use. One of the unused bits on port B (RB.5) can send debugging info to a serial port for monitoring the translation process. A small DIP switch was added to configure the way the translation is handled. Alternately, instead of a DIP switch, an output port on the robot or another device could be used to control these settings.

In the example program provided, DIP switch 1 is used to enable R2/Bio sounds instead of regular SC-01 phoneme translation, DIP switch 2 enables extra status info to be sent out the debug port, and DIP switch 3 enables a small section of code to initialize a fresh SpeakJet chip by disabling its startup READY announcement.

Now, by merely flipping a DIP switch, I can have HERO speak just like R2D2 since it translates real phonemes to equivalent R2 sounds. I don't know if the real R2 would understand it but everyone that hears it seems to like it! So, not only will this project effectively emulate an SC-01, but also adds value by leveraging some extras within the SpeakJet.

Ideas for Improvement

- Better matching of audio output circuitry here.
- Monitor RC circuit that sets SC-01 timing and adjust overall timing of emulation.
- Tweak phoneme lookup table.
- Add another mode to enable more special SpeakJet features like sound effects.
- Make another version to translate from the SC-02 (SSI263) to the SpeakJet.

It should be noted that either the SpeakJet or SpeakGin chips can be used interchangeably as the target speech chip with this project. For those that may not be aware, these two devices are actually the exact same chip. The co-developers decided to pursue different markets and each have their own brand name for this particular speech chip.

Eventually, this can all be put on a little hybrid module as a nice tidy plug-in replacement package. For those of you interested in trying out the translator yourself, preprogrammed SX28 chips with a resonator will be available from the author. SV

Note

The source code for the translator is available on the *SERVO* website at **www.servomagazine.com**.

About the Author

Robert has been working on personal robots since building one of the early HERO 1 robot kits when they came out. He enjoys repairing/rebuilding/upgrading all the robots from that era. It can be challenging at times, but it is rewarding to keep these old robots going.