SPHERE
NEWSLETTER

DECEMBER 1979

VOLUME IV ISSUE 3

EDITORS: ROGER J. SPOTT
JEFF BROWNSTEIN

SOFTWARE

HARDWARE

EDITOR'S MAILBAG

PLEASE SEND TYPED MATERIAL FOR NEXT ISSUE TO:

DR. JEFF BROWNSTEIN
2 TOR ROAD
WAPPINGERS, NY. 12590

MORE SOFTWARE COMMENTS

J. RAEHL
ESCONDIDO CA.

1. MEMORY TEST AND OTHER PROGRAMS

The attached memory test program supplements the random-pattern tests commonly available. The idea was taken from the article "The M-1 Worm" in Personal Computing magazine for July 1979. The test covers the execution of a variety of instructions as well as stack operations, at a given location. It proves, for example, that the screen memory cannot handle stack operations. At one point in the process of solidifying my memory boards, my random-pattern test ran perfectly. The WORM, however, bombed every couple hundred locations (stack operation problems, I think). The test consists of a small program whose sole function is to incrementally relocate itself in memory, one byte per pass. At the end of each pass, the current location of the program is displayed on the screen.

There are two methods of bombout entrapment. First, if a relocated byte does not compare equal with the original, the program will hang with the last displayed location on the screen. A reset is needed to clear this hang. Secondly, the program covers its trail with SWI instructions. If memory has been preset to SWI instructions before running the program, a branch to oblivion should encounter an SWI, and go to the SWI interrupt address in PDS.

The WORM program itself begins at location WORM4 in the listing. A handy routine for storing the program at a common location (such as in PROM), moving the program to the area to test, and presetting memory to SWI instructions, is also included. An interface to the Raehl extended monitor is included for those who have this monitor.

There appears to be interest in connecting the Sphere to an acoustic coupler, and talking to other computers. I have included the dumb terminal monitor portion of my extended monitor, to assist those who may need such a program.

The GETCHR routine included provides for software key repeat. Just hold the key for a quarter second or more. Control-A will provide a software keyboard shift lock between upper case and lower case letters, if you have an upper/lower case keyboard and have added a lower-case character generator to the CRT board. I also have a variation of the above GETCHR which will run two keyboards at the same time (fetch from either keyboard). It can be obtained from me by sending me a self-addressed envelope with stamps for two ounces.

I also have a program which uses arithmetic overflows to create an interesting kaleidescope of patterns--worms going across the screen, arcs, ellipses, stars, and so forth. Liberal instructions are provided for modifying it. If it is not published in this newsletter (I submitted it) due to lack of space, it can be obtained from me by sending an SASE with stamps for 2 ounces.

I revised the CRT monitor routine in the February 1979 newsletter to use low memory locations for split-screen operation per John Rible's suggestion, and made it location-independent (runs anywhere in memory). I also have an initialization routine to obtain screen parameters from PDS. This, too, is available for an SASE with stamps for 2 ounces.

2. SOFTWARE AVAILABLE

To make a few coins for hardware, I am providing my two biggest software packages for a small fee. The Raehl extended monitor is still available at $10 for the source listing and $7 for source and object on cassette. It occupies about 2K bytes and provides 30 handy functions for software scrounging, including cassette verify, string search, four memory dump and two memory change

flavors. A GETCHR with software key repeat is included. To see this monitor in action is to love it (so I've heard). No bugs have been reported in two years of use. See February 1978 newsletter for details, and review in April 1978 newsletter.

My latest software project has been a location-independent (will run anywhere in memory) timeshare terminal monitor, for connecting a Sphere to any other computer via a modem or acoustic coupler (see elsewhere for related hardware hints). This monitor uses the CRT monitor program included in the February 1979 newsletter, modified as described above in Memory Test and Other Programs. This section of the timeshare monitor is independent, and explicit instructions are included for use in other software. Several control-code (hex 00 to 1F) functions are provided for saving, displaying, and transmitting data in a memory buffer. An interface to all the functions of the Raehl extended monitor is also included (nice, as the cassette load and store are in the monitor).

I also plan to include a split-screen editor for data in the memory buffer, but the above is now available in a well-documented 40-page listing (including symbol cross-reference) for $5. Since I used a cross-assembler, source is not available on cassette. My address is Jim Raehl, 943 Begonia, Escondido, CA 92027, or call me at home (714) 741-1434.

3. CROSS-SOURCE CODE GENERATOR

I wrote a cross-source code generator in COBOL some time back. Now that I have access to a machine with COBOL, I can decode software for which source is not available. I currently have listings of SWTPC BASIC V2.0 and SWTPC CO-RES editor/assembler. If anyone wants copies, I think $5 each plus proof of purchase of the product should cover the mailing expenses. The listings include ready-to-assemble code, generated symbols for all data and procedure references, and crossreference listings of the symbol definitions and references.

If anyone wants something run through the source code generator, send the object on cassette, plus a little money to cover mailing back the listing (I mail third class, but even that costs a couple of dollars for 50-page listings).

It isn't worthwhile for me to do it, but if anyone else wants to do a COBOL to 6800 translation of the source code generator, be my guest. Contact me for a source listing of the COBOL code.

4. PLEA FOR SOFTWARE STANDARDS

There was a plea for more software for the Sphere, in an earlier newsletter. My view, as a professional software writer, is that there will probably be very little software written explicitly for the Sphere. At 100 to 200 hours per 1000 bytes of debugged and documented code, it doesn't pay for a pro to write for a few hundred systems, for profit. Personal time crunches take care of many non-profit operators.

However, considering the tendency to modify Sphere systems, and to make it easier to adapt code for these modified systems as well as other 6800 systems (SWTPC in particular), may I suggest some coding standards.

- Use the CRT monitor program mentioned above, for all (repeat, all) screen displays. The monitor will properly handle any memory-mapped video board. Only an initialize routine need be written to adapt to a given board. Do not directly poke at the screen, unless you are writing for a specific video board.
- Obtain the keyboard location from PDS location FC64 (FE instruction instead of

CE instruction). Keyboard routines will thus work with either the CPU1 or CPU2 board. Do all keyboard fetches from a common location.

- Obtain screen parameters from PDS also. Screen start is at FC38, screen end at FC40, line length at FD63, last line at FD47. Do not so radically alter PDS that these locations no longer contain these values.
- Write modular code. Think in terms of doing a "function", and then a description of the function, followed by a block of code to do that function. Use subroutines when code is common to more than one function. Isolate system-dependent code to a small number of areas, well-documented. Minimize system-dependent code. Most programs could be made to run on any 6800 system with little modification, if care is taken in the design and coding.
- Use higher-level languages where possible. Avoid adding "features" to converted higher-level software; they only make possible incompatible programs.
- Share that software. I only have two programs (mentioned above) that I charge more than expenses for (for pocket money for hardware). The rest is available for mailing expenses or swap. I have the following BASIC (SWTPC V2.0) programs: STARWAR (full bells & whistles, requires 32K), LOAN, CHASE, LIFE INSURANCE, BLUFF, QUBIC (4x4x4 TIC-TAC-TOE), LUNAR LANDER, CRAPS, WUMPUS, HAMMURABI, BLACKJACK, NIMBLE, WORLD POWER, HVOLT. Several of these still need some debug work, but at least the typing is done.

MORE HARDWARE COMMENTS

1. SS50 VIDEO BOARD

For a long time, I wanted a video board with more screen format control than the Sphere CRT1. F&D Associates (1210 Todd Road, New Plymouth, OH 45654) has recently advertised such a board for the SWTPC SS50 bus, in On-Line. I received their literature, and was amazed at the features. Virtually any line length by lines per screen format is available, plus 4K bytes of screen memory and a PROM for 128x48 (TRS80) graphics. They provide discounts for group purchases, so we may want to consider this.

I ordered one (using my VISA card, in case of trouble), and received it in less than two weeks. The board cost $37.50 (bare), and the documentation was phenomenal. They covered every possible problem I could think of, and had listings of modifications to several SWTPC software products. The board is very dense and requires very careful soldering.

I haven't completed debug of the board yet, but my experiences thus far may be of help. I was able to obtain almost all of the parts from a combination of Jameco Electronics, Jade Co., Hobby World, and Advanced Microcomputer Products. The parts list specifies two AMP company number 53137-1 hex switches, which I was unable to obtain (see below). The most expensive parts are the CRT controller IC at $40, the memories at $6.50 each (two minimum, eight maximum), the character generator at $13, and an optional 2708 or 2716 graphics EPROM (theirs is $15, programmed for TRS80 graphics). Total parts cost is an estimated $120 to $150, plus the board.

Selection of the crystal frequency is somewhat a seat-of-the-pants matter. The specified crystal in the instructions is 12.3 mHz, for 64 by 16 format. I chose 15 mHz, for 80 by 24 format. Whether I will get the entire 80 by 24 is yet to be seen. I developed a formula for crystal frequency, which should help. This is,

clock rate = 1.3 * (((CPL+25)*LINES*96)/16667)

where

clock rate is in units of mHz,
CPL is displayed characters per line,
LINES is lines per screen,

1.3 allows for retrace (a variable fudge factor),
25 extra characters per line allows for horizontal margin (variable).
The Motorola MC6845 specification booklet gives data on using 80 by 24 format, on a premium monitor.

I had to invent a solution for the unavailable hex switches. I didn't want to use jumper wires, as I might want to move around the address blocks. Some IC pinout studies indicated I could use a 7408 quad AND-gate IC, in place of the hex switches. To get an address of Fxxx, I just bent vertical pins 1 and 8, wired pin 9 to pin 7, and wired together pins 4, 5, 6, 11, 12, 13, 14. That was for the CRT controller selection switch. To get Dxxx for memory selection (I use D000 so that someday I can use MIKBUG at E000), I again bent up pins 1 and 8, wired together pins 6, 7, 9, and wired together pins 4, 5, 11, 12, 13, 14. The IC's then can be mounted with pins 1-7 of the IC in pins 2-8 of the switch position. I thus have no jumpers, at the price of two 7408's (reversible by reversing the pin bending and wiring).

Wiring the board is straightforward, per their instructions. I suggest using gold pin soldertail standard sockets from Jameco, and mounting them first. Solder only two corner pins, and make sure the other components will fit before soldering down the sockets. The 2uf tantalum capacitor is a particularly tight fit between two memory sockets. There are a number of jumper options. I chose J4 for graphics when D7 is a one, J7 and J11 for black characters on white controlled by the PIA, and J8 for composite sync. Jumpers J1 and J2 or J1 and J3 are required, if you want to limit the CRTC/PIA address selection. If you don't use them, this knocks out an entire 4K memory section for only 4 addresses. With them, only 256 bytes are knocked out. I chose F500 for this address. Inverters are available at pins 1-2, 3-4, and 5-6 of IC27, for inverting the A8 and A10 address lines. Wiring this on the back side is easy. This would put the CRTC/PIA selection at F000-F0FF. An appropriate switch selection can then be used, for the specific starting address of the 4-byte block.

The CRT board uses regulators on-board for the power. I eliminated the LM323K-5 regulator and put a jumper (18-gauge wire) between its input and output pins. I also omitted the two zener diodes for regulating the -5 and +12 supplies. The 51 ohm and 180 ohm resistors were replaced with jumpers, but could probably be left in.

To convert from the SS50 bus to the Sphere bus, I mounted sockets on an 8x10 piece of perf board, as they would normally be on a Sphere board. I then mounted the CRT board on the back of the perf board (solder side of the sockets), with the SS50 holes near the socket pins. I then used ribbon cable parallel to the CRT board, and fed the leads out at the proper places. I found that using an 8-lead ribbon from X3 to the data pins on the CRT board, a 9-lead ribbon from X2 to the A0-A9 address pins, and a 7-lead ribbon from X1 to the rest of the address pins, worked quite nicely. The other pins required a collection of wires.

The pinout of the SS50 bus is as follows: NOT D0 to NOT D7, A15 to A0, GND (3 pins), +5 volts (3 pins), -12 volts (I used -5 volts), +12 volts, Index Pin, Manual Reset, NMI, NOT IRQ, User Defined 2, User Defined 1, NOT Q2, NOT VMA, R/NOT W, NOT RESET, Bus Available, Q1, HALT, 110 Baud, 150 Baud, 300 Baud, 600 Baud, and 1200 Baud.

Since Sphere uses data lines that are inverted from SWTPC, these must all be inverted between the sockets on the perf board, and the CRT board. I mounted sockets between X3 and the power socket at X6, and routed the data lines through these. DM8835 inverter/drivers went into the sockets. In addition, Q2 and VMA must be inverted. I used a 7404 for these. NOT IRQ, R/NOT W, and NOT RESET

need not be inverted.

2. SPHERE CARD RACKS, AND TERMINATION

I was able to obtain a Sphere card rack for a very good price, from Charlie Matteson, although he didn't mention them in his list in the February 1979 newsletter. I put the spare in the CRT area, and moved the power terminal strip under the keyboard, close to the card racks. Now I have space for 14 boards.

I had Jameco Electronics make me some custom cables with enough connectors for a dozen boards. My only complaint on the cables was that they led me to believe the connectors were high quality. They turned out to be cheap Ansley ones with tin pins. Fortunately, they are compatible in pin orientation with my spare Augat ones. As pins break, I can replace the connectors with Augat ones.

While investigating the noise on my address and data busses, I ran into some interesting results. My method may be useful to others. I wanted to determine the contribution of each board to the noise problem on the bus. To do this, I put in a program that would reference the addresses on the board in question as heavily as possible. The results are:

- CPU board--I just did a reset to check this. This board did not generate much noise on the data bus, but lots on the address bus. In addition, it caused a low value in the data lines (3.5 volts for high).
- MEM board--I put in a program with a tight loop at location 1000H. The program in hex is 20 FE. I then jumped to this program. Results indicated not much noise on the data line, and a little on the address line. My MEM boards are heavily modified, so that may affect them.
- PROM board--I jumped to my extended monitor (in PROM), to test this. This board had lots of noise on both the address and data buses.
- SIM board--I put in the program B6 F0 60 20 FB to address the ACIA, and jumped to it. Results indicated this board generated a fair amount of noise.

The extra length of the cables (nearly four feet) caused noise problems, but building a termination board per the December 1978 newsletter really helped. I noticed that the pullup resistor/pulldown resistor combination mentioned in the newsletter, caused my lines to have a high value of only 3 to 3.5 volts. This isn't too hot for memories. As a rule of thumb for pullup resistor values, use the highest resistance you can get away with and still pull up the high voltage properly. There are two considerations--the RC time constant of the resistor and the capacitance of the chips being affected, and the power drawn by the resistors. The RC time constant affects rise times of the signals, and power increases as resistance decreases. A high resistance value also lessens the load on the DM8833 drivers. Resistors of 330 ohms or so will draw a full TTL chip load on the lines.

A pullup/pulldown resistor combination forms a voltage bridge. The bridge will give the signals better edges and tend to cut reflected signals on the lines (ringing). Since all TTL tends to pull down better than up, I first tried a set of 5600 ohm pullup resistors. That didn't affect the ringing much. I then tried 1K pullup and 5600 ohm pulldown, which wasn't much better. Tried the reverse and got a vast improvement. 3300 pulldown and 5600 pullup was better. It seemed that pulldown was best in the 1000 to 3300 ohm range, and pullup in the 3300 to 5600 ohm range. Since a voltage bridge tends to pull in the direction of the lower-valued component, I finally settled on 3300 for both pullup and pulldown, except 1K pulldown on the A12-A15 address lines.

I used a shortcut in playing with the voltage bridge values. Two resistors on each of the 35 used lines is a lot of wiring. Using the scope helped some, but changing just the 8 data lines or 4 A12-A15 address lines and running memory tests, would generally indicate if the new values were worse than the old.

Changing the data lines should affect the error rate everywhere, while the address lines would affect 4K (decimal) blocks of memory. By holding one of pullup/pulldown constant and varying the other, the shortcut would show the boundaries of the varied one.

Two other possible fixes were mentioned to me. One is to use MM5280 memories. My experience with them was not too good. My 4060's seemed to do better. When I ordered the 5280's, I specified that only that part would do. Had to do that after checking with a company that advertised 5280's. They admitted that they were really selling 2107's. The other fix was to connect together the ground planes of all the boards with #16 or heavier wire (near the power socket will do). Connect the wire to the power terminal strip in the computer. This did help some, but is a lot of trouble when fixing boards.

3. RS232 DB25-PIN CONNECTOR FOR SIM

Heavy cables, such as audio cables, don't hang very solidly on the SIM board X24 and X25 sockets. I solved this by wiring a 25-pin connector typically used for RS232 interfaces, to the holes at the top edge of the SIM board. If you hold the board solder side facing you so the holes are at the top of the board, and look at the row of holes above X24 and X25, the holes correspond in configuration to pins on X24 and X25. From the left, the first 7 pairs are for X25, the next two are spares (but the bottom ones are useful ground sources), and the next 7 are are for X24. The rest are spares. You can verify the configuration with an ohmmeter.

I used some quality ribbon cable, and connected the holes to the RS232 connector based on the following configuration. The configuration makes for a neat layout, using the most desired X24 and X25 pins.

When connecting the cable, first solder two lengths of 14-lead (strip back to 12 and 13 leads to correspond to the connector) ribbon cable to the RS232 connector. Use heat-shrink tubing to cover the connections.

Hold the board solder side facing you, with X24 and X25 at the top. Hold the connector with its pins facing you, parallel to X24 and X25 (pin rows horizontal). The 13-pin row should be on top. Pin 1 should be on the left and pin 13 on the right. Now plug the wires into the holes on the solder side of the board, and solder. Be careful when soldering, as the holes are very close together. Use a tiny bit of solder for each hole.

The configuration--

13	(X24-14) CASS2 +12 volts	25 (X24-1)	CASS1 earphone
12	(X24-13) CASS1 aux input	24 (X24-2)	CASS2 on-off
11	(X24-12) CASS2 aux input	23 (X24-3)	CASS2 on-off
10	(X24-11) CASS2 earphone	22 (X24-4)	CASS1 on-off
9	(X24-10) CASS1 +12 volts	21 (X24-5)	CASS1 on-off
8	(X24-9) 20ma -12 volts	20 (X24-6)	Tx 20ma
7	(X24-8) Rx 20ma	19 (X24-7)	CASS1 ground
6	(X25-14) TTL ground	18 (X24-7+)	CASS2 ground
5	(X25-13) Tx TTL	17 (X24-7+)	TTL/RS232 ground
4	(X25-10) Rx TTL	16 (X25-3)	Rx RS232
3	(X25-9) -12 volts	15 (X25-5)	Tx RS232
2	+5 volts	14 (X25-7)	RS232 ground
1	+12 volts		

4. CONNECTING THE SPHERE TO A TIMESHARING SYSTEM

This project was one of the original goals for my Sphere. I had hoped to use the modem section on one of my SIM boards, and built that up. Connected mine to

David Lake's one day at a workshop per Sphere's instructions, put in a simple program to repeatedly send U's, receive them, and display them on a screen, and nothing happened going either way. If anyone has got this circuit working, please write up your experiences in the newsletter, or at least let us know there is at least one successful person. I then went through a surplus Teletype acoustic coupler, a modem (required a DAA), and a modem board that I never was able to verify was hooked up properly. This equipment (except the Teletype coupler) was supposedly in working order when shipped, so if anyone wants it, I'm willing to haggle.

I finally bought a used, but restored to original specifications, commercial RS232 acoustic coupler from U. S. Brokers Company, 7300 N. Crescent Blvd, Pennsauken NJ 08110. It was made by MI2, was both originate and answer, and cost $140 (a fair price). The U. S. Brokers branch I got it from was in Texas. Tried to get schematics, and after a lot of phone calls, found only two sets. MI2 was bought by another company, and the coupler was made so long ago that the company no longer had schematics. The only schematics I could get a copy of were owned by U. S. Brokers, and were engineers' blueprints. It would have cost more than I was willing to pay to copy them, so I gave that up. The only problem I've had with the coupler is that its acoustic isolation is poor. Once in a while I'll get a garbage character or two when our baby squeals, or some similar thing.

Since the coupler was a standard commercial one, I could check it out at work. Connected it to a terminal, set it to answer mode, and called the terminal from a nearby one. The two terminals could, in fact, talk to each other. I then set the coupler in originate mode, and called the in-house computer. Originate mode also worked. That verified that the coupler worked.

The next step was to get a functioning RS232 circuit on my SIM board. To do this, the following jumpers are required:

- 20ma jumper between sockets X24 and X25 must be in (Rx 20ma to -12 volts)
- TTL jumper just above E14 must be in (Rx TTL to ground)
- TTY2 jumper to the right of and below E28 must be in (CTS to ground). In my old schematic this is labelled TTY1 (a bug in the schematic).
- 300 baud must be jumpered in (this rate is pre-strapped).
- The modem chip E34 must be left out of the board.
- The clock generated by the 9601 must be divided by 4 (see elsewhere for hardware hints).

The remainder of the jumpers are correctly pre-strapped via the board etches, and need not be changed. Note that the RS232 jumper is not in.

Since I had the RS232 circuits connected to a male DB-25 pin connector (see DB-25 PIN CONNECTOR writeup), I could make up a female connector with the Rx RS232 and Tx RS232 pins connected together, and plug it in. That enabled me to verify that the RS232 circuit could talk to itself. I started up the dumb terminal monitor program, and verified that pushing a keyboard key would cause the character to appear twice on the screen (once for the keyboard entry, and once for the ACIA receiver). This handy test checks both the RS232 transmitter and receiver circuits. The duplicate character should appear immediately after the original. I've had cases where it appeared several seconds later, due to a bad clock. I also checked the clock rate with a frequency counter. This should be within half a percent of on target, so a crystal-controlled clock may be necessary.

I knew the acoustic coupler used the EIA RS232C standard interface. Studied the usage of the 25 pins. Some were handshaking signals from the acoustic coupler to the terminal, which I could safely ignore. The pins that could matter narrowed down to

pin 2--Transmitted Data, terminal origin (connect to Tx RS232)

pin 3--Received Data, modem origin (connect to Rx RS232)
pin 4--Request to Send, terminal origin (connect to +12 volts)
pin 7--Signal Ground (connect to ground)
pin 20--Data Terminal Ready (connect to +12 volts)

After wiring up a cable between my DB-25 pin connector on the SIM board and the acoustic coupler, I was ready to talk to the in-house system. Dialed it up, and found I was in luck. I got garbage. That was better than the nothing I'd got for several years. Obviously, the ACIA initialization was wrong. The question was what combination of parity, start-stop signals, and character lengths was being used. Our in-house system is a Control Data CYBERNET CYBER 175. Nobody could help, including the CDC communications expert on-site. A co-worker finally found a gold mine in the CDC 752 terminal manual (CDC publication 62957300). It described the baud rates, half/full duplex operation, transmit/receive word sizes and formats, code sets, parity, and the RS232 pinouts. In other words, all the technical information I needed. The needed initialization for the ACIA was a hex 13 (master reset), followed by a hex 09 (divide by 16, 7-bit, even parity).

Made that change to the dumb terminal monitor, and I was on the air. The next step was to expand the dumb terminal monitor to convert and filter the control characters between the Sphere and CYBER. I made a chart of corresponding control codes (like backspace, carriage return, and line feed), and implemented the conversions. The Motorola M6800 Applications Manual shed light on what a "break" condition was, as well as the manual for our TI Silent 700 terminal at work. I'd wondered what a break key did, as I couldn't find a character code for it. Break is a "space" condition (logical 0) in the line for 150-300 milliseconds. I chose 233 milliseconds per the Motorola manual, and put in code to program the ACIA for break when I hit the control-B key (hex 13 followed by hex 69). Since I didn't know whether break was maintained by the ACIA, I put in a 233 millisecond delay loop followed by re-initializing the ACIA to normal. Discovered while experimenting with the TI terminal, that a NUL (hex 00) code also is recognized as break by the CYBER. That explained why I had trouble some years back, reading Teletype tapes with NULL frames.

That gave me a fully functioning dumb terminal. I now started expanding the monitor by adding commands to store characters received from the ACIA into memory, transmit from memory to the ACIA, save lines scrolled off the top of the screen into memory, hex display of incoming characters, half/full duplex, and so forth. Added the revised CRT display routine for split-screen operation, and an interface to my extended monitor. I now have full control of my Sphere while talking to the central system, can capture files from the system, and generate data off-line and then later transmit it to the system (saving it on cassette meanwhile).

I have several pages of documentation that I went through, that I'm willing to send free to anyone who sends me a self-addressed envelope stamped for 2 ounces. It includes pages from the CDC 752 manual. The EIA RS-232-C Terminal Interface specification is available for $6.90 from Electronic Industries Association, Engineering Dept., Standards Orders, 2001 EYE St. NW, Washington DC 20006. You might request their EIA and JEDEC Standards and Engineering Bulletins catalog (free) before ordering.

5. CLOCK AND TTL/RS232 SWITCHES

Some time back, I decided to use the spare IC position at E31 for a set of seven dip switches to control my jumpers in the ACIA1 circuit. Cut the +5 lead to the position, and wired the two sides of each jumper to the two sides of each switch, in the following configuration

1 -- TXC not CAS (pre-strapped position for CAS jumper).

2 — RXC not CAS (pre-strapped position for CAS jumper).
3 — RXC CAS (other position for CAS jumper).
4 — TXC CAS (other position for CAS jumper).
5 — CAS1
6 — 20ma jumper between sockets X24 and X25
7 — RS232 jumper above E5.

The TTL jumper was not controlled in this configuration (see below).

This was okay for occasional circuit changes, but when I added the acoustic coupler to the system, I needed switches that were easier to get at. I just glued two Radio Shack miniature SPDT toggle switches to the top of the solder side of the board. One I used to control the clock, with one input connected to the 300 baud clock output (be sure to cut the 300 baud etch), one switch input to the 9600 baud clock output, and the switch output to the clock output on the other end of the jumpers. The other switch was used to control the RS232 and TTL jumpers, with one switch input connected to the RS232 side of the RS232 jumper, the other switch input to the TTL side of the TTL jumper, and the switch output to a nearby ground lead. Now I have the digital data recorder on TTL and the acoustic coupler on RS232, both on the same ACIA. I go back and forth between them just by flipping the two switches.

6. CRYSTAL CLOCK REVISITED

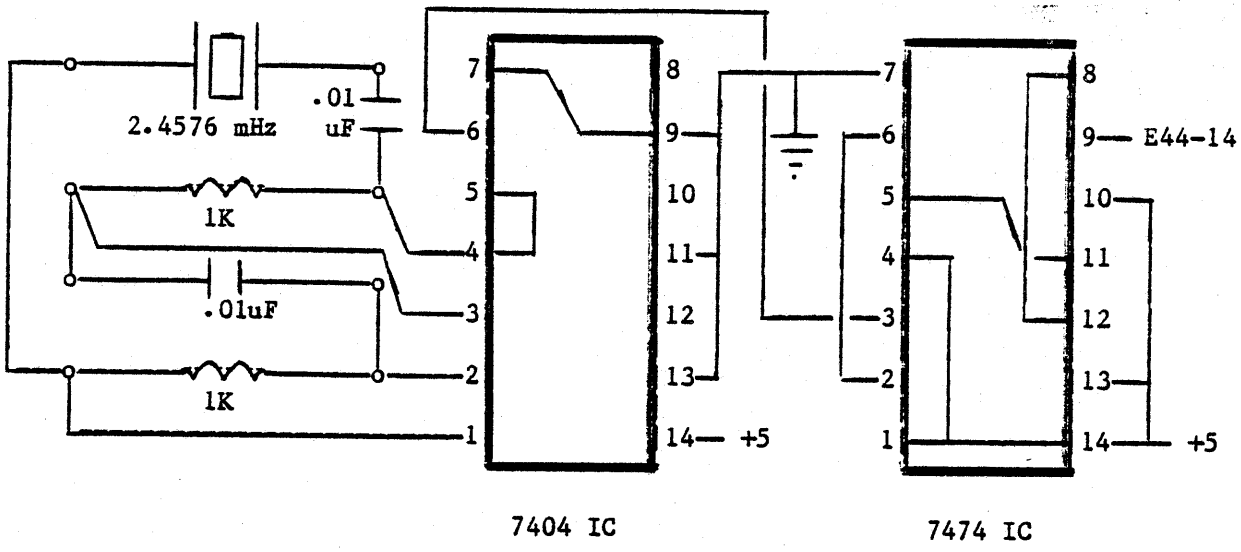
Since publishing the crystal clock circuit in the August 1978 newsletter, I have found several sources for alternate circuits using an IC in the circuit. Byte magazine for October 1978 has an oscillator using a 7404 IC, in an article describing hints for speeding up the SWTPC computer. CQ (Amateur Radio) magazine for July 1979 has an oscillator using a 7400 IC, in an article on schematics. The Quality Computer Parts ad in Byte magazine states that they include free oscillator schematics with any crystal order. The best source I've seen is an article in Ham Radio magazine for February 1979 on crystal oscillators. This article has an excellent description of how such oscillators work, a number of circuits of which some use IC's, and several references for further study. A 2.4576 mHz crystal from Jade Co. can be used in the oscillator circuit. This produces the same frequency as the 9601 that Sphere uses.

Unfortunately, the crystal clock circuit which I published lacks one essential component. The 7493A IC to which the oscillator output connects (at pin 14), cannot handle the clock signals. Common practice is to run the clock through a 7474 IC. If the spare IC position at E31 is available, solder a socket at that position. A 7474 has two sections, each of which divides the clock by 2. Using both sections divides the clock by 4, which is required for connecting a modem as mentioned above in Connecting the Sphere to a Timesharing System. Wire the 7474 as follows:

- Connect together pins 2 and 6.
- Connect together pins 1, 4, 10, 13, 14 (+5 volts).
- Connect together pins 8 and 12.
- Connect together pins 5 and 11 (connects the two sections).
- Connect pin 3 (7474 input) to the oscillator output. This lead formerly went to pin 14 of the 7493A which replaced the 9601 at E43.
- Connect pin 9 (7474 output) to pin 14 of the 7493A IC E44.
- Cut the lead between the 9601 IC E43 pin 8 and E44 pin 14, as close to E44 as possible. This is the thin lead between the two IC's.

Since the 9601 will presumably not be rewired, the ground to the oscillator must be rerouted. This can be done by cutting the lead between pin 13 of the 9601 E43 and capacitor C48. Then connect the junction of the crystal Y75', capacitor C48', and resistor R48' to pin 1 of the 9601 E43.

The following wiring diagram illustrates use of an alternate oscillator circuit, per the article in the October 1978 Byte, as well as wiring of the 7474.



7404 IC

7474 IC

The 7474 can also be used to divide the standard Sphere oscillator output by 4. To do this, wire as above, except pin 3 of the 7474 is connected to pin 8 of the 9601 instead of the oscillator output. Note that under the crystal clock scheme and a 7474, the 9601 IC position (E43) is not used.

If the spare socket at E31 is not available, but the divide by 4 is required, a 7493A (which I will call E44T) can be mounted piggyback onto the 7493A at E44. Proceed as follows:

- Clip off pins 4, 6, 7, 9, 11, 12, 13, 14 of E44T. They are unused.
- Bend out pins 1 and 8 of E44T at right angles.
- Put E44T on top of E44 and solder together pins 2, 3, 5, 10 of E44T to E44 (pin 2 of E44T to pin 2 of E44, etc.).
- Pin 1 of E44T is the clock input. Cut the circuit board lead from pin 8 of E43 to pin 14 of E44, as close to E44 as possible. This is the thin lead between the two IC's. Connect a jumper from the cut lead on the E43 side, to pin 1 of E44T.
- Pin 8 of E44T is the divided by 4 output. Connect it to pin 14 of E44.

7. CRT ADDRESS AT D000

If MIKBUG or a similar monitor is to be run at E000, the screen must be relocated to D000 or some other convenient address. To relocate it to D000 requires a trivial change to the CRT board addressing, and several changes to the PDS monitor. The binary code for E is 1110, and D is 1101. Inspection indicates that the difference is that the A12 and A13 address lines are inverted between the two values. Line A12, as originally implemented, goes through an inverter at pins 11-10 of IC E13. A13 is not inverted. By reversing the polarity of the two lines, the conversion can be implemented. To do this,

- Cut the traces under the X1 socket that tie to pins 10 and 11.
- Connect X1 pin 11 to E7 pin 5.
- Connect X1 pin 10 to E13 pin 11

The two address lines are now reversed, so that A13 is inverted, and A12 is not, whereas formerly the reverse was true.

All references to the screen in PDS and the SIM PROM must now be changed from Exxx to Dxxx. See the newsletter for August 1977, for these changes.

8. WIRING TTL CIRCUIT TO ACIA2 ON THE SIM BOARD

I started with a bare SIM board on this project. The goal was a board with RS232 at 300 baud on one ACIA, and TTL at 9600 baud on the other ACIA. Some of the things I ran into include

- Make sure all the power components are on the board. I used 1000 uF for C12, and 250 uF for the 47 uF capacitors (3 of these). Diodes D1-D8 are required, all 1N4001's.
- Omit unneeded circuits. It makes things easier to find. I omitted the CAS1, CAS2, modem, and 20ma circuits.
- E5 (MC1489) and E13 (MC1488) are required for RS232. E14 (7404) and E36 (7407) are required for both TTL and RS232.
- Crystal clock is required for use with an acoustic coupler. See Crystal Clock above.
- Only four pins of the ACIA2 need be changed--pin 2 (RxD), pin 3 (RxC), pin 4 (TxC), and pin 6 (TxD).

Disconnect pin 3 of ACIA2 (E32) from E39 pin 3. Disconnect pin 4 of ACIA2 from E40 pins 9 and 12. Omitting E39 and E40 will do. Connect pins 3-4 of ACIA2 to the desired baud rate output above E44-E45. My clock runs at 4.9152 mHz, and with only a divide by 4 in the 7474, I was 2X high. I therefore used the 4800 baud rate to get 9600 baud.

Inspection of the Rx circuit for ACIA1 indicates that the Rx TTL should at least go through the 7407 at E36. Also, the Rx TTL circuit has a bug which produces inverted output. I bypassed the inverter at E14 pins 9-8 by cutting the lead going to pin 9, next to pin 9, and connecting a jumper from the hole just above pin 9 (normally used to ground the TTL line), to the hole just above and to the left of the letters "R38" on the component side of the board. I then cut the short lead from this hole to the hole above it, on the solder side. This lead normally connects E14 pin 8 to E36 pin 11.

Disconnect ACIA2 pin 2 from E47 pin 12 Omitting E47 will do. Connect E36 pin 10 to ACIA2 pin 2. A 1K pullup resistor is required on this line. There are two holes in the lead from ACIA2 pin 2 to E47 pin 12. Either can be used to connect the resistor.

E36 is messy for this operation, as all of the TTL, RS232, and 20ma receive circuits tie together here at pins 6, 8, and 10. Pins 8 and 10 can be cut apart on the component side. Pins 6 and 10 connect under E36, on the component side. Cutting them apart may be difficult. Since I had omitted the 20ma circuit, I just ran a jumper from E14 pin 11 to ground (near pin 14).

Inspection of the Tx circuit for ACIA1 indicates the Tx TTL circuit should probably go through a pair of inverters, for driving capability. Disconnect ACIA2 pin 6 from E46 pin 8. Again, omitting E46 will do. Connect ACIA2 pin 6 to E14 pin 9. This is the inverter freed up from the Rx TTL circuit. Connect E14 pin 8 to pin 5. A jumper from the other end of the short lead cut mentioned above (hole just to left of letters "R37" on the component side) to pin 5 will do. Connect E44 pin 6 to X25 pin 13. Cut the lead from X25 pin 13 to E13 pin 13, near E13 pin 13.

This completes rewiring of ACIA2. ACIA1 (E19) needs only a minor adjustment to the Rx circuit. E36 pin 8 is no longer connected to ACIA1 pin 2. Therefore, connect a jumper from E36 pin 8 to the hole in the lead coming from pin 6 on the solder side (near E42 pin 5). Cut this lead between the hole and pin 6.

In my case, since my clock was high, I had to cut the 300 baud lead above E45, and connect the 110 (actually 150) baud rate in its place.

#P!

Oct. 18, 1979

Robert Ennis
9322 Laurel Ave.
Fontana, Calif. 92335

Jeffrey E. Brownstein
2 Tor Road
Wappingers Falls, N.Y. 12590

Dear Jeff;

Received your note a couple of weeks ago, and you asked about any other interesting software. I have included a few listings of some BASIC stuff you may enjoy, also a break test (↑C or whatever) that patches my I/O in EPROM to MSI disk extended BASIC version 1.4. Also included a program that will allow your SPHERE to talk! If you need a version of it at a different address, or the source, let me know.

I have been using a dual-drive MSI FD-8 disk system with my SPHERE for about 2 years now, with EXCELLENT results. The only hardware required to interface the disk controller board (which is inside the drive Ø cabinet) is to address a single PIA chip someplace. I put mine at $FØ7Ø-$FØ73, a SPHERE PIM board would do the job nicely and have PIAs left over. Also required is memory in the $AØØØ-$AFFF range so you can quit wondering 'where to put the stack'; also I moved DOS to start at $A1ØØ. With the disk I/O in a 2708 EPROM (on one of Mel's piggy-back boards) booting up DOS is done by typing ↑A from Sphere EXEC. MSI's disk software is great and gives you a disk text editor that will handle long files in segments, a text processor, disk assembler, etc,etc; and they just released a new version (1.4) of disk BASIC with all the goodies - print using, renumber, paged listings, etc.

There are three guys in the area building up systems with SPHERE boards obtained from Charles Matteson; I am working to get the clock speed on mine up to 2 MC. We look forward to the 64 character/line mod in the next newsletter.

Thanks;

Bob.

```
0349                    * PACK 8 BITS FROM BUFFER
0350                    * INTO ONE BYTE
0351                    *
0352  E9E0 37           PACK    PSHB
0353  E9E1 C6 08              LDAB  #8
0354  E9E3 7F 0018            CLR   BYTE
0355  E9E6 B6 F070      PACK1   LDAA  PIAAD
0356  E9E9 84 40              ANDA  #$40
0357  E9EB 48                 ASLA
0358  E9EC 74 0018            LSR   BYTE
0359  E9EF 9A 18              ORAA  BYTE
0360  E9F1 97 18              STAA  BYTE
0361  E9F3 7C F072            INC   PIABD
0362  E9F6 26 05              BNE   PACK2
0363  E9F8 7C 0011            INC   BUFFH
0364  E9FB 8D 0D              BSR   RPIABF
0365  E9FD 5A           PACK2   DECB
0366  E9FE 26 E6              BNE   PACK1
0367  EA00 96 18              LDAA  BYTE
0368  EA02 33                 PULB
0369  EA03 39                 RTS
0370                    *
0371                    * SET UP CONTROLLER BUFFER
0372                    * FOR WRITE
0373                    *
0374  EA04 37           WPIABF  PSHB
0375  EA05 36                 PSHA
0376  EA06 C6 2C              LDAB  #$2C
0377  EA08 20 04              BRA   4+*
0378                    *
0379                    * SET UP CONTROLLER BUFFER
0380                    * FOR READ
0381                    *
0382  EA0A 37           RPIABF  PSHB
0383  EA0B 36                 PSHA
0384  EA0C C6 04              LDAB  #4
0385  EA0E 37                 PSHB
0386  EA0F 96 11              LDAA  BUFFH
0387  EA11 C6 0B              LDAB  #$B       BUFFER MSB CNTL WORD
0388  EA13 8D 5C              BSR   WINIT
0389  EA15 7F F073            CLR   PIABC
0390  EA18 C6 FF              LDAB  #$FF
0391  EA1A F7 F072            STAB  PIABD
0392  EA1D 33                 PULB
0393  EA1E F7 F073            STAB  PIABC
0394  EA21 7F F072            CLR   PIABD
0395  EA24 C6 1C              LDAB  #$1C      PERM LSB BUFF ENABLE
0396  EA26 F7 F070            STAB  PIAAD
0397  EA29 32                 PULA
0398  EA2A 33                 PULB
0399  EA2B 39                 RTS
0400                    *
0401                    * WAIT FOR SECTOR BEFORE THE
0402                    * ONE YOU WANT AND THEN
0403                    * PULL TRANS FLAG
0404                    *
0405  EA2C 96 01        WAITS   LDAA  SECTOR
```

81 10 CmpA #16

```
0407  EA30 25 03          BCS   WAITS1
0408  EA32 7E E961        JMP   SEKERR
0409  EA35 96 01    WAITS1 LDAA  SECTOR
0410  EA37 26 02          BNE   2+*
0411  EA39 86 10          LDAA  #16
0412  EA3B 4A             DECA
0413  EA3C 48             ASLA
0414  EA3D 97 1D          STAA  COUNTR
0415  EA3F 8D 21          BSR   WAISEC
0416  EA41 7C 001D        INC   COUNTR
0417  EA44 8D 1C          BSR   WAISEC
0418  EA46 17       TRANS TBA
0419  EA47 C6 09          LDAB  #9
0420                      *
0421                      * PULSES CA2 FOR WRITE ENABLE
0422  EA49 36             PSHA
0423  EA4A 8D 53          BSR   PWRITE
0424  EA4C 86 34          LDAA  #$34
0425  EA4E B7 F071        STAA  PIAAC
0426  EA51 32             PULA
0427  EA52 8D 1F          BSR   WINIT1
0428  EA54 C6 18          LDAB  #$18
0429  EA56 8D 5B          BSR   PBIN
0430                      *
0431  EA58 8D 2C    TRANS1 BSR   RINIT1
0432  EA5A 84 04          ANDA  #4
0433  EA5C 27 FA          BEQ   TRANS1
0434  EA5E C6 19          LDAB  #$19
0435  EA60 20 0F          BRA   WINIT
0436                      *
0437  EA62 37       WAISEC PSHB
0438  EA63 C6 1A    WAISE1 LDAB  #$1A
0439  EA65 8D 1D          BSR   RINIT
0440  EA67 84 1F          ANDA  #$1F
0441  EA69 91 1D          CMPA  COUNTR
0442  EA6B 26 F6          BNE   WAISE1
0443  EA6D 33             PULB
0444  EA6E 39             RTS
0445                      *
0446                      * WRITE A CONTROL WORD
0447                      *
0448  EA6F C6 0D    CONWRT LDAB  #$D
0449                      *
0450                      * WRITE TO CONTROLLER
0451                      *
0452  EA71 8D 2C    WINIT  BSR   PWRITE
0453                      * SECOND ENTRY
0454  EA73 F7 F070  WINIT1 STAB  PIAAD
0455  EA76 B7 F072        STAA  PIABD
0456  EA79 CA 10          ORAB  #$10
0457  EA7B F7 F070        STAB  PIAAD
0458  EA7E C4 2F          ANDB  #$2F
0459  EA80 F7 F070        STAB  PIAAD
0460  EA83 39             RTS
0461                      *
0462                      * READ FROM CONTROLLER
0463                      *
0464  EA84 8D 2B    RINIT  BSR   PREAD
```

DRIVE?00
NAME TRK SEC #SEC TYPE 15 START END CALL

DISK #44
ASSEMBLER SOURCE

OLDBASIC,,,,,,,,	03	00	004F	22	0100	3FFF	0100
SPHEREBA,,,,,,,,	07	0F	004C	22	0200	3D00	0200
,,,,,,,,	0C	0B	004F	00	0100	3FFF	0100
,,,,,,,,	11	0A	004F	00	0100	3FFF	0100
$PFILES ,,,,,,,,	16	09	0003	11	0100	02FF	0100
$COPYFIL,,,,,,,,	16	0C	0006	11	0100	04FF	0100
$DCORES ,,,,,,,,	17	02	002E	11	0020	23FF	0100
$COPY ,,,,,,,,	1A	00	0004	11	0200	04EB	0200
MUSICKBD,,,,,,,,	1A	04	0033	77	4000	67DA	3230
MEMTALK ,,,,,,,,	1D	07	0013	77	4000	4BE4	3730
MORSEND ,,,,,,,,	1E	07	0022	77	4000	5D79	3430
PORT ,,,,,,,,	20	09	000D	77	4000	49EF	3530
MEMTALKR,,,,,,,,	21	06	002C	22	AB00	CDC5	CC80
MUSIC ,,,,,,,,	24	02	0008	22	0200	0749	0200
TALKER ,,,,,,,,	24	0A	0016	77	4000	504E	3130
TALKLOW ,,,,,,,,	26	00	0016	77	4000	505A	3130
TALKINST,,,,,,,,	27	06	0004	88	0000	0000	0000
BASPATCH,,,,,,,,	27	0A	000B	77	4000	47B5	3330
SWTPI/O ,,,,,,,,	28	05	002A	77	4000	630A	3530
MON5 ,,,,,,,,	2A	0F	002C	77	4000	62B5	3030
DEDITPAT,,,,,,,,	2D	0B	0006	77	4000	4381	3230
EDITPATC,,,,,,,,	2E	01	0006	77	4000	4384	3930
MINIDOS ,,,,,,,,	2E	07	000B	77	4000	46BE	3930
COPY ,,,,,,,,	2F	02	002B	77	4000	642A	3530
RTTYCOMM,,,,,,,,	31	0D	003F	77	4000	789B	3630
COMM.GWT,,,,,,,,	35	0C	0042	77	4000	742D	3630
COMM.RFU,,,,,,,,	39	0E	0045	77	4000	7417	3630
PIE ,,,,,,,,	3E	03	004F	77	4000	7E55	3930
$PORT ,,,,,,,,	43	02	0003	11	7000	712B	7000
SPEAK ,,,,,,,,	43	05	0007	77	4000	4424	3430
$ASSEMBL,,,,,,,,	43	0C	0024	11	0020	23FF	0100

DOS READY
PFILES

PRINTOUT ON TERMINAL OR PRINTER (T OR P)? T
DRIVE?01
NAME TRK SEC #SEC TYPE START END CALL

DISK #34,
MISC

$BASIC ,,,,,,,,	03	00	004F	11	0100	3FFF	0100
SA-140 ,,,,,,,,	07	0F	000C	55	4000	4829	6E00
POKER ,,,,,,,,	08	0B	0022	55	4000	5A2F	6E00
STARWARS,,,,,,,,	0A	0D	003A	55	4000	6DA7	6E00
ELIZA ,,,,,,,,	0E	07	0020	55	4000	5890	6E00
ELCALC ,,,,,,,,	10	07	0011	55	4000	4CB4	6E00
BIORYTH ,,,,,,,,	11	08	000D	55	4000	49F8	6E00
ASEMBLER,,,,,,,,	12	05	0013	22	0200	107B	0200
COMM ,,,,,,,,	13	08	0007	22	0200	069F	0200
$COPYFIL,,,,,,,,	13	0F	0006	11	0100	04FF	0100
DCORES ,,,,,,,,	14	05	002E	22	0020	23FF	0100
$PACK ,,,,,,,,	17	03	0004	11	00C0	0294	0100
BASPATCH,,,,,,,,	17	07	0007	77	4000	4481	3130
SWTPI/O ,,,,,,,,	17	0E	002C	77	4000	6240	3130
TEXTEDIT,,,,,,,,	1A	0A	001B	22	00B0	1492	0200
CHEKBOOK,,,,,,,,	1C	05	0002	55	4000	4096	6E00
$PORT ,,,,,,,,	1C	07	0003	11	7000	712B	7000
$PFILES ,,,,,,,,	1C	0A	0003	11	0100	02FF	0100

RTTY COMM V.1.9
LETTER

NEWER VERSION - OF YOU PUBLISHED IN NEWS

16

```
$PFILESP,,,,,,,,  1C 0D 0004 11 0100 034F 0100
$DCORES ,,,,,,,,  1D 01 002E 11 0020 23FF 0100
$BBASIC ,,,,,,,,  1F 0F 002F 22 0100 25FF E800
AMAZIN  ,,,,,,,,  22 0E 0011 55 4000 4C0B 6E00
MADLIB  ,,,,,,,,  23 0F 0018 55 4000 52A1 6E00
PARTSORD,,,,,,,,  25 07 0009 55 4000 4698 6E00
TEST    ,,,,,,,,  26 00 0002 88 0000 0000 0000
$EDIT   ,,,,,,,,  26 02 002A 11 0040 2095 0200
SPL/H   ,,,,,,,,  28 0C 003B 22 0300 316A FC00
$MINIDOS,,,,,,,,  2C 07 0002 11 AB00 ABB8 AB00
$ASSEMBL,,,,,,,,  2C 09 002E 11 0020 23FF 0100
ASSTEST ,,,,,,,,  2F 07 0003 88 0000 0000 0000
$BASIC2 ,,,,,,,,  2F 0A 004F 11 0100 3FFF 0100
COPY    ,,,,,,,,  34 09 002B 77 4000 642A 3530
$COPY   ,,,,,,,,  37 04 0004 11 0200 04EB 0200
DEDITPAT,,,,,,,,  37 08 0006 77 4000 4381 3230
PORT    ,,,,,,,,  37 0E 000D 77 4000 49EF 3530
RTTYCOM ,,,,,,,,  38 0B 003F 77 4000 789C 3630
SWTPCOMM,,,,,,,,  3C 0A 0045 77 4000 7417 3630
COMM1   ,,,,,,,,  40 0F 004D 77 [illegible]
DIAL    ,,,,,,,,  45 0C 0007 77 [illegible]

DOS READY
PFILES

PRINTOUT ON TERMINAL OR PRINTER (T OR P)? T
DRIVE?00
NAME             TRK SEC #SEC  TYPE   START  END   CALL

$BASIC  ,,,,,,,,  03 00 003E 11 0100 3E6E 0100
$PORT   ,,,,,,,,  06 0E 0003 11 7000 712B 7000
$EDIT   ,,,,,,,,  07 01 002A 11 0040 2095 0200
$COPY   ,,,,,,,,  09 0B 0004 11 0200 04EB 0200
$COPYFIL,,,,,,,,  09 0F 0006 11 0100 04FF 0100
$DCORES ,,,,,,,,  0A 05 002E 11 0020 23FF 0100
$PFILES ,,,,,,,,  0D 03 0003 11 0100 02FF 0100
AMAZIN  ,,,,,,,,  0D 06 0011 55 4561 5152 6E00
ELIZA   ,,,,,,,,  0E 07 0020 55 4561 5DF1 6E00
BIORYTH ,,,,,,,,  10 07 000E 55 4561 4F59 6E00
POKER   ,,,,,,,,  11 05 0022 55 4561 5F90 6E00
MADLIB  ,,,,,,,,  13 07 001A 55 4561 5809 6E00
STARWARS,,,,,,,,  15 01 003B 55 4561 7312 6E00
HAIKU   ,,,,,,,,  18 0C 0017 55 4561 562A 6E00
POSTER  ,,,,,,,,  1A 03 0016 55 4561 55CD 6E00
FORT    ,,,,,,,,  1B 09 002B 55 4561 6699 6E00
APHORISM,,,,,,,,  1E 04 0007 55 4561 499A 6E00
BLACKJAK,,,,,,,,  1E 0B 001F 55 4561 5C31 6E00
TICTAC  ,,,,,,,,  20 0A 0024 55 4561 60E0 6E00
DECISION,,,,,,,,  22 0E 0016 55 4561 55B3 6E00
LUNAR   ,,,,,,,,  24 04 000D 55 4561 4E8A 6E00
PASART  ,,,,,,,,  25 01 000D 55 4561 4E8D 6E00
PATTERNS,,,,,,,,  25 0E 0008 55 4561 4AED 6E00
LIFETIME,,,,,,,,  26 06 0026 55 4561 62BE 6E00
CALENDER,,,,,,,,  28 0C 0015 55 4561 544B 6E00
CHECKERS,,,,,,,,  2A 01 0012 55 4561 5251 6E00
MORSE   ,,,,,,,,  2B 03 0004 55 4561 47C3 6E00

DOS READY
```

DISK #40,
BASIC GAMES, ETC

TYPE	= FILE TYPE
11	= SYSTEM
22	= OBJECT CODE
33	= BASIC DATA FILE
55	= BASIC SOURCE
77	= DCORES ASSEMBLER SOURCE
88	= TEXT EDITOR FILE MAY be ASSEMBLER OR BASIC SOURCE Also.

JEFF-
IF ANYTHING HERE LOOKS INTERESTING, MAYBE WE SWAP SOME SOFTWARE
O

PAGE 001 BASPATCH

```
00010                          NAM    BASPATCH
00020                  *
00030                  * I/O PATCHES FOR MSI BASIC VERSION 1.4
00040                  *      by Bob Ennis, Aug. 1979
00050                  *
00060                  *     BREAKTEST
00070                  * RETURNS CARRY SET AND CHARACTER
00080                  * IN ACC A IF INTERRUPT FROM
00090                  * CURRENT I/O DEVICE
00100                  *
00110       F040       CPUPIA EQU    $F040     PIA ON CPU/2 FOR KBD & BITBAN
00120       F050       TTACIA EQU    $F050     SIM/1 ACIA FOR TTY, ETC
00130       F060       MTACIA EQU    TTACIA+16  ACIA #2 FOR CASSETTE
00140       EC6E       INCHAR EQU    $EC6E     I/O EPROM
00150       EC69       OUTCHR EQU    $EC69
00160       A090       CURSOR EQU    $A090     ADDRESS OF CURSOR ON CRT
00170       A092       IOPORT EQU    $A092     CURRENT I/O TERMINAL #
00180       A0A0       XTEMP1 EQU    $A0A0     INDEX TEMP
00190                  *
00200 0235                        ORG    $235
00210                  *
00220 0235 B6 A092 BRKTST LDA A  IOPORT   GET I/O TERMINAL ASSIGNMENT
00230 0238 4A             DEC A
00240 0239 26 0C          BNE    NOTCRT
00250                  *
00260 023B B6 F041 CONSOL LDA A  CPUPIA+1 CONTROL REGISTER
00270 023E 01             NOP
00280 023F 48             ASL A
00290 0240 48             ASL A
00300 0241 24 25          BCC    RETURN
00310 0243 B6 F040        LDA A  CPUPIA   GET CHARACTER
00320 0246 39             RTS
00330                  *
00340 0247 4A       NOTCRT DEC A
00350 0248 26 05          BNE    NOTTTY
00360                  *
00370 024A CE F050 TELETY LDX    #TTACIA  POINT @ ACIA
00380 024D 20 06          BRA    ACIA
00390                  *
00400 024F 4A       NOTTTY DEC A
00410 0250 26 0B          BNE    BAUDOT
00420                  *
00430 0252 CE F060 MAGTAP LDX    #MTACIA
00440 0255 A6 00    ACIA  LDA A  0,X       GET STATUS BYTE
00450 0257 47             ASR A
00460 0258 24 0E          BCC    RETURN
00470 025A A6 01          LDA A  1,X       GET CHAR
00480 025C 39             RTS
00490                  *
00500 025D B6 F042 BAUDOT LDA A  CPUPIA+2
00510 0260 43             COM A                CARRY SET IF B7=0 (SPACING)
00520 0261 48             ASL A
00530 0262 24 04          BCC    RETURN
00540 0264 BD EC6E        JSR    INCHAR
```

```
00550 0267 0D              SEC             FLAG INTERRUPT
00560 0268 39       RETURN RTS


00570                      *
00580                      * I/O PATCHES TO MAKE
00590                      * CURSOR BACKSPACE, ETC
00600                      *
00610 0269 B6 A092  INCHR  LDA A  IOPORT
00620 026C 4A              DEC A
00630 026D 27 03           BEQ    CONSLE
00640 026F 7E EC6E  JUMPIN JMP    INCHAR   MULTIPORT INCHAR SUBROUTINE
00650 0272 8D FB    CONSLE BSR    JUMPIN
00660 0274 81 08           CMP A  #8       ↑H BACKSPACE
00670 0276 26 F0           BNE    RETURN
00680 0278 FF A0A0         STX    XTEMP1
00690 027B FE A090         LDX    CURSOR
00700 027E 09              DEX
00710 027F FE A090         STX    CURSOR
00720 0282 36              PSH A
00730 0283 86 20           LDA A  #'
00740 0285 A7 00           STA A  0,X      CLEAR BAD CHARACTER
00750 0287 32              PUL A           RESTORE BACKSPACE CHARACTER
00760 0288 FE A0A0         LDX    XTEMP1
00770 028B 39              RTS
00780                      *
00790 028C 84 7F    OUTCH  AND A  #$7F     STRIP PARITY
00800 028E 81 7F           CMP A  #$7F     IGNORE RUBOUTS
00810 0290 27 D6           BEQ    RETURN
00820 0292 37              PSH B
00830 0293 F6 A092         LDA B  IOPORT
00840 0296 5A              DEC B
00850 0297 26 06           BNE    OK
00860 0299 81 5F           CMP A  #$5F
00870 029B 23 02           BLS    OK
00880 029D 80 20           SUB A  #32      MAKE UPPERCASE
00890 029F 33       OK     PUL B
00900 02A0 7E EC69         JMP    OUTCHR
00910                      *
00920                      *
00930                      END

TOTAL ERRORS 00000
```

ENTER PASS : 1P,2P,2L,2T

November 5, 1979

Dear Jeff and Roger:

I have promised several times to send you the hardware interface for the Sphere to Smoke Signal Broadcasting (SSB) 5¼" disk drive. I have finally finished it up and it works well.

A few precautions are in order:

1) The Sphere system must be run at a clock rate somewhere above approximately 850KHZ. I have been running at about 1 MHZ. In order to run this fast the 1702 PROM's must be changed, their access time is typically too slow. I recommend the 2708 change over with the add-on board available from Programma International.

2) Memory must be made up of static RAM. I use the up D410 chips from Nippon Electric Co. (NEC). I tryed several schemes to use the present dynamic RAM chips, but was never successful.

3) In order to make the software work, since the SSB disc system is essentially Southwest Tech based, use a MIKBUG (Motorola) or a SMARTBUG ROM from Ed Smith's Software Works. This requires address space at $E000_{16}$ (right in the middle of Sphere's screen memory). I moved by screen addressing to $D000_{16}$ to eliminate the conflict.

4) Memory must be added, 4K bytes at 7000_{16}, to provide RAM storage for the SSB DOS operating system. Also, memory is required at $A000_{16}$, 128_{16} bytes worth, to handle the stack and scratchpad of the SMART-BUG or MIKBUG ROM's. I did this by strategically placed 16K RAM boards, partially populated. It

is a kind of messy way to get there. I tryed adding a 128 byte RAM chip on the CRT board to take care of $A000_{16}$ but failed in that attempt, because of the unique way RAM is accessed on CRT, ie, both by refresh counters and CPU board.

I am sure there are easier ways to adapt discs to the Sphere system, but my machine now has the advantage of being a Southwest Tech. machine when desired, with all the available software for that machine.

Enclosed find the schematic for the interface. I wired it up on a piece of perf board extending the SSB controller board to make it the same size as a Sphere board so it will fit in the Sphere rack. I bolted it onto the SSB controller board and used male MOLEX connectors to connect to the SSB board connectors.

The only change to the SSB controller board requires shorting out the + 5V on-board regulator since Sphere supplies +5V directly with out on-board regulators.

The cost of the SSB Disc System is around $795 for one drive and $1190 with two drives. Software support is included. You get the DOS and File BASIC (Southwest Tech BASIC modified to work with disk files) free.

Sincerely,

Warren Weimer

Warren Weimer

WARREN W. WEIMER
23025 KINARD AVE.
CARSON, CALIFORNIA 90745

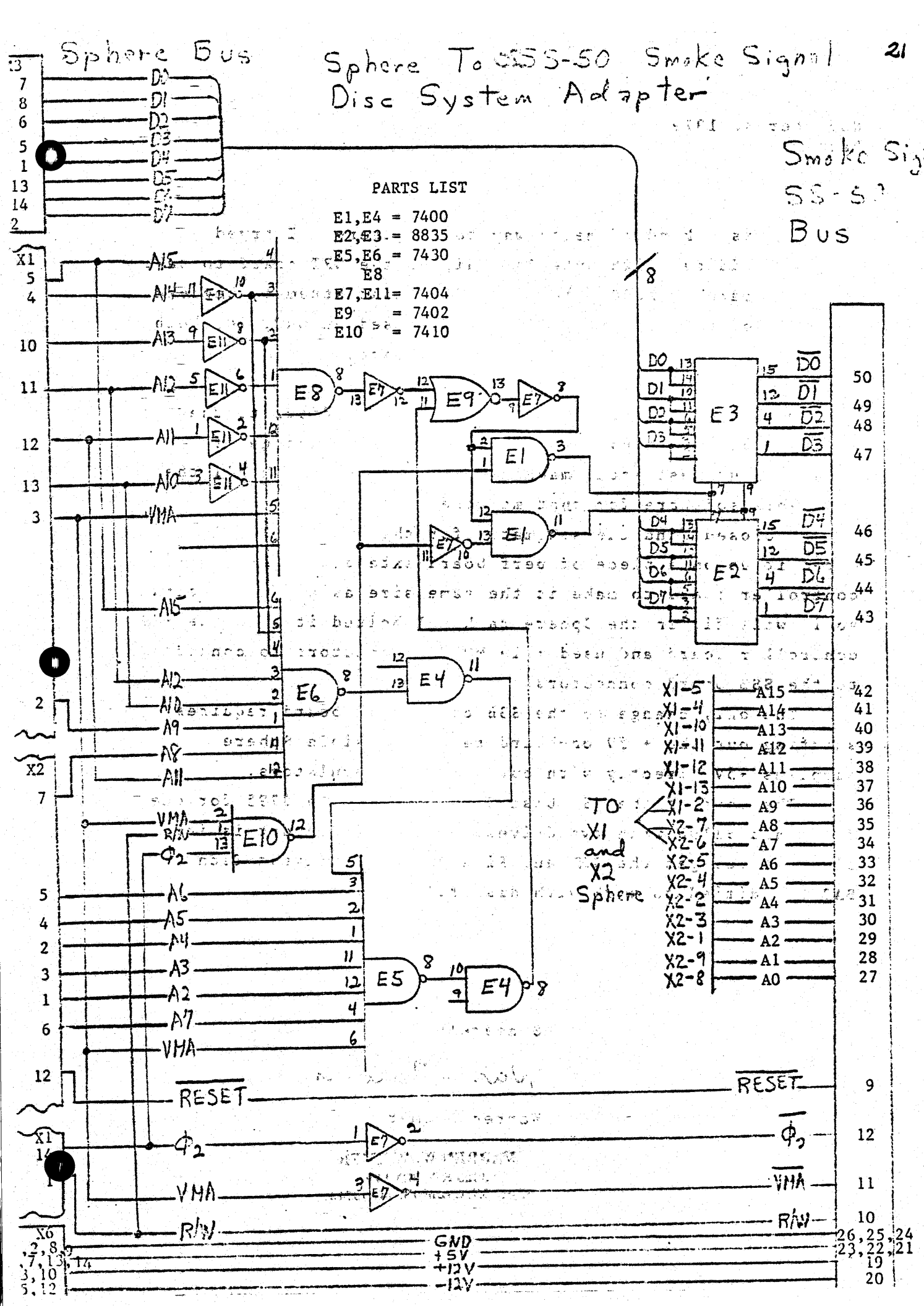
Sphere Bus

Sphere To SSS-50 Smoke Signal Disc System Adapter

Smoke Signal SS-50 Bus

PARTS LIST

E1,E4 = 7400
E2,E3 = 8835
E5,E6 = 7430
E8
E7,E11 = 7404
E9 = 7402
E10 = 7410

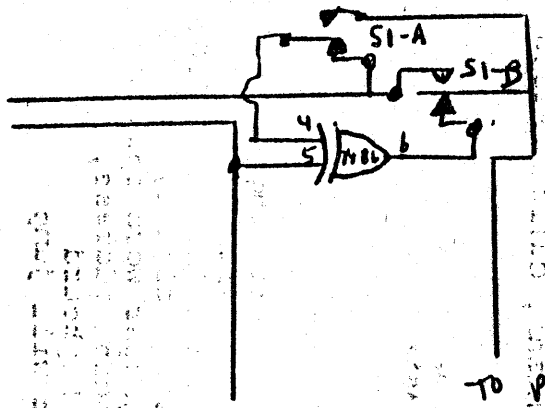


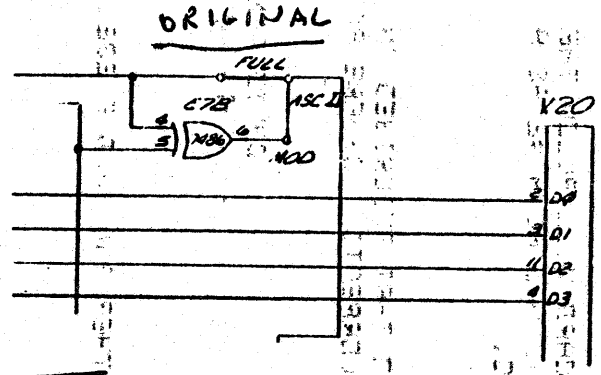
Modify Switch makes Keyboard 2 Act Like Typewriter

By G. H. LATTA

PARTS

1 DPDT switch





CRT 1 With 64 Character Line Displays Upper + Lower Case

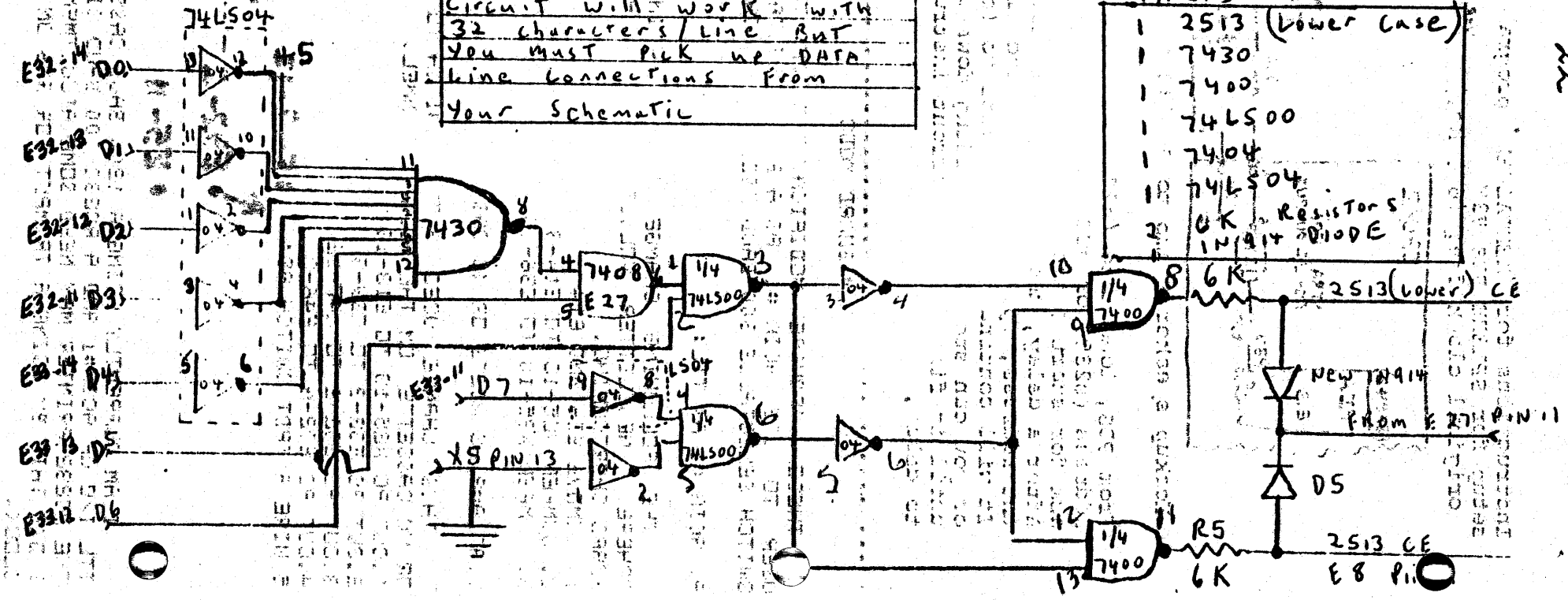
No Software Changes Needed

By Jeff Brownstein

Circuit will work with 32 characters/line but you must pick up DATA line connections from your schematic

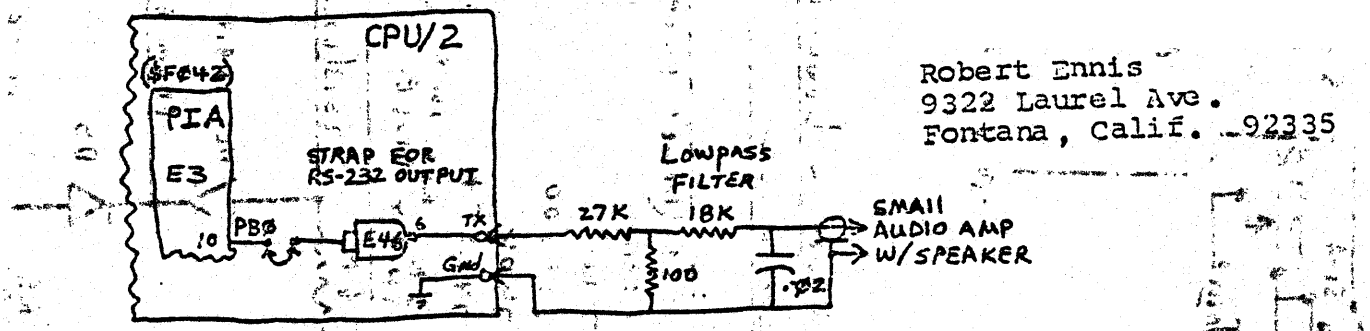
PARTS

1	2513 (Lower case)
1	7430
1	7400
1	74LS00
1	7404
1	74LS04
2	6K Resistors
	1N914 DIODE



Instructions for "MEMTALKER" program
setup for SPHERE PDS V3N
original clock freq (670 KC.) — Ver. 1.1 9/79

Robert Ennis
9322 Laurel Ave.
Fontana, Calif. 92335



Hookup speaker & amp to RS-232 output from CPU/2 PIA B0:

From PDS, load tape ($200-$252F or $AB00-$CE32). Open starting address ($2300 or $CC80) and hit control 'G'. Program will ask for start & ending addresses, enter as 4 hex digits (no CR or ESC) After a delay, the program will start speaking (as hex digits) the address, op code, and operand (if any), with proper word spacing. It will continue until it gets to (or past) the ending address, or you can stop it by holding down any key on the keyboard until it finishes the current instruction, then it will jump to debug.

SIM BOARD MODS

THESE HARDWARE MODIFICATIONS OF THE SIM BOARD WILL ALLOW THE USER TO USE ACIA #1 AS A BAUD SELECTABLE TTL I/O AND ACIA #2 AS A SWITCH SELECTABLE 300 BAUD RS232 OR KC CASSETTE I/O.

A. ACIA #1 AS TTL:

THE FOLLOWING CHANGE IS SIMILAR IN RESULT TO THAT DESCRIBED IN SPHERE NEWSLETTER, VOL.III, ISSUE 1 P. 19 AND CORRECTS A SPHERE PC BOARD SHORTCOMING.

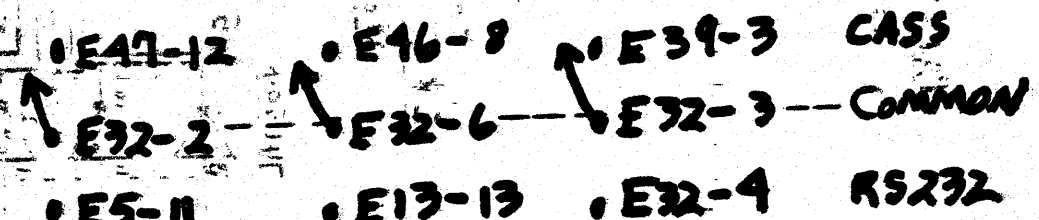
1. REMOVE E36 AND E14
2. JUMPER X25-13 TO E14-3
3. JUMPER X25-10 TO E36-10

DAVID DEMOREST
SANTA ROSA, CA.

B. ACIA#2 AS RS232 OR KC CASS:

THIS CHANGE USES A TPDT SWITCH TO SELECT EITHER 300 BAUD RS232 OR KC CASSETTE ON THE ACIA#2 CHANNEL.

1. CUT PC X25-13 TO E13-13
2. CUT PC E32-3 TO E39-3
3. CUT PC E32-2 TO E47-12
4. CUT PC E32-6 TO E46-8
5. WIRE A TPDT SWITCH AS:



I AM CURRENTLY USING A 1200 BAUD CASSETTE INTERFACE WIRED TO THE TTL I/O OF ACIA#1 AND A RS232 300 BAUD SILENT 700 TERMINAL ON ACIA#2. THE RS232 PRINTER MAY NEED SOME ADJUSTMENT OF 300 BAUD CLOCK CIRCUIT. I DID THIS BY USING A TRIMPOT FOR R75 ON SIM BOARD TO FINE ADJUST THE CLOCK.

3 Puddingstone Road
Morris Plains, N.J., 07950
October 13, 1979

Dr. Jeffrey Brownstein
2 Tor Road
Wappingers Falls, N.Y., 12590

Dear Jeff,

At long last, I have managed to put something together for the newsletter. Forgive the delay as we have been very busy with a wedding in the family and preparations for a long awaited trip abroad.

I am including a program I wrote for resequencing a program written in SWTPC basic. I hope it isn't too long for you. The included sheet explaining its use should be sufficient for anybody who can write a basic program. The only problem I can think of is the possible relocation of the OUTSTRing routine (at $FD8E in the original ROM) in the new ROM from Programma.

I was most interested in the letter regarding CSS basic in the August newsletter. Your previous remarks about it have led me to think it would fit my needs better than the SWTPC basic I'm using now, so please include me in any group purchase if it isn't too late. I'm including a check for 22.50.

Another subject that has been interesting me lately is the possibility of adding a disk to my system. I seem to recall that you were toying with the idea, and that Warren Weimer was tinkering around also, but haven't noticed any recent references to the subject. The thing that has been bothering me about such a move is the prospect of replacing all my dynamic memory with static RAM, it sounds kind of expensive going that route. I have been meaning to take a look at the possibility of using a separate microprocessor and enough static memory to use as a buffer. In any case, I'd be very interested to hear what others have been doing in this area.

Best regards,

John Gibbon

Please Write the Editors for a Copy

My Disk system is the PerSci 8" dual drives with intelligent controller. This is the system that Mel Norell wrote about two years ago. It interfaces directly to the address and data busses, and only requires a 180 byte software interface. Let me know if any reader would like more info.

Roger

```
*****************************************************************
*        SYSTEM EQUATES TO ALLOW USE OF RENUMBER PROGRAM        *
*          BY M. FERGUSON '68 MICRO JOURNAL VOL 1 #4            *
*              JUNE 1979  WITH SPHERE-CSS BASIC V 4.0           *
*                BY JEFF BROWNSTEIN AND ROGER SPOTT             *
*****************************************************************
*
*
TEMP1           EQU     $00F2       *
TEMP2           EQU     $00F4       * THE FOLLOWING TOKENS MUST BE CHECKED FOR
TEMP3           EQU     $00F6       *
TEMP4           EQU     $00F8
TEMP5           EQU     $00FA
ASTM1           EQU     $00FC
ASTM2           EQU     $00FD
ASTM3           EQU     $00FE
ASTM4           EQU     $00FF
MSLINE          EQU     $003E
LSLINE          EQU     $003F
SOURCE          EQU     $0020
NEXTBA          EQU     $0022
TESTNO          EQU     $0483
SKIPSP          EQU     $0969
BASERR          EQU     $0B5A

*
```

```
GOSUB           1F99
GOTO            1F91
RESTORE         1FD2
ERR             20C4
USING           20E5
LINPUT          20DD
```

Dear Roger:

We would like to obtain a set of Schematics and a parts list for the SPHERE parallel Interface board (RIM?). Do you know of someone in the user's group that could send us a copy?

Sincerely,

Mike Scholl

SCHOLLS INC.
4440 ROUND LAKE ROAD WEST
P.O. BOX 43116 • ST. PAUL, MINNESOTA 55164
(612) 636-0892

ERRATA

LATEST CORRECTIONS FOR CSS BASIC

1. CHANGE THE CONTENTS OF 142C,142D TO 25DB (THIS ALLOWS DETECTION OF OVERFLOW FOR THE FOR-NEXT LOOP BUFFER)

2. CHANGE THE FIVE BYTES BEGINNING AT 1D9F FROM TO:

(OLD)	(NEW)
FE 1CF2	39

(MAKES ARCTAN WORK PROPERLY)

3. CHANGES TO BASIC AFTER YOU IMPLEMENT THE 64 CHARACTER PER LINE SCREEN MOD:

ERROR CHANGES TO EDIT:

~~2208 TO 40~~ → 22C8 TO 40
2308 TO 40
22D1 TO 40
24C0 FROM E1A0 TO E380 (FOR LIST COMMAND)

4. CHANGE 18DB FROM 36 TO 29 (THIS IS AN ERROR IN THE TAPE)