Ipso Facto

Issue 42 September 84

INDEX PAGE

A PUBLICATION OF THE ASSOCIATION OF COMPUTER-CHIP EXPERIMENTERS

1984-1985 EXECUTIVE OF THE ASSOCIATION OF COMPUTER CHIP EXPERIMENTERS

President:	John Norris	416-239-8567	Vice-President:	Tony Hill	416-876-4321
Treasurer:	Ken Bevis	416-277-2495	Secretary:	Fred Feaver	416-637-2513

Directors: Ken Bevis - Fred Pluthero - John Norris - Mike Franklin

Newsletter:

Production Manager: Fred Pluthero 416-389-4070

Editors: Fred Feaver
Tony Hill

Publication: Dennis Mildon
John Hanson

Product Mailing: Ed Leslie 416-528-3222 (Publications)

Fred Feaver 416-637-2513 (Boards)

Development Hardware: Mike Franklin
Tony Hill
Ken Bevis

Software: Michael Smith
Wayne Bowdish
Rob Erlich
Dan Thomas

Club Mailing Address

A.C.E.
c/o Fred Feaver
P.O. Box 581
Burlington, Ontario
Canada
L7R 3X5
416-637-2513

ARTICLE SUBMISSIONS:

The content of Ipso Facto is voluntarily submitted by Club Members. While ACE assumes no responsibility for errors nor for infringement upon copyright, the Editors verify article content as much as possible. ACE can always use articles, both hardware and software, of any level or type, relating directly to the 1802 or to micro computer components, peripherals, products, etc. Please specify the equipment or support software upon which the article content applies. Articles which are typed are preferred, and are usually printed first. Please send originals, not photocopy material. We will return photocopies of original material if requested.

PUBLICATION POLICY:

The newsletter staff assume no responsibility for article errors nor for infringement upon copyright. The content of all articles will be verified, as much as possible, and limitations listed (i.e. Netronics Basic only, Quest Monitor required, require 16K at 0000-3FFF etc.). The newsletter will be published every other month, commencing in October. Delays may be incurred as a result of loss of staff, postal disruptions, lack of articles, etc. We apologize for such inconvenience - however, they are generally caused by factors beyond the control of the Club.

MEMBERSHIP POLICY:

A membership is contracted on the basis of a Club year - September through the following August. Each member is entitled to, among other privileges of Membership, all six issues of Ipso Facto published during the Club year.

EDITOR'S CORNER

This marks my eighteenth, and last, editorial. The next issue, number 43, will be edited by Fred Plethero, and should be issued in October.

This edition also marks the end of the seventh club year, so don't forget to renew for next year. Please take a few minutes to fill out the questionaire on the Club Communiqué at the back of this issue. The data will help the new Executive to meet your needs.

For the patient board orderers of the last few months, you will be happy to know that all orders have been shipped, so heat up your soldering irons. The VDU 80 terminal board is coming along well,and club interest has been high. The prospect of a good high resolution memory mapped terminal and its capability for Text Editing and Forth, Basic and Pilot are obvious.

Projects in the works are: an A/D:D/A board, a new keyboard, and a modem. As you may remember, some of these ideas have been around for awhile, and sometimes it takes awhile for a board to be made, it depends on user interest, and the availability of a designer.

Software projects in the works includes an Editor companion to SYMON and SYDOS (SYED of course!), an operating system in Forth, and a new Forth game. In this issue, you will find two game programs, Conquest in Forth, and a TV Typewriter in CHIP 8 AE. A note about the last Club Communiqué which offered Conquest on cassette. That was in error, Conquest is too long to be offered on Cassette, and is available for Disk owners on a disk only. The game requires some 32k of RAM, and of course, ACE Forth.

In the new year, the hardware and software project leaders would like to embark upon a club project. It has been suggested that a computerized burglar alarm/home controller would be a good project, not only producing a good micro application project, but also a useful and needed product in our homes. The project will be based upon the clubs micro board, and will use an A/D:D/A circuit. All that is needed to get the project off the ground is support from the membership by letters, circuit suggestions and software ideas. The best ideas will be put together on an issue by issue basis, and all particpants will be recognized. The first issue will concentrate on the organization and functions of the alarm sensor circuit, so write me with your ideas, with or without schematics.

Finally, I would like to thank all the members of the club, who in the true spirit of the club, have shared their triumphs and the trials with the members.

Happy computing, Mike Franklin.

HIGH QUALITY VIDEO FOR 1802 SYSTEMS

by - Tony Hill 30-481 Pitfield Rd. , Milton , Ontario

After playing around with my 6847 based video display for the past couple of years, I have become tired of trying to do any serious work on its meagre 32 x 16 display. Some interesting programs have been written for the 6847, but to make my computer really useful I wanted a professional quality display. Lacking an acceptable commercial source for such a unit at a price I could justify, I elected to "roll my own". This article is to document my results, which will be generalized this fall and released as an ACE product to the club membership.

In view of the fact that ACE will "soon" be releasing a PCB version of this circuit, I would not recommend anyone trying to hand wire one. In fact, I have not included pin numbers on the schematics (resulting in easier to read drawings) because I do not expect anyone to try to build directly from them. They have been included for the benefit of the dedicated hackers out there who may want to rework it to their own standards anyways. Mostly, this article is to wet your interest for the ACE PCB version of this display when it gets done !

DISPLAY FEATURES:

1) 80 character by 25 line display format standard. (Can be modified for anything from 1x1 to 132x30)
2) Hardware controlled scrolling.
3) Block or underline cursor - solid or flashing.
4) Full upper and lower case character set.
5) Memory mapped video memory for high speed updates.
6) Choice of half intensity, reverse video or extended character set for each displayed character.
7) Horizontal and vertical line drawing capability

REQUIREMENTS:

1) 10 MHz or better video monitor (80 character version)
2) 2K of 1802 address space (D800-DFFF in the schematic)
3) 2-1802 I/O ports (ports 6,7 in schematic - could be modified to use only one port with additional hardware)

The heart of the display is a Motorola 6845 video display generator chip. It looks after display memory addressing, generates the required video timing signals, handles the cursor and allows a hardware scroll function. The use of this chip in the system greatly simplifies the circuit's design by replacing a whole mess of discrete IC's that would otherwise have been required to do the job. It also allows more flexability in adapting the design to other systems.

The 6845 is basically a glorified programmable timer. It divides a dot clock signal down into required video timing signals based on values stored in its internal registers by a CPU. It also generates synchronized address information to allow atached

memory to output required pixel states to turn a CRT beam ON or OFF at the correct time and position to produce a stable display.

THEORY OF OPERATION

The VDU80 circuit can be divided into three major sections, as shown on schematic diagram sheets 1,2 and 3. These are the CPU interface, the data section and the video section.

The CPU interface is a fairly standard combination of 1802 memory and I/O port interface techniques. Memory addresses D800 - DFFF are decoded by U1 & U3 and latched along with the required high order address bits by U5. (These addresses are immediately below those used on the ACE VDU board). U1 and U2 convert 1802 buss control data to the required signals for video memory access. U7 and U8 are data buss buffer/seperators. U4 converts TPA and TPB into an enable (or refresh) clock signal that combines with two 1802 port I/O instructions (ports 6,7) to allow access to the 6845 internal registers (U6). Note that port mapping is used to address the 6845 because it provides the easiest method of matching the timing characteristics of a 68xx series product to the 1802 buss.

The data section of the system contains the 6845 video display control chip (U6), a 2Kx8 RAM chip for data to be displayed (U12) and a 2716 EPROM programmed as a character generator (U14). U9,U10 and U11 switch the video RAM addresses between the 1802 buss and the 6845, with priority going to the 1802. U13 latches ASCII data from the video RAM to select character data from the EPROM character generator U14. This data is passed to the video section of the circuit for output to a CRT monitor.

The video section of the circuit generates the master clock signal used to syncronize the whole system (U18). It also takes parallel data from U14 and converts it to a serial bit stream with a parallel load shift register (U15). This bit stream can be modified by U16,U22 to give reverse video, half intensity or blank characters. Finally, the signal is mixed with a composite sync signal generated by U17,U21,U22 to produce a standard 1 V composite video output.

CONSTRUCTION NOTES

If you decide to build this display from scratch, there are a couple of things to watch out for. I built mine on a prototype card using point to point wiring because wire wrap sockets are expensive and take up a lot of room. This technique works quite well as long as you keep the video section components close together. You may have to play with the values of R1,C1 to get data to latch into U15 correctly but I didn't have too much problem getting it to work.

SOFTWARE

This display requires some specialized software to run it. Next issue I will present an output subroutine that conforms to club standards and will discuss how to program the 6845's internal registers for any particular display format.

SIGNAL DEFINITIONS

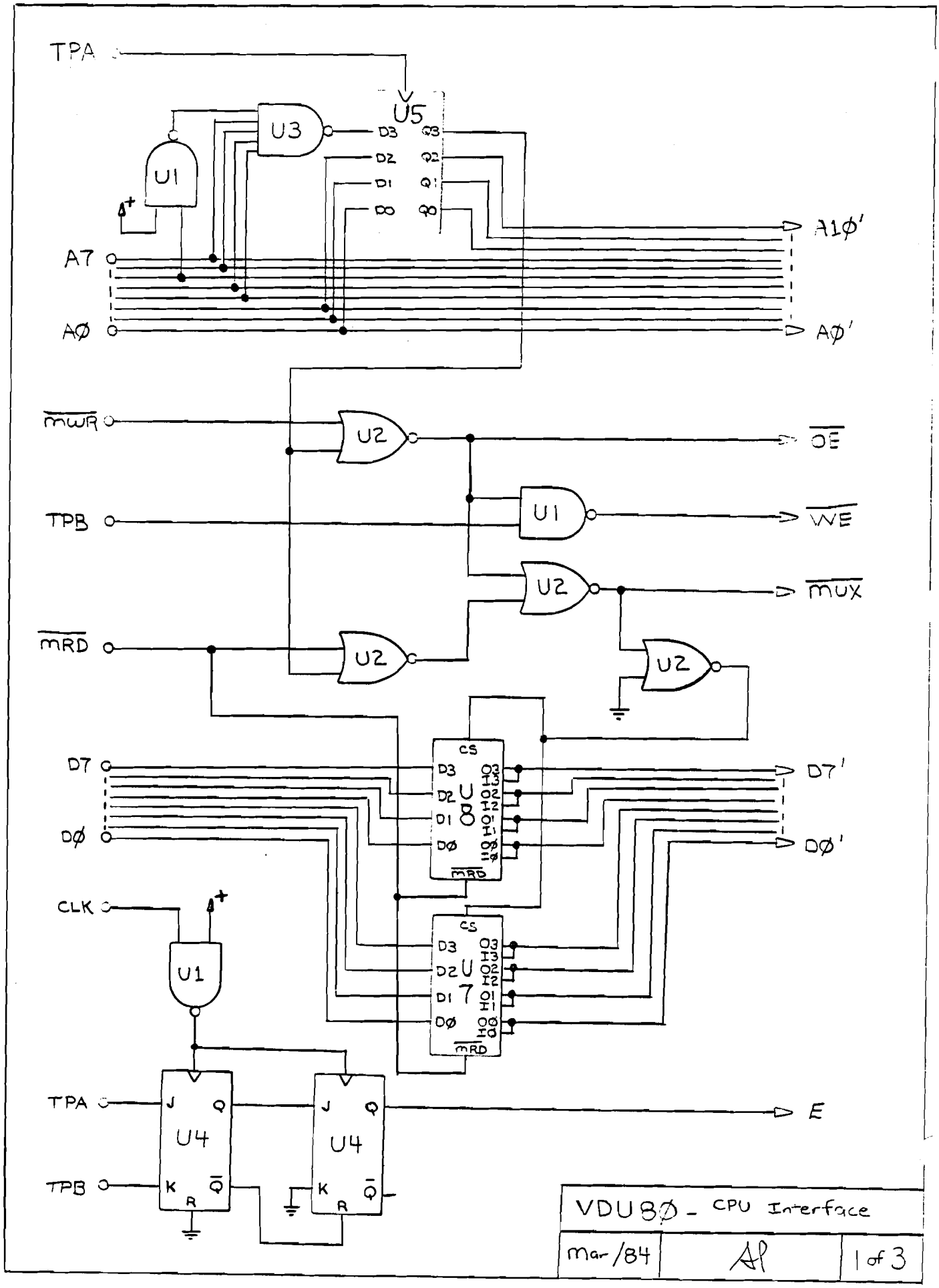
A7-A0	1802 multiplexed address buss (from processor)
A10'-A0'	1802 decoded address buss
CCLK	Character frequency clock
CLK	1802 system clock (from processor)
CURSOR	Cursor display sync signal
D7-D0	1802 data buss (from processor)
D7'-D0'	Video memory data buss
DCLK	Dot frequency clock
DISENA	Display enable signal
DIM	Half intensity character (connect to VD8 to select)
E	Enable signal for the 6845 (internal refresh clock)
HSYNC	Horizontal sync signal
INVERT	Character invert signal (connect to VD8 to select)
$\overline{MRD}$	1802 memory read signal (from processor)
$\overline{MUX}$	Video RAM address buss multiplexor select
$\overline{MWR}$	1802 memory write signal (from processor)
N0,N1,N2	1802 buss N-lines (from processor)
$\overline{OE}$	Video RAM output enable signal
TPA	1802 timing pulse A signal (from processor)
TPB	1802 timing pulse B signal (from processor)
VD7-VD0	Character generator video data buss
VD8	Video RAM memory bit 8 (selects character attributes)
VSYNC	Vertical sync signal
$\overline{WE}$	Video RAM write enable signal

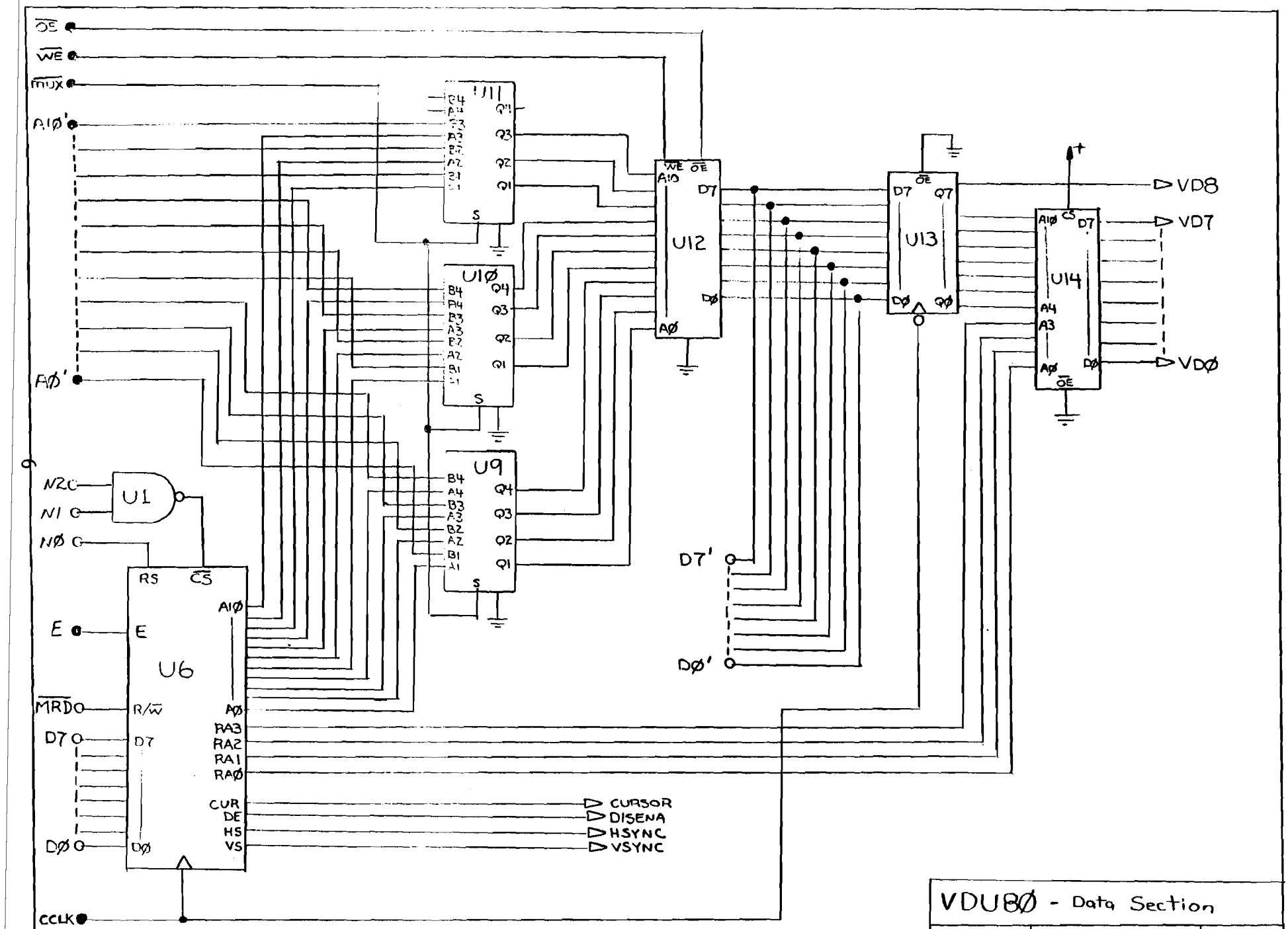
6845 INTERNAL REGISTERS

AR	register mapped by port 6 that selects which register (from R0 to R17) is to be accessed by port 7
R0	total number of characters per display line
R1	number of characters to be displayed per line
R2	horizontal sync pulse position
R3	sync pulse width
R4	total number of character rows per screen
R5	vertical row total adjust (% of full row)
R6	number of character rows to be displayed per screen
R7	vertical sync pulse position
R8	interlace mode and skew select
R9	maximum scan lines per screen select
R10	cursor starts at scan line #
R11	cursor ends at scan line #
R12	video memory start address offset (high byte)
R13	video memory start address offset (low byte)
R14	cursor position in memory (high byte)
R15	cursor position in memory (low byte)
R16	light pen position detected (high byte)
R17	light pen position deteced (low byte)

VDU 80 - PARTS LIST

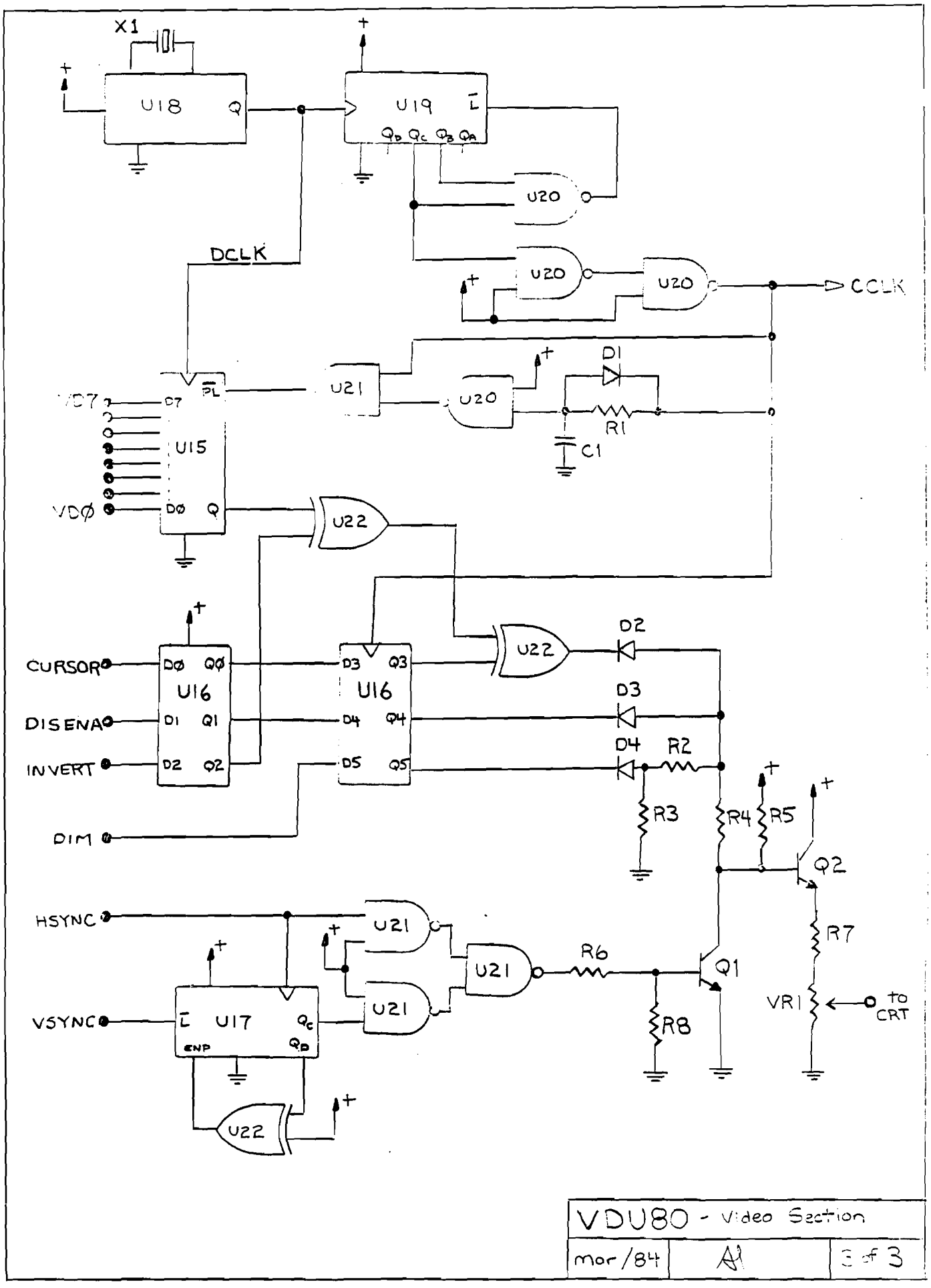
PART #	TYPE	DESCRIPTION
U1	4011	CMOS Quad NAND
U2	4001	CMOS Quad NOR
U3	4068	CMOS 8 input NAND
U4	4027	CMOS Dual D-type flip-flop
U5	4042	CMOS Quad D-type latch
U6	MC6845	Motorola Video Display Controller
U7,U8	CD1856	RCA Bi-directional buss buffer
U9,U10,U11	74HC157	High Speed CMOS 2 to 1 Multiplexor
K12	2016	2K x 8 NMOS RAM (video memory)
U13	74LS373	lsTTL 8 BIT latch
U14	2716	2k x 8 EPROM (character generator)
U15	74LS165	8 bit shift register - parallel load
U16	74LS174	lsTTL Hex Latch
U17,U19	74LS163	lsTTL Counter
U18	74S124	sTTL VCO
U20,U21	74LS00	lsTTL Quad QAND
U22	74LS86	lsTTL Quad Exclusive Or
Q1,Q2	2N2222	Small signal high frequency transistor
R1	4K7	Resistor
R2	680	"
R3	1K5	"
R4	150	"
R5	470	"
R6	1K0	"
R7	50	"
R8	330	"
VR1	50	Variable resistor (brightness adjust)
C1	22pf	Capacitor
C2-C10	.1uf	Capacitors (distributed around the board)
D1,D2	1N914	Small signal high frequency signal diode
D3,D4	"	


TPA
U5
U3
U1
D3
D2
D1
D0
Q3
Q2
Q1
Q0
A7
AØ
A1Ø'
AØ'
mWR
U2
OE
TPB
U1
WE
U2
mUX
mRD
U2
U2
D7
DØ
CS
D3
D2
D1
DØ
U8
O3
I3
O2
I2
O1
I1
OØ
IØ
mRD
D7'
DØ'
CLK
U1
CS
D3
D2
D1
DØ
U7
O3
I3
O2
I2
O1
I1
OØ
IØ
mRD
TPA
J
Q
U4
K
R
Q̄
J
Q
U4
K
R
Q̄
TPB
E
VDU8Ø - CPU Interface
Mar/84
Al
1 of 3



VDUBØ - Data Section

Mar/84 | A1 | 1 of 3



VDU80 - Video Section		
mar/84	A1	3 of 3

CHIP 8 AE - T.V. TYPEWRITER

by M.E. Franklin, 690 Laurier Ave., Milton, Ont. Canada, L9T 4R5

the following CHIP 8 program is based upon an article appearing in Quest data Vol. 2, Issue 10, by David Crawford, entitled TVT -4K . David's article contained the character table and driver routines that became the basis of the built in character table and the FX29, FX33 and FX95 commands in CHIP 8 AE.

At the time I wrote this article, I was looking for a large format character set which would be easy for my young children to read. The T.V. Typewriter outputs 6 lines of 20 characters on each of 2 pages. (assuming you are using the ACE VDU board and low RES mode).

Cursor commands are as follows:

ASCII	Symbol	Function
5D	]	cursor right
5B	[	cursor left
7B		cursor up
7D		cursor down
0D	CR	new line
7F	DEL	delete char.

Display commands are as follows:

	Control	
01	A	erase screen erase all Mem. home cursor
02	B	delete home to cursor
03	C	delete cursor to end of page
04	D	display page 1
05	E	display page 2
06	F	overlay pages
07	G	TAB to centre
10	P	clear screen
11	Q	game at 0400h
12	R	message at 03C0

The control Q and R commands are useful for messages or games to be called up from memory. The program, however, makes no provision for saving or loading games, other than through the system monitor.

A use for the overlay command is in math tests or quizzes. Place the questions on the 6 lines on page 1 in spaces 1 to 10. Answers would be places in spaces 11 to 14 by use of the TAB command. Place the correct answers on page 2 in positions 16 to 20, at a previous time of course, and, after the questions have been answered, overlay page 2 on top of page 1 to check the results. This game can be fun and educational!

```
0200  2354  CALL ILS AT 0354
0202  6C00  SET VC EQ 00
0204  FC0E  SET VC EQ ASCII
0206  4C00  SKIP IF VC NEQ 00
0208  1204  GOTO 0204
020A  220E  CALL ILS AT 020E
020C  1202  GOTO 0202
020E  4C5D  SKIP IF VC NEQ 5D
0210  1300  GOTO 0300
0212  4C5B  SKIP IF VC NEQ 5B
0214  130C  GOTO 030C
0216  4C7B  SKIP IF VC NEQ 7B
0218  1318  GOTO 0318
021A  4C7D  SKIP IF VC NEQ 7D
021C  1324  GOTO 0324
021E  4C0D  SKIP IF VC NEQ 0D
0220  1296  GOTO 0296
0222  4C7F  SKIP IF VC NEQ 7F
0224  1330  GOTO 0330
0226  4C01  SKIP IF VC NEQ 01
0228  1354  GOTO 0354
022A  4C02  SKIP IF VC NEQ 02
022C  136A  GOTO 036A
022E  4C03  SKIP IF VC NEQ 03
0230  138C  GOTO 038C
0232  4C04  SKIP IF VC NEQ 04
0234  12BA  GOTO 02BA
0236  4C05  SKIP IF VC NEQ 05
0238  12F0  GOTO 02F0
023A  4C06  SKIP IF VC NEQ 06
023C  12F8  GOTO 02F8
023E  4C07  SKIP IF VC NEQ 07
0240  139E  GOTO 039E
0242  4C10  SKIP IF VC NEQ 10
0244  1360  GOTO 0360
0246  4C11  SKIP IF VC NEQ 11
0248  1400  GOTO 0400
024A  4C12  SKIP IF VC NEQ 12
024C  13C0  GOTO 03C0
024E  4A7C  SKIP IF VA NEQ 7C
0250  2296  CALL ILS AT 0296
0252  4D7D  SKIP IF VD NEQ 7D
0254  22B0  CALL ILS AT 02B0
0256  4DFC  SKIP IF VD NEQ FC
0258  23A8  CALL ILS AT 03A8
025A  226C  CALL ILS AT 026C
025C  2290  CALL ILS AT 0290
025E  F995  POINT I TO V9 FOR ASCII
0260  DAB7  SHOW 7 AT VA,VB
0262  FC95  POINT I TO VC FOR ASCII
0264  DAB7  SHOW 7 AT VA,VB
0266  7A06  SET VA EQ VA + 06
0268  2290  CALL ILS AT 0290
026A  1202  GOTO 0202
026C  2278  CALL ILS AT 0278
026E  8900  SET V9 EQ V0
0270  80C0  SET V0 EQ VC
0272  2280  CALL ILS AT 0280
0274  7D01  SET VD EQ VD + 01
0276  00EE  ILS RETURN
0278  A500  POINT I AT 0500
027A  FD1E  SET I EQ I + VD
027C  F065  GET V0 TO V0 FROM I
027E  00EE  ILS RETURN
0280  A500  POINT I AT 0500
0282  FD1E  SET I EQ I + VD
0284  F055  SAVE V0 TO V0 AT I
0286  00EE  ILS RETURN
0288  0000  ?
028A  0000  ?
028C  0000  ?
028E  00FC  ?
0290  A288  POINT I AT 0288
0292  DAB8  SHOW 8 AT VA,VB
0294  00EE  ILS RETURN
0296  2290  CALL ILS AT 0290
0298  7A06  SET VA EQ VA + 06
029A  7D01  SET VD EQ VD + 01
029C  3A7C  SKIP IF VA EQ 7C
029E  1298  GOTO 0298
02A0  600D  SET V0 EQ 0D
02A2  2280  CALL ILS AT 0280
02A4  6A04  SET VA EQ 04
02A6  7B09  SET VB EQ VB + 09
02A8  4B3A  SKIP IF VB NEQ 3A
02AA  12B0  GOTO 02B0
02AC  2290  CALL ILS AT 0290
02AE  1274  GOTO 0274
02B0  00E0  CLEAR SCREEN
02B2  6B04  SET VB EQ 04
02B4  6D80  SET VD EQ 80
02B6  2290  CALL ILS AT 0290
02B8  00EE  ILS RETURN
02BA  6D00  SET VD EQ 00
02BC  00E0  CLEAR SCREEN
02BE  6A04  SET VA EQ 04
02C0  6B04  SET VB EQ 04
```

```
02C2  2278  CALL ILS AT 0278
02C4  4000  SKIP IF V0 NEQ 00
02C6  12D4  GOTO 02D4
02C8  400D  SKIP IF V0 NEQ 0D
02CA  12DE  GOTO 02DE
02CC  4020  SKIP IF V0 NEQ 20
02CE  12D4  GOTO 02D4
02D0  F095  POINT I TO V0 FOR ASCII
02D2  DAB7  SHOW 7 AT VA,VB
02D4  7A06  SET VA EQ VA + 06
02D6  7D01  SET VD EQ VD + 01
02D8  3A7C  SKIP IF VA EQ 7C
02DA  12C2  GOTO 02C2
02DC  7D01  SET VD EQ VD + 01
02DE  6A04  SET VA EQ 04
02E0  4B31  SKIP IF VB NEQ 31
02E2  12E8  GOTO 02E8
02E4  7B09  SET VB EQ VB + 09
02E6  12C2  GOTO 02C2
02E8  6A04  SET VA EQ 04
02EA  6B04  SET VB EQ 04
02EC  6D00  SET VD EQ 00
02EE  1290  GOTO 0290
02F0  6D80  SET VD EQ 80
02F2  22BC  CALL ILS AT 02BC
02F4  6D80  SET VD EQ 80
02F6  00EE  ILS RETURN
02F8  22BA  CALL ILS AT 02BA
02FA  6D80  SET VD EQ 80
02FC  2290  CALL ILS AT 0290
02FE  12BE  GOTO 02BE
0300  4A76  SKIP IF VA NEQ 76
0302  00EE  ILS RETURN
0304  2290  CALL ILS AT 0290
0306  7A06  SET VA EQ VA + 06
0308  7D01  SET VD EQ VD + 01
030A  1290  GOTO 0290
030C  4A04  SKIP IF VA NEQ 04
030E  00EE  ILS RETURN
0310  2290  CALL ILS AT 0290
0312  7AFA  SET VA EQ VA + FA
0314  7DFF  SET VD EQ VD + FF
0316  1290  GOTO 0290
0318  4B04  SKIP IF VB NEQ 04
031A  00EE  ILS RETURN
031C  2290  CALL ILS AT 0290
031E  7BF7  SET VB EQ VB + F7
0320  7DEB  SET VD EQ VD + EB
0322  1290  GOTO 0290
0324  4B32  SKIP IF VB NEQ 32
0326  00EE  ILS RETURN
0328  2290  CALL ILS AT 0290
032A  7B09  SET VB EQ VB + 09
032C  7D15  SET VD EQ VD + 15
032E  1290  GOTO 0290
0330  3D00  SKIP IF VD EQ 00
0332  4D80  SKIP IF VD NEQ 80
0334  00EE  ILS RETURN
0336  2290  CALL ILS AT 0290
0338  7DFF  SET VD EQ VD + FF
033A  6C00  SET VC EQ 00
033C  226C  CALL ILS AT 026C
033E  4A04  SKIP IF VA NEQ 04
0340  234E  CALL ILS AT 034E
0342  7AFA  SET VA EQ VA + FA
0344  F995  POINT I TO V9 FOR ASCII
0346  390D  SKIP IF V9 EQ 0D
0348  DAB7  SHOW 7 AT VA,VB
034A  7DFF  SET VD EQ VD + FF
034C  1290  GOTO 0290
034E  7BF7  SET VB EQ VB + F7
0350  6A7C  SET VA EQ 7C
0352  134C  GOTO 034C
0354  A500  POINT I AT 0500
0356  6000  SET V0 EQ 00
0358  F055  SAVE V0 TO V0 AT I
035A  7D01  SET VD EQ VD + 01
035C  3D00  SKIP IF VD EQ 00
035E  1356  GOTO 0356
0360  00E0  CLEAR SCREEN
0362  6A04  SET VA EQ 04
0364  6B04  SET VB EQ 04
0366  6D00  SET VD EQ 00
0368  1290  GOTO 0290
036A  8ED0  SET VE EQ VD
036C  6D80  SET VD EQ 80
036E  8DE5  SET VD EQ VD - VE
0370  6D00  SET VD EQ 00
0372  4F00  SKIP IF VF NEQ 00
0374  1378  GOTO 0378
0376  6D80  SET VD EQ 80
0378  6000  SET V0 EQ 00
037A  2280  CALL ILS AT 0280
037C  7D01  SET VD EQ VD + 01
037E  5ED0  SKIP IF VE EQ VD
0380  1378  GOTO 0378
0382  6D80  SET VD EQ 80
```

```
0384  8ED5  SET VE EQ VE - VD
0386  3F00  SKIP IF VF EQ 00
0388  12F0  GOTO 02F0
038A  12BA  GOTO 02BA
038C  8ED0  SET VE EQ VD
038E  4D7F  SKIP IF VD NEQ 7F
0390  1382  GOTO 0382
0392  4DFF  SKIP IF VD NEQ FF
0394  1382  GOTO 0382
0396  6000  SET V0 EQ 00
0398  2280  CALL ILS AT 0280
039A  7D01  SET VD EQ VD + 01
039C  138E  GOTO 038E
039E  2290  CALL ILS AT 0290
03A0  6E40  SET VE EQ 40
03A2  89A0  SET V9 EQ VA
03A4  89E5  SET V9 EQ V9 - VE
03A6  3F00  SKIP IF VF EQ 00
03A8  13B6  GOTO 03B6
03AA  7D01  SET VD EQ VD + 01
03AC  7A06  SET VA EQ VA + 06
03AE  5AE0  SKIP IF VA EQ VE
03B0  13AA  GOTO 03AA
03B2  6A40  SET VA EQ 40
03B4  1290  GOTO 0290
03B6  7DFF  SET VD EQ VD + FF
03B8  7AFA  SET VA EQ VA + FA
03BA  5AE0  SKIP IF VA EQ VE
03BC  13B6  GOTO 03B6
03BE  13B0  GOTO 03B0
03C0  2290  CALL ILS AT 0290
03C2  6A1C  SET VA EQ 1C
03C4  6B16  SET VB EQ 16
03C6  6C48  SET VC EQ 48
03C8  23F8  CALL ILS AT 03F8
03CA  6C45  SET VC EQ 45
03CC  23F8  CALL ILS AT 03F8
03CE  6C4C  SET VC EQ 4C
03D0  23F8  CALL ILS AT 03F8
03D2  6C4C  SET VC EQ 4C
03D4  23F8  CALL ILS AT 03F8
03D6  6C4F  SET VC EQ 4F
03D8  23F8  CALL ILS AT 03F8
03DA  6C20  SET VC EQ 20
03DC  23F8  CALL ILS AT 03F8
03DE  6C4C  SET VC EQ 4C
03E0  23F8  CALL ILS AT 03F8
03E2  6C41  SET VC EQ 41
03E4  23F8  CALL ILS AT 03F8
03E6  6C55  SET VC EQ 55
03E8  23F8  CALL ILS AT 03F8
03EA  6C52  SET VC EQ 52
03EC  23F8  CALL ILS AT 03F8
03EE  6C41  SET VC EQ 41
03F0  23F8  CALL ILS AT 03F8
03F2  6C21  SET VC EQ 21
03F4  23F8  CALL ILS AT 03F8
03F6  1362  GOTO 0362
03F8  FC95  POINT I TO VC FOR ASCII
03FA  DAB7  SHOW 7 AT VA,VB
03FC  7A06  SET VA EQ VA + 06
```

1802/SPO256-AL2 TALKING CLOCK

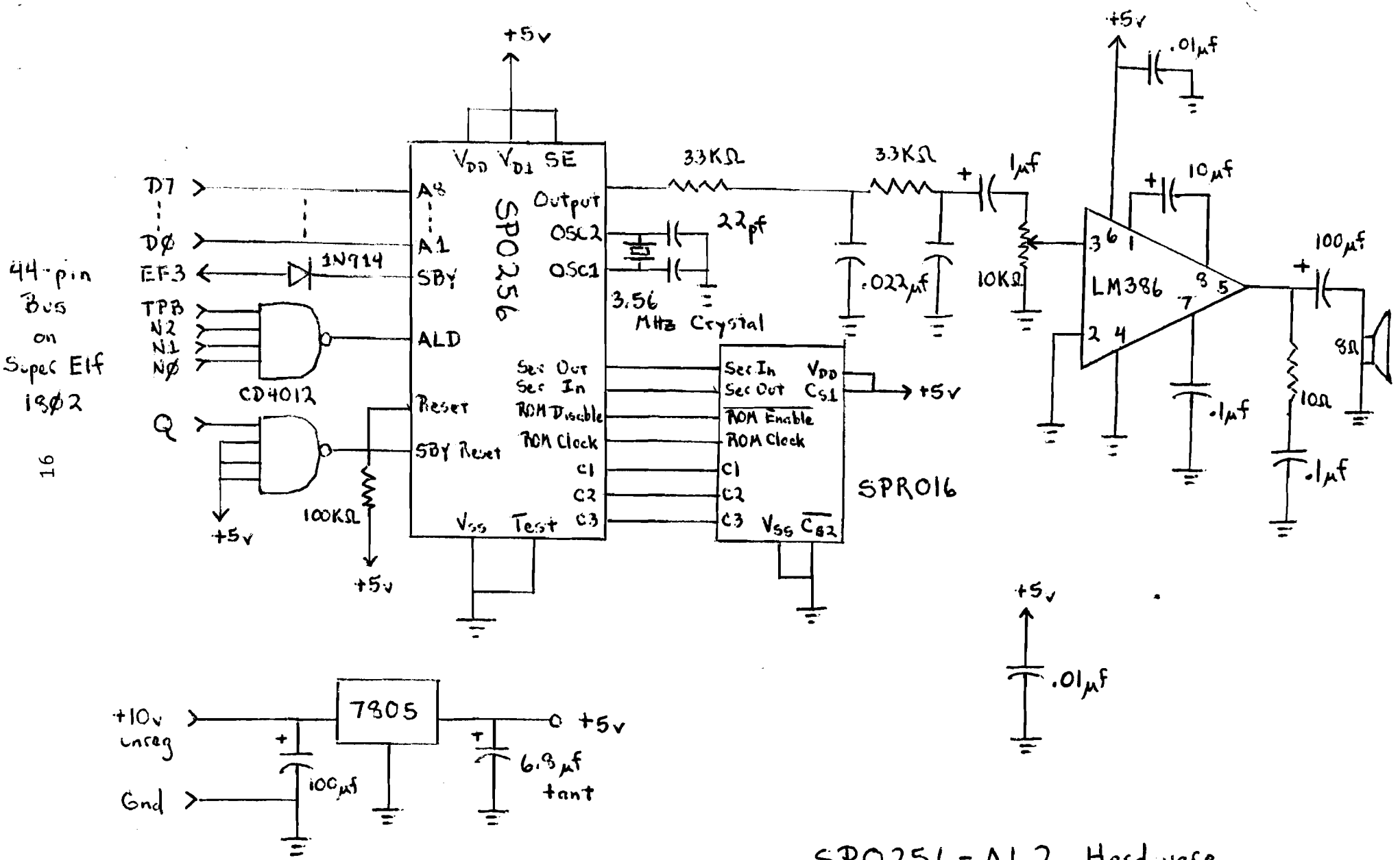
by P.B. Liescheski III, Dept. of Chemistry, U of Texas, Austin TX

After purchasing the SPO256-AL2 voice synthesizer chip set (catalog #: 276-1783) from Radio Shack, it was realized that it is a "canned-styled" voice synthesizer programmed especially for a clock. With this, it was decided to interface this synthesizer with the 1802 microprocessor and program it to be a talking clock. As a result, the program, TALKING-CLOCK, was written.

TALKING-CLOCK is easy to use. First one must obtain the SPO256-AL2 voice synthesizer chip set. The hardware is built according to the schematic. The design is basically identical to the schematic given in the datasheet which accompanies the chips. The major differences are the details on interfacing with the 1802, the use of a color-burst crystal, and the reset circuit. The SPO256 must be reset frequently. It has been discovered that resetting before using works fine. The circuit is built on a VECTOR #0028 3677-2DP D.I.P. Plugbord, since it easily plugs into the Quest Super Elf 44-pin connector.

After the hardware, TALKING-CLOCK is entered into memory. The initial hour minus one is entered into location 0002, while the value for the minutes is entered into location 0005. So if one should start the clock at 2:32, one enters #01 into location 0002 and #20 into location 0005. With this, execute the program with R3 as program counter starting at location 0000. Upon pressing the input (I) key, one will hopefully hear: " It is two thirty two ", or whatever time it may be.

Hopefully there are enough comment statements in the program so that someone can understand what is going on in the program. First the major registers are defined. R2 is the stack pointer which begins at location 00FF. R7.1 is the hour counter and R7.0 is the minute counter. R4 is the program counter for the subroutine UTTER. Next the program has a triple-nested loop called TIMELOOP which requires exactly (well close enough) one minute to execute. TIMELOOP is designed with a 1.79 MHz clock frequency in mind. If this is not the case, then the value of TIME1 can be adjusted so that TIMELOOP consumes one minute. In TIMELOOP, the program checks whether the input (I) key is pressed for time. After TIMELOOP has finished its one minute task. CLOCK is entered. CLOCK bumps the minute counter R7.0. It maintains R7.0 as a sixty counter and R7.1 as a twelve counter. If it is a new hour, CLOCK increments R7.1 and goes to CHIMES. After this, it returns to TIMELOOP for another minute. If the Input (I) key is pressed, the processor goes to TIMETALK. TIMETALK takes the information in R7 and tells the time. The subroutine UTTER drives the voice synthesizer. The hardware is designed so that setting the Q output line will reset the circuit. According to the datasheet, Q must be high for at least 100 microseconds or reset will not be performed. The voice synthesizer is wired to the parallel output port #7. Hopefully these short notes will give the essence of TALKING-CLOCK.



SP0256-AL2 Hardware

```
802 CROSS ASSEMBLER V1.0 - SECOND PASS    DATE  24 JUN 84
                    *
                    *  TALKING-CLOCK
                    *    DRIVER FOR SP0256-AL2 VOICE SYNTHESIZER
                    *
                    *  PHILLIP B. LIESCHESKI III ** 6-14-84
                    *
                              ORG       @0000
                    *
                    HOUR      EQU       #00        SET AT 1:00 O'CLOCK
                    MINUTE    EQU       #00
                    STACKTOP  EQU       #FF
                    PAGE      EQU       #00
                    TIME1     EQU       #23
                    TIME2     EQU       #4A
                    HEX60     EQU       #3C
                    HEX12     EQU       #0C
                    TUNE      EQU       #21
                    ITIS      EQU       #18
                    ONE       EQU       #01
                    ZERO      EQU       #00
                    *
                    *  SET-UP ROUTINE FOR MAJOR REGISTERS
                    *
0000:     E2        INTRO     SEX       R2         R2 IS THE STACK POINTER
0001:     F800                LDI       HOUR       R7 IS THE TIME COUNTER
0003:     B7                  PHI       R7            R7.1 = HOURS
0004:     F800                LDI       MINUTE
0006:     A7                  PLO       R7            R7.0 = MINUTES
0007:     F800                LDI       PAGE
0009:     B2                  PHI       R2
000A:     B4                  PHI       R4         R4 IS THE 'UTTER' SUBROUTINE POINTER
000B:     F8FF                LDI       STACKTOP
000D:     A2                  PLO       R2
000E:     F882                LDI       UTTER
0010:     A4                  PLO       R4
                    *
                    *  A ONE MINUTE TIMER
                    *
0011:     F823      TIMELOOP  LDI       TIME1      A TRIPLE-NESTED LOOP WHICH LOOPS
0013:     AA                  PLO       RA            FOR ONE MINUTE (CLOCK=1.79 MHZ)
0014:     F84A                LDI       TIME2
0016:     AB                  PLO       RB
0017:     AC                  PLO       RC
0018:     C4        DELAY     NOP                  MAJOR TIME KILLER
0019:     C4                  NOP
001A:     C4                  NOP
001B:     C4                  NOP
001C:     C4                  NOP
001D:     C4                  NOP
001E:     C4                  NOP
001F:     C4                  NOP
0020:     C4                  NOP
0021:     C4                  NOP
0022:     C4                  NOP
0023:     C4                  NOP
0024:     C4                  NOP
0025:     C4                  NOP
0026:     C4                  NOP
0027:     C4                  NOP
0028:     C4                  NOP
```

```
0029:   C4                  NOP
002A:   C4                  NOP
002B:   C4                  NOP
002C:   3755                B4        TIMETALK  CHECK I KEY, SOMEONE MAY
002E:   2A        REENTRY   DEC       RA           WANT THE TIME
002F:   8A                  GLO       RA
0030:   3A18                BNZ       DELAY
0032:   F823                LDI       TIME1
0034:   AA                  PLO       RA
0035:   2B                  DEC       RB
0036:   8B                  GLO       RB
0037:   3A18                BNZ       DELAY
0039:   F84A                LDI       TIME2
003B:   AB                  PLO       RB
003C:   2C                  DEC       RC
003D:   8C                  GLO       RC
003E:   3A18                BNZ       DELAY     IF NOT FINISHED, KEEP LOOPING
                  *
                  *  THE KEEPER OF HOURS & MINUTES
                  *
0040:   17        CLOCK     INC       R7        BUMP MINUTE COUNTER
0041:   87                  GLO       R7
0042:   FF3C                SMI       HEX60     CHECK IF ON THE START OF A NEW HOUR
0044:   3A11                BNZ       TIMELOOP  RETURN IF NOT, ELSE CONTINUE
0046:   A7                  PLO       R7
0047:   97                  GHI       R7
0048:   FC01                ADI       ONE       BUMP HOUR COUNTER
004A:   B7                  PHI       R7        RESET IF 12TH HOUR
004B:   FF0C                SMI       HEX12     SINCE IT IS ON THE HOUR,
004D:   3A50                BNZ       CHIMES       GO TO CHIMES
004F:   B7                  PHI       R7
                  *
                  *  THE HOURLY CHIMES
                  *
0050:   F821      CHIMES    LDI       TUNE      ON THE HOUR CHIMES
0052:   D4                  SEP       R4           FOR THE ROMANTIC TOUCH
0053:   3011                BR        TIMELOOP
                  *
                  *  THE SPEAKER OF HOURS & MINUTES
                  *
0055:   F818      TIMETALK  LDI       ITIS
0057:   D4                  SEP       R4        SAY "IT IS"
0058:   97                  GHI       R7        PREP HOUR COUNTER FOR SPEECH
0059:   FA0F                ANI       #0F
005B:   FC01                ADI       ONE       ADD ONE TO HOURS SINCE 1 0"CLOCK=0
005D:   D4                  SEP       R4        SAY THE HOUR
005E:   87                  GLO       R7
005F:   322E                BZ        REENTRY   RETURN TO LOOP, IF ZERO MINUTES
0061:   F800                LDI       ZERO
0063:   A8                  PLO       R8
0064:   87                  GLO       R7        DIVIDE MINUTES BY TEN
0065:   B8        TT1       PHI       R8          AND PREP MINUTE CNTR FOR SPEECH
0066:   FF0A                SMI       #0A
0068:   3B6D                BM        TT2
006A:   18                  INC       R8
006B:   3065                BR        TT1
006D:   88        TT2       GLO       R8        SAY THE MINUTES
006E:   FF01                SMI       ONE
0070:   3A76                BNZ       TT3
0072:   87                  GLO       R7
```

```
0073:   D4                SEP     R4
0074:   302E              BR      REENTRY
0076:   88      TT3       GLO     R8
0077:   327B              BZ      TT4
0079:   FC12              ADI     #12
007B:   D4      TT4       SEP     R4
007C:   98                GHI     R8
007D:   322E              BZ      REENTRY
007F:   D4                SEP     R4
0080:   302E              BR      REENTRY   RETURN TO LOOP
                *
                *  THE SP0256-AL2 SUBROUTINE DRIVER
                *
0082:   52      UTTER     STR     R2        PUT D ON THE STACK
0083:   3683    WAIT      B3      WAIT      CHECK IF VOICE SYNTH IS BUSY
0085:   7B                SEQ               RESET VOICE SYNTH
0086:   C4                NOP               WAIT A FEW MICROSECONDS
0087:   C4                NOP
0088:   C4                NOP
0089:   C4                NOP
008A:   C4                NOP
008B:   C4                NOP
008C:   C4                NOP
008D:   C4                NOP
008E:   C4                NOP
008F:   C4                NOP
0090:   7A                REQ
0091:   67                OUT7              OUTPUT TO VOICE SYNTH
0092:   22                DEC     R2        GET STACK BACK AS IT WAS
0093:   D3                SEP     R3        RETURN
0094:   3082              BR      UTTER
                *
                *  THE END
                *
                          END
```

An 1802 - Microsoft Basic Cross Assembler

by J. G. Cayer, POB 2034, Hinton, Alberta, Canada, T0E 1C0

Enclosed a listing of an 1802 cross-assembler written on my TRS-80 Model III, and which may be of interest to those members who have access to Microsoft Basic computers.

On running the program, you are asked for the ORIGIN. Input a four character HEX number - 0000 to FFFF - and press <ENTER>. The screen will clear and at the bottom of the screen will appear:

```
ADDR    CODE    LABEL   MNEM    OPER    COMMENTS
-------------------------------------------------------------
XXXX
```

The cursor will now flash under 'L' of LABEL. This column is used strictly for labels up to a length of 6 characters. If a label is not wanted in that line, move on to MNEM by pressing the right-arrow key. The cursor will now flash beneath 'M' of MNEM. Enter a mnemonic and then move on to the OPER column by again depressing the right-arrow key,if an operand is required. If an operand is not required, you may move on to the COMMENTS column. In any case, after the mnemonic has been entered and no more info is needed, press <ENTER>. This is also true for all columns after MNEM. On pressing <ENTER>, the input line will scroll and the appropriate code will be PRINTED @ the proper location under the CODE column.

The MNEM column also serves to enter a constant in HEX at the address listed on that line. This is entered as #XX and the one byte constant you wrote will appear in the code column. I have also added ten control commands which when used, are entered under MNEM. These I will discuss later.

In the OPER column, you may enter hex or decimal numbers. Hex numbers take the form #XX for single byte or #XXXX for two byte numbers. Thus #10 or 16 will give the same result. If the operand designates a register, just enter the registers number (0-F) and not R0-RF, otherwise the program will see R- as a label. Labels of two to six characters can also be entered here, and their addresses will be posted on pressing <ENTER> providing they have been entered previously. Otherwise, the address will appear as XX-- if that label is still to come. If in using short branch instructions, you type '<' in the operand column, the code will branch to itself.

The COMMENTS column can handle long comments but short comments look better when the assembled program is listed.

As mentioned earlier, control commands are ALL entered in the MNEM column as follows:

LIST - will list all addresses, code, mnemonics, operands & comments. This is handy when you want to get rid of the editing garbage.

COPY - will output to printer.

DELETE - enter the hex line number in the OPER column (no # needed before hex number). The line which matches that number will be scratched and all line numbers thereafter will be updated.

INSERT - enter hex number as above. Number will appear at bottom of screen. Enter code as usual. Code will be inserted and all numbers updated.

CHANGE - same as INSERT except make sure code length does not change. This routine does not update the numbers. If code length must change, then DELETE & INSERT.

HEXA - converts decimal value to hex. Enter decimal value in OPER.column.

DECI - converts hex value to decimal. Enter hex value in OPER column.

SAVE - will save to disk. Routine will ask for FILENAME and save.

LOAD - loads previously saved code. Programmer can carry on where he left off.

ASSEMB - assembles code and processes all labels.

All control commands can be used at will. The program always return to the input routine.

Short demo:

```
0000    90      BEGIN   GHI     0       DEMO
0001    B1              PHI     1
0002    A1              PLO     1
0003    21      LOOP    DEC     1
0004    81              GLO     1
0005    32--            BZ      END
0007    F800            LDI     #00     COULD BE JUST 0
0009    55              STR     5
000A    3003            BR      LOOP
000C    55      END     STR     5
```

```
0000    90      BEGIN   GHI     0       DEMO
0001    B1              PHI     1
0002    A1              PLO     1
0003    21      LOOP    DEC     1
0004    81              GLO     1
0005    320C            BZ      END
0007    F800            LDI     #00     COULD BE JUST 0
0009    55              STR     5
000A    3003            BR      LOOP
000C    55      END     STR     5
```

```
0 GOTO 1000
1 REM ***CHECK IF HEX ROUTINE***
2 IF (Z>47 AND Z<58) OR (Z>64 AND Z<71) THEN RETURN ELSE FL=1: RETURN
3 REM ***DEC TO HEX ROUTINE***
4 Z$="": F=4096: FOR I=1 TO 4: E1=INT(E/F): E=E-E1*F: Z$=Z$+MID$(G$,E1+1,1): F
/16: NEXT: RETURN
5 REM ***LOOK BACK IF WE HAVE LABEL ADDRESS***
6 FOR K=1 TO T: IF MID$(DU$,17,6)=LA$(K,1) THEN RETURN ELSE NEXT
8 FL=1: RETURN
9 REM ***FIND ALL POSSIBLE ADDRESSES AND LIST***
10 CLS: FOR K=1 TO TT: IF LEN(S$(K))<33 THEN 18 ELSE IF ASC(MID$(S$(K),33,1))>64
 AND ASC(MID$(S$(K),33,1))<91 AND ASC(MID$(S$(K),34,1))<>32 THEN 12 ELSE 18
12 FOR I=1 TO TT: IF MID$(S$(K),33,6)<>MID$(S$(I),17,6) THEN 16
13 IF MID$(S$(K),13,2)<>"  " THEN S$(K)=LEFT$(S$(K),10)+LEFT$(S$(I),4)+RIGHT$(S$
(K),LEN(S$(K))-14): GOTO16
14 S$(K)=LEFT$(S$(K),10)+MID$(S$(I),3,2)+RIGHT$(S$(K),LEN(S$(K))-12)
16 NEXT I
18 PRINT S$(K): NEXT K: RETURN
20 REM ***LIST ROUTINE***
22 CLS: FOR K=1 TO TT: PRINT S$(K): NEXT: RETURN
29 REM ***HEX TO DEC ROUTINE***
30 U=0: FOR I=1 TO LEN(E$): B=ASC(MID$(E$,I,1))-48: IF B>9 THEN B=B-7
32 U=U*16+B: NEXT: RETURN
40 REM ***CHECK IF WE HAVE A CONSTANT INSTEAD OF MNEM***
42 IF MID$(DU$,9,1)="#" THEN Z=ASC(MID$(DU$,10,1)): GOSUB 2: Z=ASC(MID$(DU$,11,1
)): GOSUB 2: IF FL=0 THEN OP$=MID$(DU$,10,2): PRINT @ 904, OP$: AD=1: RETURN ELS
E FL=0: PRINT @ 904, "ERROR": AD=0: RETURN
44 RESTORE
46 DATA ADC,74,ADD,F4,AND,F2,DIS,71,IDL,00,IRX,60,LDX,F0,LDXA,72,LSDF,CF,LSIE,CC
,LSKP,C8,LSNF,C7,LSNQ,C5,LSNZ,C6,LSQ,CD,LSZ,CE,MARK,79,NOP,C4,OR,F1,REQ,7A,RE[illegible]7
0,SAV,78,SD,F5,SDB,75,SEQ,7B,SHL,FE,SHLC,7E,SHR,F6,SHRC,76,SKP,38,SM,F7,SMB,,
48 DATA STXD,73,XOR,F3
50 REM ***SINGLE BYTE NON-REG INSTR***
52 FOR K=1 TO 34: READ M$,OP$: IF MN$=M$ THEN PRINT @ 904, OP$ ELSE NEXT K
54 IF K=35 THEN 60 ELSE AD=1: RETURN
56 DATA INP,6,OUT,6,DEC,2,GHI,9,GLO,8,INC,1,LDA,4,LDN,0,PHI,B,PLO,A,SEP,D,SEX,E,
STR,5
58 REM ***REGISTER,INP & OUT INSTR***
60 FOR K=1 TO 13: READ M$,OP$: IF MN$<>M$ THEN NEXT: GOTO 76
61 W=ASC(MID$(DU$,17,1))
62 IF M$="OUT" AND (W>48 AND W<56) THEN OP$=OP$+CHR$(W): PRINT @ 904, OP$: AD=1:
 RETURN
64 IF M$="INP" AND W=49 THEN OP$=OP$+"9": PRINT @ 904, OP$: AD=1: RETURN
66 IF M$="INP" AND W>49 AND W<56 THEN OP$=OP$+CHR$(W+15): PRINT @ 904, OP$: AD=1
: RETURN
67 IF M$="OUT" OR M$="INP" THEN 70
68 IF (W>47 AND W<58) OR (W>64 AND W<71)THEN OP$=OP$+CHR$(W): PRINT @ 904, OP$:
AD=1: RETURN
70 PRINT @ 904,"ERROR": AD=0: RETURN
72 DATA ADCI,7C,ADI,FC,ANI,FA,LDI,F8,ORI,F9,SDBI,7D,SDI,FD,SMBI,7F,SMI,FF,XRI,FB
74 REM ***IMMEDIATE INSTR****
76 FOR K=1 TO 10: READ M$,OP$: IF MN$<>M$ THEN NEXT: GOTO 88
78 W=ASC(MID$(DU$,17,1)): IF W>64 AND W<91 THEN GOSUB 6: IF FL=0 THEN OP$=OP$+RI
GHT$(LA$(K,2),2): PRINT @ 904, OP$: AD=2: RETURN: ELSE IF FL=1 THEN FL=0: OP$=OP
$+"--": PRINT @ 904, OP$: AD=2: RETURN
80 IF W=35 THEN Z=ASC(MID$(DU$,18,1)): GOSUB 2: Z=ASC(MID$(DU$,19,1)): GOSUB2: I
F FL=0 THEN OP$=OP$+MID$(DU$,18,2): PRINT @ 904, OP$: AD=2: RETURN ELSE IF F
THEN FL=0: PRINT @ 904,"ERROR": AD=0: RETURN
```

```
82 E=VAL(MID$(DU$,17,3)): IF E>255 THEN PRINT @ 904,"ERROR": AD=0: RETURN ELSE G
OSUB 4: OP$=OP$+RIGHT$(Z$,2): PRINT @ 904, OP$: AD=2: RETURN
84 DATA B1,34,B2,35,B3,36,B4,37,BDF,33,BN1,3C,BN2,3D,BN3,3E,BN4,3F,BNF,38,BNQ,39
,BNZ,3A,BQ,31,BR,30,BZ,32
86 REM ***BRANCH INSTR***
88 FOR K=1 TO15: READ M$,OP$: IF MN$<>M$ THEN NEXT: GOTO 96
89 IF ASC(MID$(DU$,17,1))=60 THEN OP$=OP$+RIGHT$(O$,2): PRINT @ 904, OP$: AD=2:
RETURN
90 GOTO 78: REM ***SAME AS IMM. INSTR***
92 DATA LBDF,C3,LBNF,CB,LBNQ,C9,LBNZ,CA,LBQ,C1,LBR,C0,LBZ,C2
94 REM ***LONG BRANCH INSTR***
96 FOR K=1 TO 7: READ M$,OP$: IF MN$<>M$ THEN NEXT: GOTO 105
98 W=ASC(MID$(DU$,17,1)): IF W>64 AND W<91 THEN GOSUB 6: IF FL=0 THEN OP$=OP$+LA
$(K,2): PRINT @ 904, OP$: AD=3: RETURN: ELSE IF FL=1 THEN FL=0: OP$=OP$+"----":
PRINT @ 904, OP$: AD=3: RETURN
100 IF W=35 THEN FOR K=18 TO 21: Z=ASC(MID$(DU$,K,1)): GOSUB 2: NEXT K: IF FL=0
THEN OP$=OP$+MID$(DU$,18,4): PRINT @ 904, OP$: AD=3: RETURN ELSE IF FL=1 THEN FL
=0: PRINT @ 904,"ERROR": AD=0: RETURN
102 E=VAL(MID$(DU$,17,5)): IF E>65535 THEN PRINT @ 904,"ERROR": AD=0: RETURN ELS
E GOSUB 4: OP$=OP$+Z$: PRINT @ 904, OP$: AD=3: RETURN
103 DATA LIST,1,COPY,2,DELETE,3,INSERT,4,SAVE,5,LOAD,6,HEXA,7,DECI,8,ASSEMB,9,CH
ANGE,10
105 FOR K=1 TO 10: READ M$,G: IF MN$<>M$ THEN NEXT: GOTO 113
107 ON G GOSUB 20,2500,2000,4000,3000,3500,5000,5500,10,6000
113 AD=0:RETURN
143 REM ***INPUT CODE***
153 PRINT @ 960, O$;: PRINT @ 976,"";: LINEINPUT DU$: DU$=DU$+"    "
163 REM ***CHECK IF WE HAVE A LABEL***
173 IF LEFT$(DU$,1)=" " THEN 233
203 REM ***LA$(T,1)=LABEL AND LA$(T,2)=ADDRESS OF LABEL***
213 T=T+1: LA$(T,1)=LEFT$(DU$,6): LA$(T,2)=O$
223 REM ***DEFINE MNEMONIC (MN$) AND ITS LENGTH***
233 MN$=MID$(DU$,9,6)
243 IF RIGHT$(MN$,1)=" " THEN MN$=LEFT$(MN$,LEN(MN$)-1): GOTO 243
253 GOSUB 40: RETURN
1000 DEFINT A-Z: CLS: PRINT CHR$(23): PRINT @ 462,"1802 CROSS-ASSEMBLER";: FOR K
=1 TO 1000: NEXT: CLS
1005 REM === J.G. CAYER  BOX 2034  HINTON  ALBERTA  T0E 1C0 ===
1010 CLEAR 25000: DIM S$(512),LA$(50,2)
1015 ON ERROR GOTO 1200
1020 G$="0123456789ABCDEF"
1030 INPUT "ORIGIN";O$: IF LEN(O$)<>4 THEN 1030
1040 REM ***CHECK IF HEX ADDRESS***
1050 FOR K=1 TO 4: I=ASC(MID$(O$,K,1)): IF (I>47 AND I<58) OR (I>64 AND I<71) TH
EN NEXT ELSE 1030
1060 REM ***FIND OUT DEC ADDRESS***
1070 E$=O$: GOSUB 30: DE=U
1080 PRINT @ 896,"ADDR    CODE    LABEL    MNEM    OPER    COMMENTS": PRINT STRIN
G$(63,"-")
1090 GOSUB 143
1100 IF AD=0 GOTO 1130
1110 TT=TT+1
1120 S$(TT)=O$+"    "+OP$+STRING$(8-LEN(OP$)," ")+DU$
1130 DE=DE+AD: E=DE: GOSUB 4: O$=Z$
1140 GOTO 1090
1200 RESUME 1090
2000 REM ***KILL ROUTINE***
2010 HA$=MID$(DU$,17,4)
```

```
2030 FOR K=1 TO TT: IF LEFT$(S$(K),4)<>HA$ THEN NEXT
2050 Z$=LEFT$(S$(K),4): IF MID$(S$(K),13,1)<>" " THEN AD=3 ELSE IF MID$(S$(K),11
,1)<>" " THEN AD=2 ELSE AD=1
2060 NA=AD
2080 FOR Y=K TO TT-1: IF MID$(S$(Y+1),13,1)<>" " THEN AD=3 ELSE IF MID$(S$(Y+1),
11,1)<>" " THEN AD=2 ELSE AD=1
2090 S$(Y)=Z$+MID$(S$(Y+1),5,LEN(S$(Y+1)))
2100 E$=Z$: GOSUB 30: E=U+AD: GOSUB 4
2110 NEXT Y
2120 TT=TT-1: DE=DE-NA: RETURN
2500 REM ***PRINTER ROUTINE***
2510 CLS: PRINT @ 960,"PRINTER READY (Y/N)?"
2520 Q$=INKEY$: IF Q$="" THEN 2520 ELSE IF Q$="Y" THEN 2530 ELSE RETURN
2530 FOR K=1 TO TT: LPRINT S$(K): NEXT: RETURN
3000  REM ***SAVE TO DISK***
3010 PRINT @ 920,"";: INPUT "FILENAME ";F$
3020 IF LEN(F$)>8 THEN PRINT @ 920, STRING$(30,32);: GOTO 3010
3030 OPEN "O",1,F$
3040 PRINT #1,TT,DE,T
3050 FOR K=1 TO T: PRINT #1,LA$(K,1): PRINT #1,LA$(K,2): NEXT
3060 FOR K=1 TO TT: PRINT #1,S$(K): NEXT: CLOSE: RETURN
3500 REM ***LOAD FROM DISK***
3510 PRINT @ 920,"";: INPUT "FILENAME ";F$
3520 IF LEN(F$)>8 THEN PRINT @ 920, STRING$(30,32);: GOTO 3510
3530 OPEN "I",1, F$
3540 INPUT #1,TT,DE,T
3550 FOR K=1 TO T: INPUT #1,LA$(K,1): INPUT #1,LA$(K,2): NEXT
3560 FOR K=1 TO TT: INPUT #1,S$(K): NEXT: CLOSE: GOSUB 20: RETURN
4000 REM ***INSERT ROUTINE***
4010 HA$=MID$(DU$,17,4)
4020 FOR K=1 TO TT: IF LEFT$(S$(K),4)<>HA$ THEN NEXT
4030 Y$=S$(K): Y=K: O$=HA$: GOSUB 143: NE$=LEFT$(Y$,4)+"    "+OP$+STRING$(8-LEN(
OP$)," ")+DU$
4040 IF MID$(NE$,13,1)<>" " THEN AD=3 ELSE IF MID$(NE$,11,1)<>" " THEN AD=2 ELSE
 AD=1
4050 FOR K=TT TO Y STEP-1: E$=LEFT$(S$(K),4): GOSUB 30: E=U+AD: GOSUB 4: S$(K+1)
=Z$+MID$(S$(K),5,LEN(S$(K))): NEXT K
4060 S$(Y)=NE$: NE$="": TT=TT+1: DU$="": DE=DE+AD: RETURN
5000 REM ***HEX ROUTINE TO SCREEN***
5002 FOR K=17 TO LEN(DU$): W=ASC(MID$(DU$,K,1)): IF W>47 AND W<58 OR W=32 THEN N
EXT ELSE PRINT @ 904, "ERROR":RETURN
5010 E=VAL(MID$(DU$,17,LEN(DU$)))
5020 IF E>65535 THEN PRINT @ 904, "TOO BIG";:RETURN
5030 GOSUB 4: PRINT @ 936, Z$: RETURN
5500 REM ***DEC ROUTINE TO SCREEN***
5510 E$=MID$(DU$,17,4)
5520 IF RIGHT$(E$,1)=" " THEN E$=LEFT$(E$,LEN(E$)-1): GOTO 5520
5530 FOR K=1 TO LEN(E$): Z=ASC(MID$(E$,K,1)): GOSUB 2: IF FL=0 THEN NEXT: GOSUB
30: PRINT @ 935, U: RETURN
5540 FL=0: PRINT @ 904,"ERROR": RETURN
6000 REM ***CHANGE ROUTINE***
6010 HA$=MID$(DU$,17,4)
6020 FOR K=1 TO TT: IF LEFT$(S$(K),4)<>HA$ THEN NEXT
6030 O$=HA$: TE=K: S$(K)="": GOSUB 143: S$(TE)=O$+"    "+OP$+STRING$(8-LEN(OP$),
" ")+DU$: RETURN
```

COSMAC CONQUEST

by - Tony Hill 30-481 Pitfield Road, Milton, Ontario

The winner of the BYTE game contest in 1982 was a space war type game written in FORTH. Unfortunately, the program was not written in fig-FORTH, and was designed to run on APPLE computers.

The program on the following pages is a rewrite of the game, using fig-FORTH words and ACE VDU board configurations for its basis. Several bugs in the original version have been corrected, and the graphics redesigned to be more interesting. A few of the definitions were written in FORTH assembler to improve the video update speed.

The basic idea of the game is to use your two battle fleets to colonize as many planets as possible while fighting off the enemy fleets (controlled by the computer) that are trying to do the same thing. The screen shows the current status of your efforts and maintains a set of menues to allow you to select your action from the possibilities available. Also shown on the screen is a graphic scan of the space immediately around the battle fleet you are currently commanding.

Game play is fairly straight forward. The computer maintains a list of the currently available commands on the bottom half of the display. You can use them to move your fleets, fire on enemy fleets, or land on planets. Once you are on a planet, a new set of commands allows you to enlist troops, buy more ships, leave troops to garrison the planet or remove troops from the garrison. You can tax your planets at any time, but be aware that they may revolt when taxed, and you will need to leave troops on the planet if you wish to crush the revolt.

A more complete description of this game can be found in the December 1982 issue of BYTE if you are interested. However the best way to learn about the game is to play it!

Screen 2 of the listing contains the only word you should have to change to adapt the game to your VDU board. The word HIRES should cause your VDU to adopt the high resolution mode (which means you will need 6K of video memory of course). The game itself will use about 16K, mostly because of the large arrays it uses to maintain information about the galaxy. Note that unlike the original version, the galaxy in this version is fixed at a 32 by 32 size, because of the assembler definitions.

Even if you don't use this game, the listing may be worth a look, just to see how ARRAYs, CASE and CODE definitions are used in more complicated FORTH programs.

```
 0        ( ************************************************ )
 1        ( *                                              * )
 2        ( *   COSMAC CONQUEST                            * )
 3        ( *   ===============                            * )
 4        ( *                                              * )
 5        ( *   WRITTEN BY - ALAN SARTORI ANGUS            * )
 6        ( *                                              * )
 7        ( *   MODIFIED FOR FIG-FORTH AND THE ACE         * )
 8        ( *   VDU BOARD BY - T. HILL  JUNE/84            * )
 9        ( *                                              * )
10        ( *                                              * )
11        ( *   ORIGINAL PUBLICATION - DEC/82 BYTE         * )
12        ( *   WINNER OF THE BYTE 1982 GAME CONTEST       * )
13        ( *                                              * )
14        ( ************************************************ )
15        -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 2 - VIDEO WORDS   840617 ALH )
 1 HEX
 2 : HIRES FF F800 C! ;       : GCLR E000 1800 00 FILL ;
 3
 4 : PAGE      HERE  100 +  FF00 AND  HERE -  ALLOT  ;
 5
 6 E000 VARIABLE CURSOR                ( VIDEO TEXT DRIVER CURSOR )
 7 0000 VARIABLE SCREEN    79 ALLOT    ( BUFFER FOR SCAN DATA )
 8 0000 CONSTANT (GALAXY)              ( ARRAY STORAGE POINTER )
 9 0000 CONSTANT CHAR                  ( CHARACTER SHAPE TABLES )
10
11 : GALAXY    20 *  +    21 -   (GALAXY)   +  ;
12
13 PAGE   HERE   ' (GALAXY)   !   400 ALLOT
14
15        HERE   ' CHAR       !      -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 3 - SHAPE TABLE      840617 ALH )
 1
 2 (     ) 0000 , 0000 , 0000 , 0000 , 0018 , 1818 , 0000 , 1800 ,
 3 ( "   ) 0028 , 2828 , 0000 , 0000 , 0024 , 247E , 247E , 2400 ,
 4 ( $   ) 0010 , 3C50 , 3C0A , 3C10 , 0000 , 5555 , 5555 , 0000 ,
 5 ( &   ) 0000 , 0000 , 0000 , 0000 , 0000 , 0000 , 0000 , 0000 ,
 6 ( (   ) 0000 , 0000 , 0000 , 0000 , 0000 , 0000 , 0000 , 0000 ,
 7 ( *   ) 0000 , 0000 , 0000 , 0000 , 0000 , 0000 , 0000 , 0000 ,
 8 ( ,   ) 0000 , 0000 , 1818 , 0808 , 0000 , 003C , 0000 , 0000 ,
 9 ( .   ) 0000 , 0000 , 0018 , 1800 , 0000 , 0000 , 0000 , 0000 ,
10
11 -->
12
13
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 4 - NUMBERS SHAPE TABLE  840617 ALH)
 1
 2 ( 0 ) 1824 , 2424 , 2424 , 1800 , 0818 , 0808 , 0808 , 3C00 ,
 3 ( 2 ) 1824 , 0418 , 2020 , 3C00 , 3E02 , 040C , 0222 , 1C00 ,
 4 ( 4 ) 0818 , 2848 , 7C08 , 0800 , 3E20 , 203C , 0222 , 1C00 ,
 5 ( 6 ) 0608 , 101C , 1212 , 0C00 , 3E02 , 0408 , 1010 , 1000 ,
 6 ( 8 ) 1824 , 2418 , 2424 , 1800 , 1824 , 241C , 0404 , 0C00 ,
 7 ( : ) 0000 , 1818 , 0018 , 1800 , 0000 , 0000 , 0000 , 0000 ,
 8 ( < ) 0000 , 0000 , 0000 , 0000 , 0000 , 3C00 , 3C00 , 0000 ,
 9 ( > ) 0000 , 0000 , 0000 , 0000 , 1824 , 0408 , 1000 , 1000 ,
10 ( @ ) 0000 , 0000 , 0000 , 0000 ,
11 -->
12
13
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 5 - LETTERS SHAPE TABLE  840617 ALH)
 1
 2 ( A ) 041B , 1111 , 1F11 , 1100 , 1E09 , 090E , 0909 , 1E00 ,
 3 ( C ) 0E11 , 1010 , 1011 , 0E00 , 1E09 , 0909 , 0909 , 1E00 ,
 4 ( E ) 1F10 , 101E , 1010 , 1F00 , 1F10 , 101E , 1010 , 1000 ,
 5 ( G ) 0F10 , 1012 , 1111 , 0F00 , 1111 , 111F , 1111 , 1100 ,
 6 ( I ) 0E04 , 0404 , 0404 , 0E00 , 0101 , 0101 , 1111 , 0E00 ,
 7 ( K ) 1112 , 1418 , 1412 , 1100 , 1010 , 1010 , 1010 , 1F00 ,
 8 ( M ) 111B , 1515 , 1111 , 1100 , 1111 , 1915 , 1311 , 1100 ,
 9 ( O ) 1F11 , 1111 , 1111 , 1F00 , 1E11 , 111E , 1010 , 1000 ,
10 ( Q ) 0611 , 1111 , 1516 , 0B00 , 1E11 , 111E , 1412 , 1100 ,
11 ( S ) 0E11 , 0804 , 0211 , 0E00 , 1F04 , 0404 , 0404 , 0400 ,
12 ( U ) 1111 , 1111 , 1111 , 0E00 , 1111 , 110A , 0A04 , 0400 ,
13 ( W ) 1111 , 1115 , 151B , 1100 , 1111 , 0A06 , 0A11 , 1100 ,
14 ( Y ) 1111 , 0A04 , 0404 , 0400 , 1F01 , 0204 , 0810 , 1F00 ,
15   -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 6 - GRAPHIC SHAPE TABLE  840617 ALH )
 1
 2 ( [ ) 3C20 , 2020 , 2020 , 3C00 , 0040 , 2010 , 0804 , 0200 ,
 3 ( ] ) 3C04 , 0404 , 0404 , 3C00 , 003C , 3C3C , 3C3C , 3C00 ,
 4 ( _ ) 0000 , 0000 , 0000 , 0000 ,
 5
 6 (   ) 0000 , 0000 , 0000 , 0000 , 0018 , 3C7E , FF7E , 3C18 ,
 7 (   ) 0000 , 1824 , 2418 , 0000 , 0000 , 1824 , 2418 , 0000 ,
 8 (   ) 0018 , 187E , 7E18 , 1800 , 1C7F , 491C , 1C49 , 7F1C ,
 9 (   ) 4949 , 497F , 4949 , 4900 , 4924 , 8944 , 1488 , 5594 ,
10
11 -->
12
13
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 7 - TEXT I/O DRIVER  840617 ALH )
 1
 2 CODE GEMIT
 3  CURSOR LDI,     E PLO,    CURSOR 100 / LDI,     E PHI,
 4  E LDA,     8 PHI,     E LDN,     8 PLO,
 5  9 INC,     9 LDN,    0D XRI,
 6  Z IF,     00 LDI,     8 PLO,        ( CARRIAGE RETURN ? )
 7  ELSE,     07 XRI,
 8  Z IF,      8 GHI,     01 ADI,    8 PHI,     ( LINE FEED ? )
 9  ELSE,      06 XRI,
10  Z IF,      FF LDI,     8 PLO,    F7 LDI,     8 PHI,   ( FF ? )
11              8 SEX,
12             BEGIN,     00 LDI,       STXD,      8 GLO,
13             Z IF,       8 GHI,   E0 SMI,   ENDIF,
14             Z UNTIL,    8 STR,
15 -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 8                        840617 ALH )
 1
 2  ELSE,     04 XRI,   ( BACKSPACE ? )
 3  Z IF,     8 GLO,    Z IF,    8 GHI,   E0 SMI,   Z NOT
 4                            IF,   1F LDI,    8 PLO,    8 GHI,
 5                                   1 SMI,    8 PHI,    ENDIF,
 6                        ELSE,   8 DEC,   ENDIF,
 7  ELSE,   00 LDI,   F PLO,   9 LDN,    SHL,    SHL,
 8          DF IF,      F INC,    F INC,    ENDIF,
 9          SHL,   7 PLO,    F GLO, CHAR 100 / 1 - ADCI,   7 PHI,
10  BEGIN,   7 LDA,   8 STR,   8 GLO,   20 ADI,   8 PLO,   DF UNTIL,
11          08 INC,   8 GLO, 20 SMI,
12          Z IF,   8 PLO,   8 GHI,   01 ADI,   8 PHI,   ENDIF,
13          ENDIF,    ENDIF,   ENDIF,   ENDIF,
14 -->
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 9                          840617 ALH )
 1
 2   8 GHI,   F8 SMI,   Z IF,   E1 LDI,   8 PHI,   E0 LDI,   7 PHI,
 3   00 LDI,   8 PLO,   7 PLO,
 4   BEGIN,   8 LDN,   7 STR,   8 INC,   7 INC,   8 GLO,
 5            Z IF,    8 GHI,   F8 SMI,   ENDIF,   Z UNTIL,
 6   BEGIN,   00 LDI,   7 STR,   7 INC,   7 GLO,
 7            Z IF, 7 GHI,   F8 SMI,   ENDIF,   Z UNTIL,
 8   F7 LDI,   8 PHI,
 9   ENDIF,   8 GLO,   E STR,    E DEC,   8 GHI,   E STR,
10   9 DEC,   9 DEC,   9 DEC,
11 NEXT
12
13 -->
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 10 - PRINT WORDS   840617 ALH )
 1
 2 : GTYPE      OVER + SWAP  DO I C@ GEMIT LOOP ;
 3
 4 : GSPACES DUP IF 0 DO 20 GEMIT LOOP ELSE DROP ENDIF ;
 5
 6 : G.R        >R S->D   SWAP OVER DABS
 7            <# #S SIGN #>  R> OVER - GSPACES   GTYPE ;
 8
 9 : G.       0  G.R ;
10
11 : (G.")      R COUNT DUP 1+ R> + >R   GTYPE ;
12
13 :  G."       22  ?COMP   COMPILE (G.")   WORD HERE
14              C@ 1+ ALLOT ;     IMMEDIATE
15 -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 11 - SCANNER DISPLAY   840617 ALH )
 1
 2 PAGE
 3
 4 CODE   SCAN
 5  E SEX,
 6  SCREEN LDI,     E PLO,      SCREEN 100 / LDI,    E PHI,
 7  9 LDA,   7 PHI,    9 LDN,   7 PLO,
 8  1F ANI,   15 SMI,  00 LDI,   SHLC,   6 PLO,
 9  9 DEC,   9 DEC,   9 DEC,
10  E1 LDI,    8 PHI,
11  CHAR 100 / 2 + LDI,    F PHI,
12
13 -->
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 12                        840617 ALH )
 1
 2  BEGIN,
 3      14 LDI,    08 PLO,
 4      BEGIN,    7 LDA,    SM,    Z NOT
 5                IF,   ADD,   E STR,   F PLO,
 6                     BEGIN,   F LDA,   8 STR,   8 GLO,
 7                              20 ADI,   8 PLO,
 8                     DF UNTIL,
 9                ENDIF,   IRX,
10                7 GLO,   1F ANI,   Z IF,   7 DEC,   7 GLO,
11                                           1F SMI,  7 PLO,
12                                      ENDIF,
13               8 INC,  8 GLO,  1F SMI,
14      Z UNTIL,
15 -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 13                    840617 ALH )
 1
 2    6 GLO,  SHR,  7 GLO,
 3    DF IF,   35 ADI,
 4       ELSE, 15 ADI,
 5       ENDIF,
 6    7 PLO,  7 GHI,  00 ADCI,  7 PHI,
 7    (GALAXY)  100 / 4 + SMI,
 8    DF IF,  (GALAXY) 100 / ADI,   7 PHI,  ENDIF,
 9    8 GHI,  1 ADI,  8 PHI,  EC SMI,
10    Z UNTIL,
11    2 SEX,  NEXT
12
13 -->
14
15
```

```
 0 ( COSMAC COMQUEST - SCREEN 14 - LINE DRAWING   840617 ALH )
 1
 2 : HLINE   1F OVER C!
 3           OVER SWAP 1+
 4           DO  FF I C!  LOOP
 5           F0 SWAP C! ;
 6
 7 : VLINE   DO   10 I C!  20 +LOOP ;
 8
 9 : PIXEL    100 *  80 +  +  E000 + ;
10
11 -->
12
13
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 15 - DRAW-BOXES  840617 ALH )
 1 DECIMAL
 2
 3 : DRAW-BOXES
 4 0 8 PIXEL  0 0 PIXEL VLINE  0 15 PIXEL  0 9 PIXEL VLINE
 5 0 23 PIXEL  0 16 PIXEL VLINE  18 8 PIXEL  18 0 PIXEL VLINE
 6 18 15 PIXEL  18 9 PIXEL VLINE  19 15 PIXEL 19 0 PIXEL VLINE
 7 31 15 PIXEL  31 0 PIXEL VLINE 31 23 PIXEL 31 16 PIXEL VLINE
 8 18 0 PIXEL 0 0 PIXEL HLINE  18 2 PIXEL 0 2 PIXEL HLINE
 9 18 8 PIXEL 0 8 PIXEL HLINE  18 9 PIXEL 0 9 PIXEL HLINE
10 18 11 PIXEL 0 11 PIXEL HLINE  18 15 PIXEL 0 15 PIXEL HLINE
11 31 16 PIXEL 0 16 PIXEL HLINE  31 23 PIXEL 0 23 PIXEL HLINE
12 31 0 PIXEL 19 0 PIXEL HLINE  31 12 PIXEL 19 12 PIXEL HLINE
13 31 15 PIXEL 19 15 PIXEL HLINE  ;
14
15 -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 16 - J , VHTAB   840617 ALH )
 1
 2 HEX
 3
 4 CODE J
 5    9 INC,   9 INC,   9 INC,   2 GHI,   7 PHI,   2 GLO,   7 PLO,
 6    7 INC,   7 INC,   7 INC,   7 INC,   7 INC,   7 LDA,   9 STR,
 7    9 DEC,   7 LDN,   9 STR,   NEXT
 8
 9 CODE VHTAB
10    CURSOR 1+ DUP   100 / LDI,   8 PHI,    LDI,   8 PLO,
11    9 INC,   9 LDN,   8 STR,    9 DEC,   9 DEC,   8 DEC,
12    9 LDN,   E0 ADI,   8 STR,   9 DEC,   9 DEC,   9 DEC,     NEXT
13
14
15 DECIMAL  -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 17 -  CONSTANTS   840617 ALH )
 1
 2 32            CONSTANT SIZE
 3 SIZE 2 *      CONSTANT NO-OF-STARS
 4 SIZE 3 * 2 / CONSTANT NO-OF-PLANETS
 5 4             CONSTANT NO-OF-B-HOLES
 6 200           CONSTANT W1
 7 5             CONSTANT W2
 8 10            CONSTANT W3
 9 20000         CONSTANT SPEED
10
11 -->
12
13
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 18  - VARIABLES    840617 ALH )
 1
 2 0 VARIABLE TEMP1
 3 0 VARIABLE VTAX
 4 0 VARIABLE C-LEGIONS
 5 0 VARIABLE CLASS-TOTALS
 6 0 VARIABLE C-FLEETS
 7 0 VARIABLE LEN
 8 0 VARIABLE TROOPS
 9 0 VARIABLE RAND1
10 0 VARIABLE RAND2
11 0 VARIABLE X
12 0 VARIABLE Y
13 -->
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 19  - VARIABLES    840617 ALH )
 1
 2 0 VARIABLE BUY-S
 3 0 VARIABLE BUY-T
 4 0 VARIABLE LEG
 5 0 VARIABLE NEW
 6 0 VARIABLE COMP-START
 7 0 VARIABLE COMPUTER
 8 0 VARIABLE DIFF
 9 0 VARIABLE C-PLANETS
10 0 VARIABLE PLANETS
11 0 VARIABLE FLEET-FLAG
12 0 VARIABLE START
13 250 VARIABLE CREDIT
14 -->
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 20 - ARRAYS     840617 ALH )
 1
 2 : ARRAY
 3   <BUILDS DUP C, * ALLOT DOES>
 4   ROT 1 - OVER C@ * + + ;
 5
 6 SIZE SIZE ARRAY INFO1
 7
 8 SIZE SIZE ARRAY INFO2
 9
10  2 6      ARRAY FLEETS
11
12 -->
13
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 21 -    CASE      840617 ALH )
 1
 2 : CASE
 3   ?COMP CSP @ !CSP 4 ; IMMEDIATE
 4
 5 : OF
 6   4 ?PAIRS COMPILE OVER COMPILE = COMPILE 0BRANCH HERE 0 ,
 7   COMPILE DROP 5 ; IMMEDIATE
 8
 9 : ENDOF
10   5 ?PAIRS COMPILE BRANCH HERE 0 , SWAP 2 [COMPILE]
11  ENDIF  4 ; IMMEDIATE
12
13 : ENDCASE
14   4 ?PAIRS COMPILE DROP BEGIN SP@ CSP @ = 0= WHILE 2
15   [COMPILE] ENDIF  REPEAT CSP ! ; IMMEDIATE   -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 22 - DELAY,CLEAR-MSGE,BEEP  840615 )
 1
 2 : DELAY
 3     5000 0 DO LOOP ;
 4
 5 : CLEAR-MSGE
 6       23 17 DO   I 1 VHTAB 30 GSPACES LOOP ;
 7 -->
 8
 9
10
11
12
13
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 23 - SETUP WORDS    840617 ALH )
 1
 2 : XY@
 3   X @ Y @ ;
 4
 5 : CLEAR-SCREEN   GCLR ;
 6
 7 : CLEAR-DISP
 8    SCREEN 121 255 FILL ;
 9
10 : CLEAR-GALAXY
11   1 1 GALAXY SIZE SIZE * 0 FILL ;
12
13 : CLEAR-INFO
14   1 1 INFO1 SIZE SIZE * 0 FILL
15   1 1 INFO2 SIZE SIZE * 0 FILL ; -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 24 - RANDOM #'S      840617 ALH )
 1
 2 : RANDOM1
 3    RAND1 @ 37 * 651 + DUP RAND1  ! ABS SIZE MOD 1+ ;
 4
 5 : RANDOM2
 6    RAND2 @ 53 * 773 + DUP RAND2 ! ABS SIZE MOD 1+ ;
 7
 8 : EDGE-CHECK
 9    SIZE 1 - + SIZE MOD 1+ ;
10
11 : SHOW-CURSOR
12    95 GEMIT   08 GEMIT ;
13 -->
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 25 -   INPUT    840617 ALH )
 1 : INPUT
 2   0 BEGIN
 3       KEY  DUP 08 =
 4       IF  32 GEMIT DUP GEMIT GEMIT SHOW-CURSOR 10 / 0
 5       ELSE
 6         DUP 57 >
 7         IF DROP 1
 8         ELSE DUP 48 <
 9           IF DROP 1
10           ELSE DUP GEMIT  48 - SWAP 10 * + 0 SHOW-CURSOR
11           ENDIF
12         ENDIF
13       ENDIF
14     UNTIL ;    -->
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 26 - F, END-MSGE   840617 ALH )
 1
 2 : F
 3   FLEET-FLAG @ SWAP FLEETS ;
 4
 5 : END-MSGE
 6       19 7 VHTAB G." END OF GAME COMMANDER " ;
 7
 8 CODE  QON  SEQ,  NEXT      CODE QOFF  REQ,  NEXT
 9
10 : BEEP
11     30 0 DO QON QOFF LOOP ;
12
13 DECIMAL
14 -->
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 27 - SET-UP-GALAXY   840617 ALH )
 1
 2 : SET-UP-GALAXY
 3   NO-OF-STARS 0 DO 8 RANDOM1 RANDOM2 GALAXY C!
 4                LOOP
 5   NO-OF-PLANETS 0 DO RANDOM1 RANDOM2 OVER OVER 16 ROT ROT
 6                      GALAXY C!
 7                      RANDOM1  4 * 8 + ROT ROT INFO1 C!
 8                  LOOP
 9   NO-OF-B-HOLES 0 DO 56 RANDOM1 RANDOM2 GALAXY C!
10                 LOOP ;
11 -->
12
13
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 28 - INITIALIZE   840617 ALH )
 1
 2 : INITIALIZE
 3   BEGIN
 4     5 0 VHTAB G." WHAT DIFFICULTY LEVEL? (1-4) "
 5     INPUT DUP   5 < IF 1
 6                       ELSE DROP  0
 7                       ENDIF
 8   UNTIL
 9   DIFF !
10    GCLR                          5 9 VHTAB G." DO YOU WANT A"
11    7 9 VHTAB G." 1.  SHORT"      8 9 VHTAB G." 2.  MEDIUM"
12    9 9 VHTAB G." 3.  LONG"      11 9 VHTAB G." GAME? - "
13  -->
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 29                840617 ALH )
 1
 2   KEY 127 AND
 3   CASE
 4     49 ( 1 ) OF 350 LEN ! ENDOF
 5     50 ( 2 ) OF 700 LEN ! ENDOF
 6                 1500 LEN !
 7   ENDCASE
 8   12 GEMIT   10 9 VHTAB G." INITIALIZING "
 9   CLEAR-GALAXY  CLEAR-DISP  CLEAR-INFO  SET-UP-GALAXY
10   1 FLEET-FLAG !
11   250 CREDIT !
12   0 PLANETS !
13   0 C-PLANETS !
14   20 1 3 FLEETS ! 20 2 3 FLEETS !
15 -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 30                  840617 ALH )
 1   50 1 5 FLEETS ! 50 2 5 FLEETS !
 2   DIFF @ 4 * 0 DO
 3                 RANDOM1 RANDOM2 OVER OVER 40 ROT ROT GALAXY C!
 4                 15 ROT ROT INFO2 C!
 5                  LOOP
 6   48 22 18 GALAXY C! 48 18 22 GALAXY C!
 7   22 1 1 FLEETS C! 18 1 2 FLEETS C!
 8   18 2 1 FLEETS C!   22 2 2 FLEETS C!
 9   29 3 DIFF @ * - NEW !
10   15 DIFF @ 4 * * TROOPS !
11   20 DIFF @ * C-LEGIONS !
12   DIFF @ 4 * C-FLEETS  !
13   SPEED DUP COMPUTER !
14   COMP-START !
15   1 BUY-T !   1 BUY-S !  ; -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 31 -  DRAW-BORDERS    840617 ALH )
 1
 2 : DRAW-BORDERS
 3  CLEAR-SCREEN
 4   1  5  VHTAB  G." COSMAC "
 5   3  2  VHTAB G." PLANETS "          10  5 VHTAB G." VOGONS"
 6   13 2  VHTAB G." FLEETS  "          12  2 VHTAB G." PLANETS"
 7   14 21 VHTAB G." X="                14 26 VHTAB G." Y="
 8   4  2  VHTAB G." SHIPS  "           5  2  VHTAB G." LEGIONS"
 9   7  2  VHTAB G." SCORE  "           6  2  VHTAB G." CREDITS"
10   13 22 VHTAB G." FLEET "            14 2  VHTAB G." LEGIONS"
11  DRAW-BOXES ;
12
13 -->
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 32 - FIND-DIRECTION   840617 ALH )
 1
 2 : FIND-DIRECTION
 3   19 8 VHTAB G." WHICH DIRECTION ? "
 4    KEY 127 AND
 5   CASE
 6     87 (   UP ) OF  0 -1 ENDOF
 7     90 ( DOWN ) OF  0  1 ENDOF
 8     83 ( RITE ) OF  1  0 ENDOF
 9     65 ( LEFT ) OF -1  0 ENDOF
10                     0  0
11   ENDCASE
12  19 8 VHTAB 20 GSPACES
13  2 F C@ + EDGE-CHECK SWAP
14  1 F C@ + EDGE-CHECK SWAP ;
15 -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 33 - SHOW-COMMANDS   840617 ALH )
 1
 2 : SHOW-COMMANDS
 3     CLEAR-MSGE
 4     18 1 VHTAB G." A - LEFT       F - FIRE    "
 5     19 1 VHTAB G." S - RIGHT      T - TAX     "
 6     20 1 VHTAB G." W - UP         O - OTHER FLEET "
 7     21 1 VHTAB G." Z - DOWN       L - LAND "
 8 ;
 9
10 -->
11
12
13
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 34 - DRAW-SCAN    840617 ALH )
 1
 2 : DRAW-SCAN
 3     1 F C@ 5 -   EDGE-CHECK
 4     2 F C@ 5 -   EDGE-CHECK
 5     GALAXY   SCAN
 6      14 23 VHTAB 1 F C@ 2 G.R
 7      14 28 VHTAB 2 F C@ 2 G.R
 8  ;
 9 -->
10
11
12
13
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 35 -   DRAW-FIGURES   840617 ALH )
 1 : DRAW-FIGURES
 2    3 10 VHTAB PLANETS @ 5 G.R
 3    7 10 VHTAB PLANETS @ C-PLANETS @ - W1 *
 4               1 3 FLEETS @ 2 3 FLEETS @ + W2 * +
 5               1 5 FLEETS @ 2 5 FLEETS @ + W2 * +
 6               TROOPS @ W3 * -  0 MAX 5 G.R
 7   14 10 VHTAB TROOPS @ 5 G.R
 8   13 10 VHTAB C-FLEETS @  5 G.R
 9   12 10 VHTAB C-PLANETS @  5 G.R
10    4 10 VHTAB 3 F @  5 G.R
11    5 10 VHTAB 5 F @  5 G.R
12    6 10 VHTAB CREDIT @  5 G.R
13   13 28 VHTAB FLEET-FLAG @ 1 G.R ;
14 -->
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 36 - FLEET MOVEMENTS   840617 ALH )
 1
 2 : DRAW-DISPLAY
 3      DRAW-SCAN DRAW-FIGURES ;
 4
 5 : NEW-FLEET
 6     0 1 F C@ 2 F C@ GALAXY C!
 7     0 3 F !
 8     0 5 F ! ;
 9
10 : MOVE-FLEET
11     OVER OVER
12     0 1 F C@ 2 F C@ GALAXY C!
13     48 ROT ROT GALAXY C!
14     2 F C! 1 F C! ;
15 -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 37 - CHECK-POSITION   840617 ALH )
 1
 2 : CHECK-POSITION
 3   EDGE-CHECK SWAP EDGE-CHECK SWAP OVER OVER GALAXY C@
 4    CASE
 5      0   OF MOVE-FLEET DRAW-SCAN ENDOF
 6      56  OF CLEAR-MSGE
 7              19 9 VHTAB G." FLEET IN BLACK HOLE "
 8              MOVE-FLEET NEW-FLEET DRAW-DISPLAY DELAY DELAY
 9              CLEAR-MSGE  SHOW-COMMANDS ENDOF
10      DROP DROP
11    ENDCASE
12     ;
13 -->
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 38  -   MOVEMENTS    840617 ALH )
 1 : OTHER-FLEET
 2    FLEET-FLAG @ 1 =
 3    IF 2 FLEET-FLAG !
 4    ELSE 1 FLEET-FLAG !
 5    ENDIF
 6    DRAW-DISPLAY ;
 7
 8 : MOVE-LEFT
 9        1 F C@ 1 - 2 F C@  CHECK-POSITION ;
10 : MOVE-RIGHT
11        1 F C@ 1+ 2 F C@  CHECK-POSITION ;
12 : MOVE-DOWN
13        1 F C@  2 F C@ 1+ CHECK-POSITION ;
14 : MOVE-UP
15       1 F C@  2 F C@ 1 - CHECK-POSITION ;  -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 39 -  ENLIST    840617 ALH )
 1
 2 : ENLIST   BUY-T @ 0=
 3    IF
 4      5 BUY-T !
 5      RANDOM1 8 / XY@ INFO1 C@ 7 / + DUP TEMP1 !
 6      17 1 VHTAB G." COST PER LEGION ="  G.
 7      XY@ INFO1 C@ 6 / DUP LEG !
 8      19 1 VHTAB G." NO OF LEGIONS AVAILABLE="  G.
 9      21 1 VHTAB G." HOW MANY DO YOU REQUIRE:" INPUT
10      LEG @ MIN DUP TEMP1 @ * CREDIT @ >
11      IF    22 9 VHTAB G." NOT ENOUGH CREDIT" DROP DELAY DELAY
12      ELSE  5 F @ OVER + 5 F !
13            TEMP1 @ * CREDIT @ SWAP - CREDIT  !  ENDIF
14    ELSE 21 2 VHTAB G." NO TROOPS AVAILABLE YET."
15         DELAY DELAY  ENDIF  ;      -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 40 -  BUY SHIPS     840617 ALH )
 1
 2 : BUY   BUY-S @ 0=
 3   IF 5 BUY-S !
 4     RANDOM1 5 / XY@ INFO1 C@ 10 / + 1+ DUP TEMP1 !
 5     17 1 VHTAB G." COST PER SHIP ="  G.
 6     XY@ INFO1 C@ 8 / DUP LEG !
 7     19 1 VHTAB G." NO OF SHIPS AVAILABLE="  G.
 8     21 1 VHTAB G." HOW MANY DO YOU REQUIRE:" INPUT
 9     LEG @ MIN DUP TEMP1 @ * CREDIT @ >
10     IF  22 9 VHTAB G." NOT ENOUGH CREDIT" DROP DELAY DELAY
11     ELSE  3 F @ OVER + 3 F !
12           48 1 F C@ 2 F C@ GALAXY C!
13           TEMP1 @ * CREDIT @ SWAP - CREDIT  !  ENDIF
14   ELSE 21 2 VHTAB G." NO SHIPS AVAILABLE YET."
15        DELAY DELAY ENDIF ;      -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 41 - TROOP MOVEMENTS    840617 ALH )
 1
 2 : GATHER
 3     19 2 VHTAB G." REMOVE HOW MANY ? " INPUT
 4     XY@ INFO2 C@ MIN TEMP1 !
 5     5 F @ TEMP1 @ + 5 F !
 6     XY@ INFO2 C@ TEMP1 @ - XY@ INFO2 C! ;
 7
 8 : GARRISON
 9     19 2 VHTAB G." LEAVE HOW MANY ? " INPUT
10     5 F @ MIN TEMP1 !
11     5 F @ TEMP1 @ - 5 F !
12     XY@ INFO2 C@ TEMP1 @ + 255 MIN
13     XY@ INFO2 C! ;
14
15 -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 42 - FRIENDLY PLANET    840617 ALH )
 1
 2 : FRIENDLY-PLANET
 3   BEGIN  DRAW-DISPLAY CLEAR-MSGE
 4   17 1 VHTAB G." CLASS " XY@ INFO1 C@ 8 / G.  G."  PLANET."
 5   18 1 VHTAB G." LOCAL GARRISON IS " XY@ INFO2 C@
 6        G.  G."  LEGIONS"
 7   19 1 VHTAB G." DO WISH TO:"
 8   20 1 VHTAB G." 1. LEAVE TROOPS"
 9   20 16 VHTAB G." 2. REMOVE TROOPS"
10   21 1 VHTAB G." 3. BUY SHIPS"
11   21 16 VHTAB G." 4. ENLIST TROOPS"
12   22 1 VHTAB G." 5. LEAVE"
13   KEY 127 AND CLEAR-MSGE
14 -->
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 43                    840617 ALH )
 1
 2     CASE
 3        49 OF GARRISON 0 ENDOF
 4        50 OF GATHER 0 ENDOF
 5        51 OF BUY 0 ENDOF
 6        52 OF ENLIST 0 ENDOF
 7        1 ENDCASE
 8    UNTIL  ;
 9 -->
10
11
12
13
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 44 - COLONIZE    840617 ALH )
 1 : COLONIZE   CLEAR-MSGE
 2      XY@ INFO1 C@ 8 / RANDOM1 1 - 5 / 7 + * 10 / DUP TEMP1 !
 3      5 F @ > IF
 4              18 9 VHTAB G." YOUR FORCES RETREAT "
 5              20 9 VHTAB G." YOUR LOSSES="
 6              5 F @ 2 / DUP  G.  5 F @ SWAP - 5 F !
 7              DELAY DELAY  CLEAR-MSGE
 8            ELSE
 9              20 9 VHTAB G." PLANET CAPTURED"
10             18 9 VHTAB G." YOUR LOSSES="
11             TEMP1 @  G.  5 F @ TEMP1 @ - 5 F !
12             1 PLANETS +! 32 XY@ GALAXY C!
13             DELAY DELAY  CLEAR-MSGE FRIENDLY-PLANET
14           ENDIF ;     -->
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 45 -  EMPTY-PLANET    840617 ALH )
 1
 2 : EMPTY-PLANET
 3     18 2 VHTAB G." UNCOLONIZED CLASS " XY@ INFO1 C@ 8 /  G.
 4     G."  PLANET"
 5     19  4 VHTAB G." DO YOU WISH TO ATTACK?" KEY 127 AND 89 =
 6     IF COLONIZE ENDIF ;
 7
 8 : NOT-PLANET
 9     18 4 VHTAB G." NO PLANET HERE "
10     DELAY ;
11 -->
12
13
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 46 - ATTACK    840617 ALH )
 1
 2 : ATTACK
 3      CLEAR-MSGE
 4      XY@ INFO2 C@ RANDOM1 1 - 5 / 7 + * 10 / DUP TEMP1 !
 5      5 F @ >
 6      IF 18 3 VHTAB G." YOUR FORCES RETREAT "
 7         20 3 VHTAB G." YOUR LOSSES = "
 8         XY@ INFO2 C@ 5 F @ * TEMP1 @ / 2 / XY@ INFO2 C@ SWAP
 9         - XY@ INFO2 C!
10         5 F @ 2 / DUP  G.  5 F @ SWAP - 5 F !
11     ELSE 0 XY@ INFO2 C!
12         18 3 VHTAB G." PLANET CAPTURED "
13         20 3 VHTAB G." YOUR LOSSES = "
14         TEMP1 @  G.
15         5 F @ TEMP1 @ - 5 F !      -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 47            840617 ALH )
 1
 2       32 XY@ GALAXY C!
 3       1 PLANETS +!
 4       -1 C-PLANETS +!
 5       XY@ INFO1 C@ 8 / MINUS CLASS-TOTALS +!
 6       DELAY CLEAR-MSGE
 7       FRIENDLY-PLANET
 8    ENDIF
 9    DELAY       CLEAR-MSGE ;
10 -->
11
12
13
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 48 -  ENEMY PLANET    840617 ALH )
 1
 2 : ENEMY-PLANET
 3      XY@ INFO1 C@ 8 /
 4      18 2 VHTAB G." CLASS "  G.  G."  PLANET"
 5      20 2 VHTAB G." ENEMY GARRISON STRENGTH = "
 6      XY@ INFO2 C@  G.
 7       22 2 VHTAB G." DO YOU WISH TO ATTACK?" KEY 127 AND
 8       IF ATTACK ENDIF  ;
 9
10 -->
11
12
13
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 49 -  LAND     840617 ALH )
 1
 2 : LAND
 3     CLEAR-MSGE
 4     FIND-DIRECTION
 5     OVER OVER Y ! X !  GALAXY C@
 6     CASE
 7       16 OF EMPTY-PLANET    ENDOF
 8       24 OF ENEMY-PLANET    ENDOF
 9       32 OF FRIENDLY-PLANET ENDOF
10             NOT-PLANET
11     ENDCASE DRAW-DISPLAY SHOW-COMMANDS ; -->
12
13
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 50 -  REVOLT?    840617 ALH )
 1
 2 : REVOLT?
 3    20 1 VHTAB G." PLANET AT [" X @ G.
 4    G." ," Y @ G.  G."  ] REVOLTS" DELAY
 5    XY@ INFO1 C@ 8 / XY@ INFO2 C@ OVER OVER >
 6    IF DROP 16 XY@ GALAXY C!
 7       8 * 7 + XY@ INFO1 C! 0 XY@ INFO2 C!
 8       -1 PLANETS +!   7 EMIT
 9       22 13 VHTAB G."  SUCCEEDS"
10    ELSE SWAP 2 / - XY@ INFO2 C!
11        XY@ INFO1 C@ 7 OR XY@ INFO1 C!
12        22 13 VHTAB G." FAILS" ENDIF
13    DELAY DELAY  20 1 VHTAB 30 GSPACES
14    22 13 VHTAB 12 GSPACES ;  -->
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 51 - TAX         840617 ALH )
 1
 2 : TAX
 3    0 VTAX !   CLEAR-MSGE
 4    18 2 VHTAB G." TAX COLLECTED =" 0 5 G.R
 5    SIZE 1+ 1 DO
 6      SIZE 1+ 1 DO
 7         I J GALAXY C@ 32 =
 8           IF I J INFO1 C@ 3 * 5 / VTAX @ + DUP VTAX !
 9           18 17 VHTAB  5 G.R I J INFO1 C@ 7 AND
10           IF I J INFO1 DUP C@ 1 - SWAP C!
11           ELSE I X ! J Y ! REVOLT? ENDIF ENDIF
12    LOOP LOOP CREDIT @ VTAX @ + CREDIT !
13    DRAW-DISPLAY   SHOW-COMMANDS ;
14
15 -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 52 - COMPUTERS TURN   840617 ALH )
 1
 2 : COMPUTER-TURN
 3      -1 NEW +!  NEW @ 0=
 4      IF BEEP 1 C-FLEETS +! 64 10 DIFF @ * - NEW ! CLASS-TOTALS @
 5         16 / 20 MIN  DUP C-LEGIONS +! DUP TROOPS +!
 6         BEGIN RANDOM1 RANDOM2 OVER OVER GALAXY C@ 0=
 7               IF  OVER OVER 40 ROT ROT GALAXY C! INFO2 C! 1
 8               ELSE DROP DROP DROP 0 ENDIF
 9         UNTIL 13 10 VHTAB C-FLEETS @ 5 G.R  ENDIF
10        C-FLEETS @ 8 / 0 DO  RANDOM1 RANDOM2 OVER OVER GALAXY C@
11        CASE 16 OF OVER OVER OVER OVER 24 ROT ROT GALAXY C!
12                 C-LEGIONS @ 2 / DUP C-LEGIONS !
13                 ROT ROT INFO2 C! 1 C-PLANETS +!
14                 INFO1 C@ 8 / CLASS-TOTALS +! ENDOF
15    -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 53                        840617 ALH )
 1
 2            32 OF OVER OVER Y ! X ! INFO2 C@ C-LEGIONS @ 2 / <
 3                  IF C-LEGIONS @ 3 / C-LEGIONS !
 4                     24 XY@ GALAXY C!
 5                     XY@ INFO1 C@ 8 / CLASS-TOTALS +!
 6                     1 C-PLANETS +!
 7                     -1 PLANETS +!  CLEAR-MSGE
 8                     19 2 VHTAB G." COSMAC PLANET LOST TO VOGONS"
 9                     5 0 DO BEEP LOOP
10                     DELAY DELAY CLEAR-MSGE SHOW-COMMANDS ENDIF
11                     ENDOF
12            12 10 VHTAB  C-PLANETS @ 5 G.R
13            DROP DROP
14            ENDCASE
15        LOOP ;        -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 54 - FIRE     840617 ALH )
 1
 2 : FIRE
 3     0 X !   CLEAR-MSGE   2 F C@ 2 + DUP 3 -
 4     DO 1 F C@ 2 + DUP 3 -
 5        DO I EDGE-CHECK J EDGE-CHECK GALAXY C@ 40 =
 6           IF I EDGE-CHECK X ! J EDGE-CHECK Y ! ENDIF
 7        LOOP LOOP   X @ 0=
 8     IF   18 9 VHTAB G." NO ENEMY IN RANGE"
 9     ELSE 3 F @  XY@ INFO2 C@  OVER 4 * 10 /
10          OVER 4 * 10 / DUP  DUP  0=
11          IF DROP  18 3 VHTAB   G." ENEMY FLEET DESTROYED." ELSE
12          18 2 VHTAB G." DAMAGE REPORT-"  G.  G."  HITS."
13          ENDIF ROT ROT - 0 MAX DUP 0=
14          IF DROP TROOPS @ XY@ INFO2 C@ -
15 -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 55              840617 ALH )
 1
 2                 0 MAX    TROOPS !
 3                 0 XY@ GALAXY C!    -1 C-FLEETS +!
 4         ELSE    TROOPS @  OVER -
 5                 0 MAX    TROOPS !
 6                 XY@ INFO2 C!
 7         ENDIF
 8    - 0 MAX DUP 0= IF DROP NEW-FLEET
 9                    ELSE 3 F !   ENDIF
10    ENDIF
11    DELAY DELAY DRAW-DISPLAY  SHOW-COMMANDS ;
12
13
14  HEX
15 -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 56 - OBEY-COMMAND   840617 ALH )
 1 : OBEY-COMMAND
 2   BUY-S @ -DUP  IF 1 - BUY-S ! ENDIF
 3   BUY-T @ -DUP  IF 1 - BUY-T ! ENDIF
 4   KEY
 5     CASE   41 OF MOVE-LEFT ENDOF
 6            53 OF MOVE-RIGHT ENDOF
 7            57 OF MOVE-UP    ENDOF
 8            5A OF MOVE-DOWN  ENDOF
 9            4F OF OTHER-FLEET ENDOF
10            4C OF LAND        ENDOF
11            54 OF TAX         ENDOF
12            46 OF FIRE        ENDOF
13            1B OF WARM ENDOF
14          ENDCASE  SP! ;
15 -->
```

```
 0 ( COSMAC CONQUEST - SCREEN 57 -  COMPUTER?    840617 ALH )
 1
 2 : COMPUTER?
 3    COMPUTER @ 1 - DUP 0=
 4    IF COMP-START @ COMPUTER ! DROP 1
 5    ELSE COMPUTER ! 0 ENDIF ;
 6
 7 : GAME-END?  LEN @ 0= ;
 8
 9 -->
10
11
12
13
14
15
```

```
 0 ( COSMAC CONQUEST - SCREEN 58 - RESTART, CONQUEST  840617 ALH )
 1
 2 : RESTART
 3      CLEAR-DISP  DRAW-BORDERS DRAW-DISPLAY SHOW-COMMANDS
 4      BEGIN
 5       ?TERMINAL IF OBEY-COMMAND -1 LEN +! COMPUTER-TURN ENDIF
 6         COMPUTER? IF COMPUTER-TURN ENDIF GAME-END?
 7      UNTIL    END-MSGE ;
 8
 9 : CONQUEST
10      HIRES GCLR
11      4 5 VHTAB G." WELCOME TO COSMIC CONQUEST "
12      8 9 VHTAB G." HIT ANY KEY" KEY RAND1 !
13      10 9 VHTAB  G." AND AGAIN" KEY RAND2 !
14      GCLR
15      INITIALIZE RESTART ;    DECIMAL
```

Changing to serve you better.

RCA

Design-in contest.

Prove to yourself how easy it is to use RCA peripherals in your system. Enter your hottest design using any RCA peripheral with any 8-bit microprocessor (NMOS or CMOS).

First prize: IBM 5150 PC XT with a color display and graphic printer.

Second prize: RCA Selectavision VCR with remote control.

Third prize: RCA color video camera.

Everyone who enters will win a "Megachange" mug.

Free samples.

Whether or not you enter the contest, RCA will supply you with two free samples of the CMOS peripheral of your choice and one free QMOS logic sample. Just check the chart on the previous page and send in the coupon with your selection.

Or call toll-free (800) 526-2177.

Free Sample/Contest Entry Form.

Please mail your completed form to: RCA Solid State, P.O. Box 2900, Somerville, NJ 08876.

☐ I'd like to enter the design-in contest. Please send me an entry kit.

☐ I don't want to enter, but I do want samples.

☐ Please send me the following samples:

CMOS peripherals (2 maximum), types: _____

_____

QMOS high-speed logic (1 maximum), type: _____

_____

Name _____

Title _____

Company _____

Address _____

City/State/Zip _____

Telephone ( ) _____

To get contest entry kit or free samples, use this coupon or facsimile. Or call 800-526-2177. One entry per person. Deadline for entries is December 15, 1984. Winners will be announced by January 30, 1985. Void where prohibited by law.

NMOS systems get CMOS-cool.

Even systems based on NMOS micros can interface with CMOS savings by using RCA's broad line of CMOS peripherals.

RCA has tested and proven interfaces between CMOS peripherals and the most popular 8-bit micros, including Z80, 8085, 8048, NSC800, 6500, and of course the 1800 series and 6805.

With relatively simple redesign, even NMOS systems require less power, lower cooling costs, less weight and space with CMOS peripherals.

World's broadest line of CMOS peripherals.

RCA offers more CMOS peripherals than anyone else: 28 types tested and proven to interface with NMOS or CMOS 8-bit micros.

Cool logic too: QMOS.

To make the most of CMOS advantages, use QMOS high speed CMOS logic as glue types. Speeds are comparable to LSTTL with 1/1000 the quiescent power consumption and 75% better noise immunity RCA offers a broad line of HC parts to design-in, plus the industry's best selection of HCT drop-in replacements for LSTTL. Order a free sample today.

RCA CMOS vs NMOS peripherals.

Factor	NMOS	CMOS	CMOS Advantage
Operating power	1	1/10	Yes
Reliability	High chip temperature	Low chip temperature	Yes
Standby power	mW	μW	Yes
Speed	High	Medium/High	
Operating frequency range	Limited	DC to max	Yes
Operating voltage	4.5V to 5.5V	4V to 10V	Yes
Noise immunity	10% of supply voltage	30% of supply voltage	Yes
Temperature range	0° to 70°C	-40° to +85°C Military -55° to +125°C	Yes
Chip complexity and size	Same	Same	
Gate arrays and Semi-custom availability	No	Yes	Yes

Z80 is a registered trademark of Zilog Inc. 8085, 8048 are trademarks of [illegible]
NSC800 is a trademark of National Semiconductor Corporation

I/O PORTS

CDP1851	Programmable I/O port
CDP1852	Byte-wide I/O port
CDP1872	8-Bit input port
CDP1874	8-Bit input port
CDP1875	8-Bit output port
CDP6823	Parallel interface

SERIAL I/O

CDP1854A	UART
CDP6402	UART

BUFFERS

CDP1856	4-Bit bus buffer separator
CDP1857	4-Bit bus buffer separator

KEYBOARD INTERFACE

CDP1871A	Keyboard encoder

MEMORY I/O DECODERS

CDP1853	N-Bit 1 of 8 decoder
CDP1858	4-Bit latch/decoder
CDP1859	4-Bit latch/decoder
CDP1866	4-Bit latch/decoder
CDP1867	4-Bit latch/decoder
CDP1868	4-Bit latch/decoder
CDP1873	1 of 8 binary decoder
CDP1881	6-Bit latch/decoder (20 Pin)
CDP1882	6-Bit latch/decoder (18 Pin)

TIMER FUNCTIONS

CDP1863	8-Bit prog. freq. generator
CDP1878	Dual counter-timer
CDP1879C1	Real time clock
CDP6848	Dual counter-timer (motel bus)

INTERRUPT CONTROL

CDP1877	Programmable controller

VIDEO CONTROL GENERATOR

CDP1869	Video interface system (sound)
CDP1870	Video interface system (video)

MULTIPLY/DIVIDE

CDP1865	8-Bit programmable MDU

QMOS LOGIC

CD74HCT00	Quad nand gate
CD74HCT74	"D" flip-flop
CD74HCT138	3-to-8 line decoder
CD74HCT373	Octal latch
CD74HCT245	Octal transceiver

NAME: _____ DATE: _____

PRODUCT ORDER	QUANTITY	UNIT PRICE	TOTAL
CPU Board	_____	$50.00	_____
Backplane and I/O Board, Ver. 2	_____	40.00	_____
Front Panel (with EPROM Burner, Clock)	_____	35.00	_____
VDU Board, Ver. 2	_____	40.00	_____
64K Dynamic (4116) Board	_____	50.00	_____
Netronics - Ace Adapter Board	_____	25.00	_____
I/O Adapter for Backplane, Ver. 1	_____	20.00	_____
Disk Controller Board, Ver. 2	_____	50.00	_____
80 x 25 Video Board (Available Sept. 84)	_____	50.00	_____

Software - all cassettes supplied are in Netronics Cassette format

Fig FORTH - 6K - 0000H	_____	$10.00	_____
Tiny Pilot - 2K - 0000H	_____	10.00	_____
SYMON - 2K - 0000H	_____	10.00	_____
SYDOS - 4K - D000H	_____	10.00	_____
CHIP 8 AE - 1.5K - 1000H	_____	10.00	_____

Back Issues

"Defacto" Year 1 - 3 (Edited)	_____	$20.00	_____
Year 4 Reprint	_____	10.00	_____
Year 5 Reprint	_____	10.00	_____
Year 6 Reprint	_____	10.00	_____
Year 7 Reprint	_____	10.00	_____

Membership - Year 7

Current Year - Sept. '84 - Aug. '85
includes 6 issues of Ipso Facto

Canadian	_____	$20.00 Cdn.	_____
American	_____	20.00 U.S.	_____
Overseas	_____	25.00 U.S.	_____

PRICE NOTE

Prices listed are in local funds. Americans and Overseas pay in U.S. funds, Canadians in Canadian Funds. Overseas orders: for all items add $10.00 for air mail postage. Please use money orders or bank draft for promp shipment. Personal cheques require up to six weeks for bank clearance prior to shipping orders.

SALE POLICY

We guarantee that all our products work in an A.C.E. configuration microcomputer. We will endeavour to assist in custom applications, but assume no liability for such use. Orders will be shipped as promptly as payment is guaranteed.

NAME: _____

ADDRESS: _____

_____

_____

PHONE #: _____

Note: Please ensure that mailing address and phone number are accuate, complete, and printed .

ASSOCIATION OF COMPUTER-CHIP EXPERIMENTERS
PO Box 581,
Burlington, Ontario,
Canada,
L7R 3Y5

MEMBERSHIP RENEWAL 1984/85 (Year 8)

Please assist our Executive to provide appropriate club newsletter content, meeting content, and hardware and software products.

Micro in use : Manufacturer: _____

Ram in use: _____ k Eprom in use: _____ k Memory Map: _____ k

Monitor: Author: _____ Location: _____ Size: _____ k

Periferals in use: _____

_____

_____

Periferals Planned: _____

_____