SPHERE NEWSLETTER

OCTOBER 1979

VOLUME IV ISSUE 2
EDITORS:
ROGER J. SPOTT
JEFF BROWNSTEIN

SOFTWARE

HARDWARE

EDITOR'S MAILBAG

PLEASE SEND TYPED MATERIAL FOR NEXT ISSUE TO:
ROGER J. SPOTT
13975 CONNECTICUT AVENUE
WHEATON, MARYLAND 20906

MORE ON FORTH

I RECEIVED THE JUNE NEWSLETTER AND PROGRAMMA'S FORTH (VER. 1.1) ALMOST CONCURRENTLY. I READ TOM MEIER'S ARTICLE AND LARRY SAMBUCO'S COMMENTS (FROM JEFF'S DESK). THEN I DUG INTO FINDING OUT HOW THE SOFTWARE WORKS.

I SHOULD POINT OUT THAT THE PROGRAMMA IMPLEMENTATION IS NOT THE STANDARD IMPLEMENTATION. THE MEMORY EFFICIENCY IS LOWER THAN STANDARD FORTH BECAUSE THE PROGRAMMA FORTH WORDS ARE *ALL* EXECUTABLE CODE AND NOT ADDRESS LINKS. SECOND, THE WORD DEFINITION/RECOGINITION MECHANISM ONLY USES THE FOUR MOST SIGNIFICANT CHARACTERS, WHERE STANDARD FORTH USES FROM THREE TO FIVE CHARACTERS AND A CHARACTER COUNT. THIRD, SOME OF THE WORDS ARE NOT FORTH-78 COMPATIBLE: '<R' SHOULD BE '>R', 'ZERO' SHOULD BE 'ERASE' AND '/MOD' DOES NOT LEAVE THE STACK IN THE PROPER CONFIGURATION (THE '/MOD' AT $123F WORKED PROPERLY BUT FOR SOME REASON A '/MOD' AT $1ACF WAS INSERTED TO SWAP THE QUOTIENT AND REMAINDER). THE PROPER OPERATION SHOULD LEAVE THE QUOTIENT ON THE TOS (TOP-OF-STACK) AND THE REMAINDER ON (TOS-1). MORE INFORMATION ON FORTH-78, I.E. REFERENCE CARD, IMPLEMENTATION MANUAL, CODE LISTINGS FOR VARIOUS MICROS, ETC., CAN BE OBTAINED FROM:

FORTH INTEREST GROUP
PO BOX 11Ø5
SAN CARLOS CA 94Ø7Ø.

I ATTACKED THE IDENTIFIED PROBLEMS: '+LOOP' NOT POPPING THE NORMAL STACK (SP), 'SAVE' ACTING LIKE 'KEEP', '<' AND '>' NOT WORKING ON MSBYTE, HARDWARE STACK DOWNWARD CREEPING ON ERROR OR SCREEN CLEAR, ETC. I FOUND SOLUTIONS TO ALL OF THE PROBLEMS AND ENCOUNTERED AND SOLVED A FEW MORE. THE HARDWARE STACK CREEPING PROBLLEM IS A PROBLEM COMMON TO OTHER PROGRAMMA SOFTWARE, E.G. THEIR ASSEMBLER. IT COMES ABOUT BECAUSE OF A HABIT OF DOING A 'JMP' OUT OF THEIR SUBROUTINES UPON ENCOUNTERING AN ERROR CONDITION (NOT CONSIDERED A VERY GOOD PRACTICE). THE SOFTWARE THEN RESTARTS THE SYSTEM WITHOUT CLEANING UP THE STACK, THUS LEAVING RETURN ADDRESSES, ETC. BEHIND. A FEW OF THOSE WILL EAT THE SYSTEM! THERE ARE TWO PATCHES TO CURE THIS PROBLEM (TABLE 1).

NEXT, THE '+LOOP' PROBLEM CAN BE CORRECTED WITH A MINOR PATCH (TABLE 1). EXECUTED AS IT EXISTS WITHOUT THE PATCH, THE STACK WILL HAVE ONE '+LOOP' PREFIX VALUE LEFT ON THE STACK FOR EACH LOOP EXECUTION. IN ADDITION, '+LOOP' DOES NOT ALLOW FOR NEGATIVE LOOP COUNTS, SO THAT PROBLEM WAS CORRECTED ALSO. 'SAVE' DOES ACT LIKE 'KEEP' SINCE THE CODE IS VERY SIMILAR. I PATCHED THE SOFTWARE (TABLE 1) SO THAT 'SAVE' NO LONGER LIMITS YOU TO JUST SAVING THE SCREEN. SINCE I HAVE TWO CASSETTES, I ALSO FIXED IT SO THAT 'SAVE' USES CAS1 ($FØ5Ø) AND 'LOAD' USES CAS2 ($FØ6Ø). YOU CAN CHANGE YOUR IMPLEMENTATION BY CHANGING MY PATCH AT $8C6 AND BY CHANGING MEMORY LOCATION $8DØ. THERE IS A PROBLEM THOUGH. 'LOAD' AND 'SAVE' HAVE SHARED SOFTWARE SO THAT 'LOAD' NOW WORKS LIKE 'SAVE' INSTEAD OF LIKE 'KEEP'. THIS PROBLEM WAS EASILY SOLVED WITH A DICTIONARY UPDATE (DISCUSSED LATER). IN ADDITION, BLOCKS CAN NOW BE LOADED OR DUMPED TO ARBITRARY LOCATIONS USING 'SAVE' AND 'RECALL'. I FOUND NO PROBLEM IN VER. 1.1 WITH '<' AND '>'. HOWEVER, I DID RUN ACROSS A COUPLE OF OTHER THINGS.

FIRST, CASSETTE BLOCK NUMBERS ARE BACKWARDS FROM WHAT MIGHT BE CONSIDERED NATURAL, I.E. 3Ø31 'KEEP' PRODUCED A BLOCK NUMBER ON TAPE OF 313Ø (31->$33 AND 3Ø->$34), SO I FIXED THAT (TABLE 1). FINALLY, THE SYSTEM DOES NOT PRINT "OK" AFTER PROPER EXECUTION OF AN INSTRUCTION, AS IS STANDARD. I FIXED THAT ONE ALSO (TABLE 1). IT SHOULD BE NOTED AT THIS POINT THAT YOU CANNOT TYPE OUT MORE THAN A FOUR DIGIT NUMBER IN ANY BASE OR YOU GET MSDIGITS WHICH APPEAR STRANGE. I AM WORKING ON THIS ONE. ALSO, 'C@' DOES NOT FOLLOW THE STANDARD, I.E. TOS IS THE BYTE ADDRESS AND 'C@' SHOULD LOAD THE ADDRESSED BYTE INTO THE LSBYTE OF THE TOS, BUT OTHER ROUTINES USE IT SO YOU CAN MAKE IT A DICTIONARY UPDATE!

Table 1   Sphere FORTH Ver. 1.1 Patches

Address	Old Contents	New Contents	Remarks
2E9	8A	87	H/W Stack Clean up
47A	8A	87	
8C6	BD Ø8 79	CE FØ 5Ø	SAVE/LOAD Patch
	BD Ø8 89	2Ø Ø7 Ø1	
	BD Ø8 93	Ø1 Ø1 Ø1	
8EA	DB	E4	
3Ø9	6C	9B	New End of Program
349	6C	9B	
5B4	BD Ø7 Ø5	7E 1B 6C	Print "OK" Patch
8D9	Ø6 2E	1B 94	Backwards BLK # Patch
1328	E2 ØØ 39	7E 1B 7B	+ LOOP Patch
1B6C		BD 17 FØ	Print "OK"
		CE Ø2 92	
		DF 74	
		BD Ø7 17	
		BD Ø7 Ø5	
		39	
1B7B		E2 ØØ	+ LOOP Patch
		Ø7	
		DE 54	
		6D ØØ	
		Ø8	
		Ø8	
		2A ØA	
		DF 54	
		Ø6	
		2C Ø2	
		4F	
		39	
		86 8Ø	
		39	
		DF 54	
		Ø6	
		39	
1B94		BD 12 A7	Backwards BLK # Patch
		BD Ø6 2E	
1B9A		39	

Table 2   Sphere FORTH Ver. 1.1 Commands

* No effect on the Line Entry Buffer

Command	Action
CTL H	Erase Last Character (Screen and Line entry buffer). It has no effect in the FSE mode.
← or __	Erase entire entered line (screen and line entry buffer). No effect in FSE.
CTL X	Clear screen, reset system and start over
CTL Q*	Move Cursor up one line
CTL S*	Move Cursor down one line
CTL R*	Move Cursor right one character position
CTL T*	Move Cursor left one cursor position

Cold start address is $2ØØ
Warm start address is $2Ø3

PROGRAMMA SENT A REFERENCE MANUAL BUT THEY NEGLECTED TO SEND ANY INFORMATION ON THE USER/EDIT COMMANDS. TABLE 2 IDENTIFIES THE COMMANDS THAT ARE AVAILABLE FOR CURSOR MOVEMENT AND FOR LINE EDITING.

I SHOULD INDICATE THAT ARITHMETIC ON THE SYSTEM IS SOMEWHAT INCONSISTENT. ADDITION AND SUBTRACTION ARE 2'S COMPLEMENT WHILE MULTIPLICATION AND DIVISION ARE UNSIGNED (MAGNITUDE ONLY).

TABLE 3 GIVES ADDRESSES THAT MAY BE OF INTEREST. THERE ARE TWO 512 ($2ØØ) BYTE BUFFERS AT THE TOP OF MEMORY WHICH CAN BE USED FOR DATA/BLOCK STORAGE AND RETRIEVAL. YOU CAN USE THEM OR 'SAVE'/'RECALL' THEM FROM CASSETTE (USING THE SUGGESTED PATCHES AND DICTIONARY UPDATES). PAGE Ø MEMORY ($54 TO $85) IS USED AS WORKING STORAGE FOR POINTERS, ADDRESSES, VARIABLES, ETC. THE BEGINNING AREA OF THE PROGRAM ($2Ø6 TO $22E) CONTAINS JUMPS TO SOME FREQUENTLY USED ROUTINES AND CONTAINS INITIALIZATION VALUES FOR THE POINTERS IN PAGE Ø.

TABLE 4 SHOWS THE LISTING OF THE DICTIONARY. IT SHOWS THE PROGRAMMA SUPPLIED WORDS, MY EXPANSION WORDS TO COVER THE PATCHES/FORTH-78 COMPATIBILITY AND WORDS REQUIRED FOR THE PRINTER ROUTINES. YOU MAY HAVE NOTICED THAT THERE ARE SOME WORDS IN THE DICTIONARY THAT ARE NOT EXPLAINED IN THE REFERENCE MANUAL. TABLE 5 GOES INTO DETAIL ON THESE WORDS. MAJOR ONES OF INTEREST ARE 'K', 'L', 'Q', 'S', AND 'X'.

NOW, AS I PROMISED, I WILL TALK ABOUT THE DICTIONARY UPDATES REQUIRED BY THE PATCHES/FORTH-78 COMPATIBILITY. INSTALL THE PATCHES (TABLE 1) AND COLD START FORTH. NOW ENTER THE DICTIONARY UPDATE BLOCK (TABLE 6). YOU MIGHT WANT TO ADD: " : C@ T - C# ; ", TO CORRECT THAT PROBLEM TOO. IF EVERYTHING GOES IN OKAY, I.E. "OK" AFTER EVERY NEW DEFINITION, THEN GET A NEW CASSETTE READY FOR RECORDING. NOW ENTER " HERE . " AND WRITE DOWN THE ADDRESS PRINTED ON THE CRT. NOW ENTER " $NEW ". YOU ARE NOW IN PDS DEBUG. SET UP THE CASSETTE ROUTINES TO RECORD: BLKNAM= "FO" ($46,$4F), WITH BUFFER START ADDRESS OF $22Ø AND BUFFER END ADDRESS OF THE RESULT OF " HERE . " THAT YOU WROTE DOWN. RECORD THE UPDATED SYSTEM (TWICE FOR SAFETY MAYBE). YOU NOW HAVE A UPDATED FORTH SYSTEM. TO RESTART FORTH, DO A *COLD* START.

ONE LAST THING REMAINS TO BE COVERED: THE OTHER FORTH BLOCKS IN TABLE 6. THEY COVER TWO BASIC AREAS, PRINTING AND SOME MULTIPLE PRECISION ROUTINES. THE PRINTER DRIVER IS NEARLY THE SAME AS THE ONE I PUBLISHED IN THE NEWSLETTER V. 4, N. 4, 2/79. THE PRINTER ROUTINES USE THE EXISTING CRT ROUTINES, EXCEPT FOR 'PCR', BY MODIFYING A JUMP ADDRESS. THE JUMP ADDRESS IS RESTORED BEFORE EXIT! NOTE THAT THIS METHOD WAS EASIER AND CONSERVED MORE MEMORY THAN REWRITING ALL OF THE ROUTINES; HOWEVER, I WILL ADMIT IS IS NOT A GOOD PRACTICE SINCE NEW VERSIONS OF FORTH MAY NOT USE THE SOME ADDRESSES!!!

THE UNSIGNED MAGNITUDE TEST DEVELOPES 'M<', 'M>', 'MMAX' AND 'MMIN', WHICH ARE UNSIGNED EQUIVALENTS OF '<', '>', 'MAX' AND 'MIN', RESPECTIVELY. IT ALSO SHOWS THE USE OF ASSEMBLY CODE AND MACRO DEFINITIONS IN FORTH. THE MULTIPLE PRECISION BLOCKS ARE USED TO DEVELOP '*/MOD' AND '*/' DEFINITIONS FROM STANDARD FORTH. FOR THE MOST PART THEY ARE DONE IN HIGHER LEVEL FORTH AND NOT IN ASSEMBLY LEVEL. THIS IMPLEMENTATION MAKES THEM MORE UNDERSTANDABLE (HOPEFULLY) BUT LESS MEMORY AND SPEED EFFICIENT. 'M*' PRODUCES A DOUBLE PRECISION PRODUCT OF THE TOS AND (TOS-1) AND REPLACES TOS AND (TOS-1) WITH THE RESULT. THE MOST SIGNIFICANT HALF RESIDES ON THE TOS. 'SETUP', AFTER DOING THE 'M*', TRUNCATES THE RESULT IN PREPARATION FOR THE '*/MOD' DIVISION BY DOING A 'MOD'. FOR EXAMPLE, IF I DO " FFFF FFFF 4 */MOD ", THE RESULT OF FFFF*FFFF IS FFFEØØØ1. DIVIDING THE RESULT BY 4 GIVES A QUOTIENT OF 3FFFCØØØ, WHICH CANNOT FIT INTO 16 BITS, SO THE 'MOD' THROWS OUT ANY PORTION OF THE DIVISION OPERATION WHICH WILL NOT FIT INTO A 16 BIT QUOTIENT, I.E. LEAVES ØØØ2ØØØ1. NOW THE RESULT OF THE '*/MOD' OPERATION RETURNS A REMAINDER OF ØØØ1

Table 3   Sphere FORTH Ver. 1.1 Memory Locations

Page Ø Addr.	Initialization Addr.	Definition
54/55	2ØA/2ØB	SP Stack Pointer (Mem End-$6Ø1)
5A/5B	2Ø8/2Ø9	Dictionary Pointer (H)
5D	214	Delimiter Character (DELIM)
5E/5F	212/213	Line Entry Buffer Start Addr (Mem End - $1ØØ)
60/61	221/222	Link to Last Dictionary Word
62/63		BASE
6F/72		First 4 char of New Word
74/75		Start Addr for printable string
80/81		Line Entry Buffer End Addr.
84/85	2ØC/2ØD	RP Stack Pointer (Mem End-$6ØØ)
86/87	2ØE/2ØF	Buffer 1 Start Addr (Mem End-$5ØØ)
88/89	21Ø/211	Buffer 2 Start Addr (Mem End-$3ØØ)
	2Ø6/2Ø7	Memory End
	2ØØ/2Ø2	Cold Start
	2Ø3/2Ø5	Warm Start
	215/217	A-Reg to Next Dictionary Location
	218/21A	PUTCHR
	21B/21D	GETCHR
	21E/22Ø	Clear Screen
	223/225	Print Error Message
	226/228	X-Reg+A&B-Reg → X-Reg
	229/22B	Look for next DELIM
	22C/22E	Look for next non-DELIM

Table 4. Dictionary Listing

PDIC	1E59	GETL	1E41	LEAV	1E31	NSPA	1E11	PCRT	1DCØ
P?BA	1DBØ	P?	1DAØ	PTYP	1D9Ø	'PCH	1D7B	P.#	1D6B
P.	1D5B	PSPA	1D4B	PMSG	1D3B	PECH	1D2B	CMSG	1DØ5
RMSG	1CEA	CECH	1CC4	RECH	1CA9	PCR	1C98	'PCH	1C83
PCHR	1C71	PINZ	1C55	SNDI	1C3F	PTCL	1C24	PTPR	1C15
✱LINK	1CØØ	ERAS	1BF6	L	1BE9	LOAD	1BD6	RECA	1BCC
>R	1BC2	MOD	1BB5	/MOD	1BA8	/	1B9B	★FORG	1B5F
ARRA	1B35	TRUN	1AEF	KEEP	1ADC	/MOD	1ACF	SAVE	1AC2
LOAD	1AB2	$NEW	1A3F	ZERO	1AØD	BYE	1AØ3	DICT	199B
SECO	1962	ABIT	1955	TYPE	1912	ABS	18FA	0	18CC
S	189E	X	1856	01	182C	S1	18Ø2	SPAC	17EA
RNDM	17C4	NOT	17BA	?BAS	179E	?	178Ø	.#	16F2
MIN	16CE	MAX	16AA	ROT	168E	=	166D	>	164A
<	1627	.	154A	"	1516	(	15Ø1	THEN	14F1
ELSE	14CF	IF	14B6	ABRT	14A1	-THE	1495	#DIG	148A
+LOO	147D	END	146D	LP1	1457	BEGI	144D	LIST	1438
LOOP	142B	ALOO	13DE	TEST	13BC	DO	13A9	LY+L	1394
3	137F	LXDO	136A	AND	135Ø	LXLP	133E	XDO	132F
X+LP	13ØC	XLP	12F3	ECHO	12EØ	Ø<	12CA	Ø=	12B2
S*AB	12A1	XOR	1287	OR	126D	RTS	1261	/MOD	123F
*	1222	OVER	12Ø6	|	11FØ	R>	11D7	<R	11BF
MINU	11A9	2*	119C	1+	118F	-	1172	SWAP	1159
+	113B	FSE	1135	OCTA	111A	HEX	10FF	DECI	1ØE4
+!	1ØC7	DUP	1ØB1	HERE	1ØA4	BASE	1Ø8F	H	1Ø7A
!	1Ø61	C!	1Ø4C	@	1Ø37	C@	1Ø23	DOWN	1Ø16
DROP	1ØØ9	DWN.	ØFFC	DRP.	ØFEF	RS=	ØFD3	RS@	ØFB7
SP=	ØF9B	SP@	ØF7F	RS=X	ØF6F	X=RS	ØF5F	Ø	ØF4A
1	ØF35	SP	ØF2Ø	RS	ØFØB	INC	ØFØØ	JSR	ØEF5
+WRD	ØEE8	CLI	ØEDC	CLV	ØEDØ	PULB	ØEC4	PULA	ØEB8
STX	ØEAD	STS	ØEA2	ROR	ØE97	ROL	ØE8C	NEG	ØE81
LSR	ØE76	LDX	ØE6B	LDS	ØE6Ø	TST	ØE55	DEC,	ØE4A
COM	ØE3F	CLR,	ØE34	ASR	ØE29	ASL	ØE1E	BVS	ØEØF
BVC	ØEØØ	BSR	ØDF1	BRA	ØDE2	BPL	ØDD3	BNE	ØDC4
BMI	ØDB5	BLT	ØDA6	BLS	ØD97	BLE	ØD88	BHI	ØD79
BGT	ØD6A	BGE	ØD5B	BEQ	ØD4C	BCS	ØD3D	BCC,	ØD2E
WAI	ØD22	TXS	ØD16	TSX	ØDØA	TSTB	ØCFE	TSTA	ØCF2
TBA	ØCE6	TAP	ØCDA	TPA	ØCCE	TAB	ØCC2	SWI	ØCB6
SEV	ØCAA	SEI	ØC9E	SEC	ØC92	SBA	ØC86	RTS	ØC7A
RTI	ØC6E	RORB	ØC62	RORA	ØC56	ROLB	ØC4A	ROLA	ØC3E
DAA,	ØC32	PSHB	ØC26	PSHA	ØC1A	NOP	ØCØE	NEGB	ØCØ2
NEGA	ØBF6	LSRB	ØBEA	LSRA	ØBDE	INS	ØBD2	INCB	ØBC6
INCA	ØBBA	DES	ØBAE	DCB,	ØBA2	DCA,	ØB96	COMB	ØB8A
COMA	ØB7E	CLRB	ØB72	CLRA	ØB66	CLC	ØB5A	CBA,	ØB4E
ABA,	ØB42	ASRB	ØB36	ASRA	ØB2A	ASLB	ØB1E	ASLA	ØB12
BITB	ØBØ7	BITA	ØAFC	CPX	ØAF1	CMPB	ØAE6	CMPA	ØADB
EORB	ØADØ	EORA	ØAC5	ANDB	ØABA	ANDA	ØAAF	ORAB	ØAA4
ORAA	ØA99	SBCB	ØA8E	SBCA	ØA83	SUBB	ØA78	SUBA	ØA6D
STAB	ØA62	JMP	ØA57	ACB,	ØA4C	ACA,	ØA41	ADB,	ØA36
ADA,	ØA2B	LDAB	ØA2Ø	STAA	ØA15	6#	Ø9EC	K	Ø9E2
,X	Ø9D3	X=SP	Ø9C2	SP=X	Ø9B1	#	Ø98D	MNUM	Ø981
LDAA	Ø976	INX	Ø96A	DEX	Ø95E	D=	Ø954	KEEP	Ø941
L	Ø926	;S	Ø91D	EXEC	Ø8F4	LOAD	Ø8F3	SAVE	Ø8A2
LDHF	Ø88D	BUFH	Ø883	BUFL	Ø873	BKSP	Ø86A	CLR	Ø854
C,	Ø846	]	Ø83E	[	Ø824	,	Ø81B	IMME	Ø8ØE
'	Ø7E7	CONS	Ø79C	VARI	Ø763	:	Ø743	:	Ø72A
MSG	Ø7ØA	CR	Ø6FF						

✱ Start of expanded Dictionary

★ Start of Programma Dictionary

Table 5. Additional Unexplained Sphere FORTH Ver. 1.1 Words

Word	Remarks
K	Clear screen (CLR)
L	Does a LOAD then an EXEC
BKSP	Backspaces Cursor, erasing character with no effect on the Line Entry Buffer
MNUM	Used as part of assembler to build the opcode & to put the opcode into the dictionary
+WRD	Finds the start of the next entry word
Ø 1 3	Constants representing these values
XLP	Increments loop count & does (Loop Cnt - Max Cnt)
X+LO	Does (TOS+Loop Cnt) → Loop Cnt & does (Loop Cnt - Max Cnt)
XDO	Does a SWAP then a <R then another <R
LXLP	Puts execution addr. of XLP on TOS
LXDO	Puts execution addr. of XDO on TOS
LX+L	Puts execution addr. of X+LO on TOS
TEST	Checks TOS sets A-Reg = { Ø if TOS = Ø & then does a DROP 1 if TOS Ø FF if TOS Ø
ALOO	Macro for LOOP to terminate/continue loop & clean up RS
LTST	Puts execution addr. of TEST on TOS
LP1	Macro for setting up IF using TEST
-THE	Routine to print an Error Ø9 message
ABRT	Puts execution addr. of -THE on TOS (flags no THEN)
S1	Uses the TOS, adds 2 to TOS, prints the value and uses the value as an addr. and prints the contents of the addr.
Q1	Same as S1 but subtracts 2
X	Puts SP on TOS & uses S1 to print the top 5 addr./values of the Normal Stack
S	Expects an addr. on TOS. It takes the addr. & displays the next 12 addr/values starting at that addr. To display the next 12 just type S. It leaves the next addr on the TOS
Q	Same as S but goes backwards through memory

ON (TOS-1) AND A QUOTIENT OF CØØØ ON TOS. WHAT ALL OF THIS AMOUNTS TO IS THAT THERE ARE 3FFF COMPLETE DIVISIONS BY 4 FOLLOWED BY 1 DIVISION BY 4 LEAVING CØØØ AND ØØØ1. YOU DO NOT SEE OR DO THE 3FFF DIVISIONS BECAUSE THE 16 BIT NUMBER SYSTEM CANNOT HOLD THE RESULT ANYWAY. THUS, FFFEØØØ1 = 3FFFØØØØ * 4 + CØØØ * 4 + ØØØ1. AS INDICATED, '*/MOD' DOES A (TOS-2)*(TOS-1)/TOS WITH A 32 BIT INTERMEDIATE RESULT. THE QUOTIENT GOES TO TOS AND THE REMAINDER GOES TO (TOS-1). '*/' IS THE SAME AS '*/MOD' EXCEPT THE REMAINDER IS THROWN AWAY.

BURIED WITHIN THESE FORTH BLOCKS ARE THREE USEFUL ROUTINES WHICH ARE ALSO FORTH-78 COMPATIBLE. YOU MAY WANT TO ADD THEM TO YOUR DICTIONARY, I.E. UPDATE IT FURTHER. THEY ARE 'NSPACES' AND 'LEAVE' (PRINTER ROUTINES PT. 3) AND 'AGAIN' (MULTIPLE PRECISION PT. 3). NOTE THAT 'NSPACES' IS NOT EXACTLY FORTH-78 COMPATIBLE--FORTH-78 USES 'SPACES', WHICH WOULD REPLACE 'SPACE' IN THE EXISTING DICTIONAUY BECAUSE OF THE WAY THE WORD IS HANDLED IN THE LINKING PROCEDURES (ONLY THE FIRST FOUR CHARACTERS ARE RETAINED). 'NSPACES' USES TOS AND OUTPUTS THAT MANY " "'S. 'LEAVE', WHEN USED WITHIN A 'DO-LOOP/+LOOP', WILL CAUSE EARLY LOOP TERMINATION. 'AGAIN' IS PART OF A 'BEGIN'-TYPE LOOP WHICH CREATES A WHILE-DO-TYPE CONSTRUCT:

```
BEGIN A T IF B AGAIN.
```

THE CONSTRUCT EXECUTES 'A' ATLEAST ONCE. IF 'T'<>Ø THEN IT EXECUTES 'B' AND BRANCHES BACK TO EXECUTE 'A'. IF 'T'=Ø THEN 'B' IS NOT EXECUTED AND THE LOOP TERMINATES. NOTE THAT 'AGAIN' IS AN 'IMMED'-TYPE OF INSTRUCTION AND THUS MUST BE USED WITHIN A ':' DEFINITION. IF 'AGAIN' IS ABSENT AFTER THE 'IF', AN "ERROR Ø9" WILL RESULT.

WELL THAT IS IT FOR NOW. IF YOU HAVE ANY QUESTIONS OR COMMENTS, PLEASE FEEL FREE TO CONTACT ME. MORE TO FOLLOW AS TIME ALLOWS. HAPPY COMPUTING.

YOURS TRULY,
CHARLES E. BURTON, PH.D.
1618 MARILYN AVE.
DAYTON OH 45420
(513) 254-2766

Table 6. FORTH Blocks

```
( DICTIONARY UPDATES )
: / /MOD DROP ;
: /MOD /MOD SWAP ;
: MOD /MOD DROP ;
: >R <R ;    : RECALL LOAD ;
: LOAD BUFL BUFH LDBF LOAD ;
: L LOAD EXEC ;   : ERASE ZERO ;
6Ø CONSTANT LINK
;S
```

Required for Forth Patches & Forth 78 compatibility

/ : (TOS-1)/TOS leaving quotient on TOS

/MOD : (TOS-1)/TOS leaving quotient on TOS & remainder on (TOS-1). Forth 78 compatible

MOD : (TOS-1)/TOS leaving remainder on TOS

>R : Forth 78 compatible

RECALL : Dual of SAVE (Note SAVE now works as advertised)

LOAD : Dual of KEEP

L : Loads Block # on TOS & executes it.

ERASE : Forth 78 compatible

LINK : Address of Dictionary Link Pointer

```
( PRINTER DRIVER )
FØ42 CONS PTPR FØ43 CONS PTCL
: SNDIT [ PSHA PTPR STAA @# 36
 LDAA # PTCL STAA @# 3E LDAA #
 PTCL STAA @# PULA ] ;
: PINZ [ PTCL CLR, @# FF LDAA #
 PTPR STAA @# 3E LDAA # PTCL
 STAA @# PTPR LDAA @# ØD LDAA #
 ' SNDIT JMP @# ;
: PCHR [ PTCL TST @# FB BPL PTPR
 TST @# ' SNDIT JMP @# ;
' PCHR CONSTANT 'PCHR   PINZ
;S
```

PTPR : Printer Data Port Address

PTCL : Printer Control Part Address

SNDIT : Send a Character

PINZ : Initalize Printer & port

PCHR : Print a character, Character to be printed must be in the A-Register. This is the entry point for printing

Note : This is the same driver as the one published in the Newsletter V.4, N.4, 2/79

```
( PRINTER ROUTINES PT. 1 )
: PCR [ 0D LDAA # ] PCHR [ 0A
 LDAA # ] PCHR ;
: RECHO 'PCHR 12ED ! ;
: CECHO FCHC 12ED ! ;
: RMSG 'PCHR 722 ! ;
: CMSG 218 722 ! ;
: PECHO RECHO ECHO CECHO ;
: PMSG RMSG MSG CMSG ;
: PSPACE RECHO SPACE CECHO ;
: P. RECHO . CECHO ;
: P.# RECHO .# CECHO ;
~~: PCHR CONS 'PCHR~~
;S
```

PCR : Prints CR/LF on printer
RECHO : Changes ECHO output routine addr. to PCHR
CECHO : Restores ECHO output routine addr to PUTCHR
RMSG : Same as RECHO for MSG
CMSG : Same as CECHO for MSG
PECHO : ECHO for printer
PMSG : MSG for printer
PSPACE : SPACE for printer
P. : . for printer
P.# : .# for printer

```
( PRINTER ROUTINES PT. 2 )
: PTYPE RECHO TYPE CECHO ;
: P? RECHO ? CECHO ;
: P?BASE RECHO ?BASE CECHO ;
: PCRT RECHO BUFH BUFL DO I 20
 TYPE PCR 20 +LOOP PCR PCR
 CECHO ;
;S
```

PTYPE : TYPE for printer
P? : ? for printer
P?BASE : ?BASE for printer
PCRT : Prints the entire CRT screen

```
( PRINTER ROUTINES PT. 3 )
: NSPACES 0 DO SPACE LOOP ;
: LEAVE DRP. I >R ;
: GETLINK E0 @ ;
: PDICT RECHO GETLINK [ HERE >R
 ] PCR 5 0 DO DUP 4 TYPE 2
 NSPACES DUP .# 5 NSPACES 4 + @
 DUP NOT IF LEAVE THEN LOOP DUP
 IF [ R> JMP @# ] THEN DROP PCR
 PCR CECHO ;
;S
```

NSPACES : Outputs the number of "6"s specified by value on TOS
LEAVE : Terminate loop at next LOOP or +LOOP
GETLINK : Gets contents of Dictionary Link Pointer
PDICT : DICT for printer

```
( UNSIGNED MAGNITUDE TESTS )
: MTST 1 LDAA # 1 BRA CLRA CLRB
 SP= ;    : 2DUP OVER OVER ;
: MADJ IF SWAP THEN DROP ;
: M< - [ 4 BCC, MTST ] ;
: M> - [ 3 BNE TSTA 6 BEQ 4 BLS
 MTST ] ;
: MMAX 2DUP M< MADJ ;
: MMIN 2DUP M> MADJ ;
;S
```

MTST : Macro for Comparison
MADJ : Stack adjust for maximum/minimum
M< : Unsigned < test. True if (TOS-1) less than TOS
M> : Unsigned > test. True if (TOS-1) greater than TOS
MMAX : Unsigned MAX
MMIN : Unsigned MIN
2DUP : Duplicates (TOS-1) & TOS & produces 2 additional values on the Normal Stack

```
( MULTIPLE PRECISION PT. 1 )
Ø VARI T1  Ø VARI T2  Ø VARI T3
Ø VARI T4          : *+ * + ;
: 2BYTES 1ØØ /MOD ;
: SPLIT ROT 2BYTES ROT ! SWAP
 ! ;    : GET2 @ SWAP @ ;
: M* T2 T1 SPLIT T4 T3 SPLIT T4
 T2 GET2 * 2BYTES T3 T2 GET2 *+
 2BYTES SWAP T4 T1 GET2 *+
 2BYTES ROT + ROT ROT 1ØØ * +
 SWAP T3 T1 GET2 *+ ;
: QUO@ T1 @ ;  : DVS@ T2 @ ;
: CNT@ T3 @ ;
;S
```

T1, T2, T3, T4 : Temporary Storage

2BYTES : Takes TOS & LSByte → (TOS-1) & MSByte → TOS. Creates an additional value on the Normal Stack

*+ : TOS * (TOS-1) + (TOS-2) Drops two values off the Normal Stack

SPLIT : Assumes two addresses are at TOS & (TOS-1) & a number is at (TOS-2). LSByte of (TOS-2) → addr at (TOS-1) & MSByte of (TOS-2) → addr at TOS

GET2 : Assumes two addresses are at TOS & (TOS-1). Contents of (TOS-1) addr → TOS & Contents of TOS → (TOS-1)

M* : Forms double precision product of TOS * (TOS-1). MS half → TOS & LS half → (TOS-1)

```
( MULTIPLE PRECISION PT. 2 )
: QUO! T1 ! ;  : DVS! T2 ! ;
: CNT! T3 ! ;  : CLRQ Ø QUO! ;
: QUO+ T1 +! ; : CNT+ T3 +! ;
: SETC 11 CNT! ;
: Ø>= Ø< NOT ;
: AGAIN SWAP JMP @# HERE SWAP !
 IMMED ;
: <SHF LDX #  1 ASL ,X Ø ROL ,X
 IMMED ;
: <DVD [ X=SP 3 ASL ,X 2 ROL ,X
 1 ROL ,X Ø ROL ,X ] ;
: SETUP >R M* ! MOD R> DVS!
 CLRQ SETC ;
;S
```

Ø>= : True if TOS >= Ø

AGAIN : BEGIN A t IF B AGAIN construct. Performs A atleast once. If t≠Ø then execute B and branch back to A. If t=Ø then B is not executed and the loop terminates

<SHF : Macro which uses TOS as an addr. & shifts the 2 bytes pointed to left one bit

<DVD : Shifts the top 4 bytes of the TOS, i.e. TOS & TOS-1, Left one bit.

SETUP : Set up for */MOD operation.

```
( MULTIPLE PRECISION PT. 3 )
  ( T1 = QUOTIENT    )
  ( T2 = DIVISOR     )
  ( T3 = SHIFT COUNT )

: */MOD SETUP [ HERE ] DUP DVS@
 M<  IF  [  T1  ] <SHF ELSE [
 HERE SWAP T1 ] <SHF 1 QUO+ DVS@
 - THEN -1 CNT+ CNT@ IF <DVD [ 3
 BCS ROT JMP @# SWAP JMP @# ]
 THEN SWAP DROP QUO@ ;

: */ */MOD SWAP DROP ;
;S
```

*/MOD : Performs (TOS-2) * (TOS-1) with a double precision result, then divides the product by TOS. The remainder → (TOS-1) & the quotient → TOS

*/ : Like */MOD but only leaves quotient on TOS

PRINTING SKILL ROUTINE TO TEXT EDIT PROGRAM

THIS ROUTINE APPENDS TO TEXT EDIT PROGRAM FOR W COMMAND. (SEE NEWSLETTER VOL II, ISSUE 3 JUNE, 1978) WHEN WE TYPE W XX,

"LINE WIDTH"WILL DISPLAY, IF WE TYPE 90 AND HIT RETURN KEY, THE PRINTER WILL PRINT OUT 90 CHARACTERS PER LINE. ETC.

"LINE OF PAGE?"WILL DISPLAY, IF WE TYPE 54 AND RETURN KEY, THE PRINTER WILL PRINT OUT 54 LINES PER PAGE. ETC.
"LENGTH OF PAPAGRAPH?"WILL DISPLAY. IF WE TYPE 2 AND RETURN KEY. THE PRINTER WILL PRINT OUT 2 POSITIONS PER PAPAGRAPH. ETC.
"SPACE OF FIRST LINE?" WILL DISPLY. IF WE TYPE 5 AND RETURN KEY, THE PRINTER WILL PRINT OUT 5 SPACES OF FIRST LINE PER PAPAGRAPH. ETC. NOW THE PRINTER IS GOING TO WORKING.
THIS ROUTINE IS A PRETTY PRINTING ROUTINE. FIRST CHARACTER AND LAST CHARACTER TO BE PLACE FIRST POSITION AND LAST POSITION EACH LINE AND PUT MORE SPACES ON THE RIGHT.
THE TEXT EDIT PROGRAM TO BE PATCHED LOCATION 0095 TO 57 06 FA AND 0200 TO CE 09 10. This routine from 06EA to 090E.

```
06EA   7C 00D4   INC
06ED   7C 00D7   INC
06F0   39        RTS
06F1   81 20     CMP A £
06F3   26 03     BNE    @06F8
06F5   7C 00D7   INC
06F8   20 32     BRA    @072C
06FA   BD 0310   JSR
06FD   37        PSH B
06FE   7E 086B   JMP
0701   33        PUL B
0702   01        NOP
0703   BD 08E2   JSR
0706   DE AE     LDX    %
0708   DF D2     STX    %
070A   A6 00     LDA A ,X
070C   08        INX
070D   81 20     CMP A £
070F   26 05     BNE    @0716
0711   A6 00     LDA A ,X
0713   08        INX
0714   20 F7     BRA    @070D
0716   09        DEX
0717   DF D2     STX    %
0719   86 3F     LDA A £
0713   97 D5     STA A %
071D   A6 00     LDA A ,X
071F   08        INX
0720   81 0D     CMP A £
0722   26 CD     BNE    @06F1
```

```
0724   DE D2     LDX    %
0726   BD 07CD   JSR
0729   7E 0761   JMP
072C   7A 00D5   DEC
072F   26 EC     BNE    @071D
0731   DE D2     LDX    %
0733   7F 00D4   CLR
0736   86 3F     LDA A £
0738   08        INX
0739   4A        DEC A
073A   26 FC     BNE    @0738
073C   A6 00     LDA A ,X
073E   81 20     CMP A £
0740   26 0C     BNE    @074E
0742   09        DEX
0743   A6 00     LDA A ,X
0745   81 20     CMP A £
0747   26 16     BNE    @075F
0749   7C 00D4   INC
074C   20 F4     BRA    @0742
074E   09        DEX
074F   A6 00     LDA A ,X
0751   81 20     CMP A £
0753   27 05     BEQ    @075A
```

```
0755   BD 06EA   JSR
0758   20 F4     BRA     @074E
075A   7C 00D4   INC
075D   20 E3     BRA     @0742
075F   8D 5C     BSR     @07BD
0761   86 3F     LDA A £
0763   97 B8     STA A %
0765   A6 00     LDA A ,X
0767   08        INX
0768   01        NOP
0769   01        NOP
076A   81 0D     CMP A £
076C   26 03     BNE     @0771
076E   7E 08B4   JMP
0771   81 00     CMP A £
0773   27 3D     BEQ     @07B2
0775   81 20     CMP A £
0777   26 19     BNE     @0792
0779   97 D6     STA A %
077B   86 00     LDA A £
077D   91 D4     CMP A %
077F   2D 0C     BLT     @078D
0781   7A 00B8   DEC
0784   27 19     BEQ     @079F
0786   86 20     LDA A £
0788   BD 06D0   JSR      print out
078B   20 03     BRA     @0790
078D   7A 00D4   DEC
0790   96 D6     LDA A %
0792   BD 06D0   JSR      print out
0795   7A 00B8   DEC
0798   27 05     BEQ     @079F
079A   A6 00     LDA A ,X
079C   08        INX
079D   20 CB     BRA     @076A
079F   86 0D     LDA A £
07A1   BD 07D2   JSR
07A4   7F 00D7   CLR
07A7   01        NOP
07A8   01        NOP
07A9   7E 0908   JMP
07AC   5A        DEC B
07AD   26 06     BNE     @07B5
07AF   BD 07D2   JSR
07B2   7E 0237   JMP
07B5   86 0D     LDA A £
07B7   BD 07D2   JSR
07BA   7E 0708   JMP
07BD   96 D4     LDA A %
07BF   9B D4     ADD A %
07C1   97 D4     STA A %
07C3   96 D7     LDA A %
07C5   90 D4     SUB A %
07C7   97 D4     STA A %
```

```
07C9   DE D2     LDX    %
07CB   39        RTS
07CC   39        RTS
07CD   86 3F     LDA A £
07CF   97 D4     STA A %
07D1   39        RTS
07D2   7F 00D7   CLR
07D5   BD 06D0   JSR     print out
07D8   7A 00D8   DEC
07DB   26 10     BNE    @07ED
07DD   37        PSH B
07DE   C6 0C     LDA B £
07E0   5A        DEC B
07E1   27 05     BEQ    @07E8
07E3   BD 06D0   JSR     print out
07E6   20 F8     BRA    @07E0
07E8   C6 31     LDA B £
07EA   D7 D8     STA B %
07EC   33        PUL B
07ED   39        RTS
07EE   7F 00D7   CLR
07F1   86 31     LDA A £
07F3   97 D8     STA A %
07F5   39        RTS
07F6   37        PSH B
07F7   DF B0     STX    %
07F9   BD FC4A   JSR
07FC   81 5F     CMP A £
07FE   26 0F     BNE    @080F
0800   DE 1C     LDX    %
0802   C6 20     LDA B £
0804   E7 00     STA B ,X
0806   E7 01     STA B ,X
0808   D6 1D     LDA B %
080A   5A        DEC B
080B   D7 1D     STA B %
080D   20 03     BRA    @0812
080F   BD FCBC   JSR
0812   DE B0     LDX    %
0814   33        PUL B
0815   39        RTS
```

```
0816   4C 49 4E 45 20 57 49 44 54 48 60 4C 49 4E 45 53
0826   20 4F 46 20 50 41 47 45 3F 60 53 50 41 43 45 20
0836   4F 46 20 46 49 52 53 54 20 4C 49 4E 45 3F 60 57
0846   49 44 54 48 20 50 41 52 41 47 52 50 48 3F 60
```

```
08DF   7E 0708   JMP
08E2   7D 0052   TST
08E5   26 02     BNE    @08E9
08E7   20 F6     BRA    @08DF
08E9   86 20     LDA A £
08EB   BD 06D0   JSR     print out
08EE   7A 0052   DEC
08F1   26 F6     BNE    @08E9
```

```
0856   CE 4E01   LDX     £
0859   86 20     LDA A £
085B   A7 00     STA A ,X
085D   08        INX
085E   BD 07F6   JSR
0861   A7 00     STA A ,X
0863   81 0D     CMP A £
0865   26 F6     BNE    @085D
0867   BD 0310   JSR
086A   39        RTS
086B   CE 0816   LDX     £
086E   BD 0528   JSR
0871   BD 0856   JSR
0874   F7 071A   STA B
0877   F7 0737   STA B
087A   F7 0762   STA B
087D   F7 07CE   STA B
0880   D7 50     STA B %
0882   CE 0821   LDX     £
0885   BD 0528   JSR
0888   BD 0856   JSR
088B   F7 07E9   STA B
088E   F7 07F2   STA B
0891   CE 0845   LDX     £
0894   BD 0528   JSR
0897   BD 0856   JSR
089A   D7 F0     STA B %
089C   D7 F1     STA B %
089E   CE 0830   LDX     £
08A1   BD 0528   JSR
08A4   BD 0856   JSR
08A7   D7 52     STA B %
08A9   D7 53     STA B %
08AB   BD 07EE   JSR
08AE   BD 05EA   JSR    initiate printer
08B1   7E 0701   JMP
08B4   5A        DEC B
08B5   26 06     BNE    @08BD
08B7   BD 07D2   JSR
08BA   7E 0237   JMP
08BD   7D 00F0   TST
08C0   26 08     BNE    @08CA
08C2   86 0D     LDA A £
08C4   BD 07D2   JSR
08C7   20 13     BRA    @08DC
08C9   7A 00F0   DEC
08CC   27 07     BEQ    @08D5
08CE   86 0D     LDA A £
08D0   BD 06D0   JSR    print out
08D3   20 F4     BRA    @08C9
08D5   BD 07D2   JSR
08D8   96 F1     LDA A %
08DA   97 F0     STA A %
08DC   BD 08E2   JSR
```

```
08F3   96 53     LDA A %
08F5   97 52     STA A %
08F7   96 50     LDA A %
08F9   90 53     SUB A %
08FB   B7 071A   STA A
08FE   B7 0737   STA A
0901   B7 0762   STA A
0904   B7 07CE   STA A
0907   39        RTS
0908   96 50     LDA A %
090A   8D EF     BSR    @08FB
090C   7E 0708   JMP
```

CSS Basic Modification:

by Roger J. Spott

This change in code increases the speed the interpreter looks through the command table by about 15-20%.

	New Code		Old Code
A01	08	A01	08
	E6 00		8C 21B1
	26 FB		27 1D
	8C 21B1		8C 1F6E
	27 19		27 1D
	8C 1F6E		E6 00
	27 19		26 F1
	08		08
	08		08

Thanks goes to Dave Lissiuk Springbok Digitronics for pointing out this one.

[illegible]

Parsing Strings in CSS Basic Version 4.0

Parsing is the process of comparing the name of a function or command in the Basic source program with all of the available ones in the command table. The match (if any) yields an address which the interpreter jumps to (thus handling a command or function). A good tutorial on 6800 parsing appeared in Kilobaud April 1979 (pp 86) by Gary Gaugler.

```
The parsing routines themselves are entered at 09E1 for commands
                                               09D5 for functions
                                               09CB for STEB or TO
```

Now the flow of the program is such that at 0ABD the program jumps to 09DA (read a character and parse). →EE LDX 00

```
                                     AD JSRX00  actually get you to
  (for the immediate mode)                      the appropriate handler
                                                routine.
```

During execution of a running program the parsing routine is reached when 0B1f 7E 156F jumps to 156F

```
156F DE 2C
     BD 09DA (parse)
     7E 0B51                0B51 EE LDX 00
                                 AD JSRX 00
                                 BD 09AA   (keep looking for a colon
                                            or end of line)
                                 20 A7  (to 0B01)
                                      CONTINUE BASIC
```

Now that we have identified the main "control loop" anotherportion of the program will need to be identified. The code which takes a line of inputted basic program (from peripheral or keyboard) which is now in the line input buffer (00A6-00FF) and absorbs it into the basic program with the line number in order. This requires a jump to ~~EE~~ 0CB9. In the case of a keyboard input 0AB5 contains the jump to 0CB9.

Remember, when using the cassette data files, that the input file buffer is limited (2325 to 21B3 inclusive). That is why records that are too long will be outputted to tape but cannot be read back.

```
10 INPUT "INPUT FILE NAME ", F$
20 OPENO F$
30 FOR X= 1TO 60
40 TWRITE X
50 NEXT X
60 CLOSE
```

FOR X= 1 TO 60 will work but FOR X = 1 TO 100 will not be read back!

ALSO- you may have difficulty if you try to put other Basic statements between OPENO (or OPENI) and TWRITE (or TREAD)

I recently ordered several copies of CSS (tape and manual) for the reduced rate on quantity purchases. There are two left for Sphere Users (25.00)

In Line Machine Language Subroutines for CSS Basic Version 4.0

The idea for this comes from the article in Interface Age Feb. 1979 by John P. Newcomer. My version used the input line buffer from 00A6 to 00FF to hold the actual machine code routine while it executes. This means that we are limited to forty two bytes of machine code when we do a PAT line.

This routine may be reused again and again in a program but remember that the machine code itself will be wiped out and re-converted from the basic line every time. Certainly a GOSUB may be used to do the job writing out the PAT line many times in a program.

The short machine language routine must end in a 39 or jump to a routine which ends in a 39.

Examples:

Store byte xx at location YYYY	PAT 86xxB7YYYY39 (sure beats decimal of POKE!)
Output a character (A)	PAT 8641bd01F139
Wait until some key is hit	PAT BDFC0A39
Execute subroutine	PAT 7E17EC HOME SUBROUTINE clears scrn.

The whole thing requires a command in the table: 50 41 54 00 2915

'PAT' MACHINE LANGUAGE SUBROUTINE FOR CSS BASIC

```
PAT     CE 00A6          USE LINE BUFFER
        DF 59            USE UNUSED POINTER
        DE 2C            GET BEGINNING BASIC LINE
READ    A6 00            LOAD CONTENTS
        27 1B            IF END OF LINE BRANCH TO EXECUTE
        8D 20            BSR INHEX (TEST AND CONVERTS HEX #)
        48
        48
        48
        48
        16
        A6 01
        8D 17            BSR INHEX
        1B
        DF 69            UNUSED POINTER FOR TEMP2
        DE 59
        A7 00            STORE HEX BYTE IN BUFFER
        08
        DF 59
        DE 69
        08
        08
        20 E1            LOOP BACK TO READ
 EXEC   BD 00A6
        39               RETURN
 ERR    7E 0F0B          ERROR #3
 INHEX  80 30
        25 F9            BRANCH IF CARRY SET TO ERR
        81 09
        23 08            BRANCH IF LOW OR SET TO END
        80 07
        25 F1            TO ERR
        81 0F
        22 ED            TO ERR
 END    39
```

CSS BASIC FOR CASSETTE VERSION 4.0 AND 4.3 LOW MEMORY POINTERS

10-20	GOSUB STACK AREA
20,21	BEGIN BASIC SOURCE CODE
22,23	NEXT BYTE AFTER BASIC PROGRAM
24,25	NEXT BYTE AFTER DEFINED VARIABLES
26,27	MEMORY LIMIT
28,29	´USER´ POINTER
2A,2B	LINE NUMBER BEING EXECUTED
2C,2D	POINTER INTO SOURCE CODE
2E,2F	NEXT LINE TO BE EXECUTED (WHEN JUMP)
30,31	POINTER TO NUMBER BUILD BUFFER
32,33	NEXT LINE TO BE EXECUTED AFTER ´CONT´
36,37	TEMP. STORAGE FOR ´LIST´ AND OTHERS
38,39	TEMP. STORAGE (UNDETERMINED)
3A,3B	TEMP. STORAGE FOR STACK POINTER
3C,3D	NEW LINE IF A JUMP
3E,3F	HIGHEST LINE NUMBER IN PROGRAM
40	FIRST ARGUMENT IN ´POKE´
40,41	USED TO STORE X WHEN ACCESSING STACK
42,43	INDEX REGISTER STACK POINTER
44,45	´DATA´ POINTER
46	´STRING=´
48	´POS´ VALUE
49,4A	HOLDS POINTER TO CONSTANT DURING TRIG. FUNCTIONS
4B	HOLDS RANGE INFORMATION FOR TRIG. FUNCTIONS
4C,4D	PLACE IN COMMAND TABLE WHERE BASIC WILL JUMP NEXT
4E	TEMP. STORAGE USED IN ´STEP´ AND ´SGN´
4F	LOCATION OF DECIMAL POINT (FROM LEFT)
50	SECOND ARGUMENT OF ´DIM´
51	FLOATING POINT CONTROL
54	STORAGE USED IN ´PEEK´
55	HOLDS FIRST ARGUMENT IN ´DIM´ AND OTHERS
58	´DIGITS´ VALUE
5D,5E	NEXT LINE TO BE EXECUTED IN BASIC SOURCE
5F,60	BEGIN ´FOR NEXT-LOOP´ BUFFER
61,62	POINTER INTO BUFFER TO SEE LOOP THAT IS ACTIVE
63	POINTER TO DEFINED VARIABLE TABLE
65,66	GOSUB STACK POINTER
67,68	USED IN ´RND´
73	NON TERMINAL FLAG
74	TEST IF STRING OR NUMERIC. NUMERIC=0
75	TEST IF ´TRACE´ IS ON
76,77	STRING BUILD BUFFER POINTER
78,79	APPROPRIATE I/O CONTROL CHARACTER POINTER
7A,7B	ACTUAL OUTPUT ROUTINE IN USE
7C,7D	ADDRESS IN PORT JUMP TABLE FOR THE INPUT PORT IN USE
7E,7F	MIKBUG PIA PORT ADDRESS
80,81	´LINE=´
82,83	BEGIN STRING BUILD BUFFER
84	´PORT#´
85-8A	FUNCTION ARGUMENT SQUARED : EXP; ATAN(2) IT CONTAINS 4
8C,90	NUMBER USED IN A FUNCTION
98,99	START OF CASSETTE FILE NAME BUFFER
9A	USED WITH CASSETTE ROUTINES IT=0 IF AT TERMINAL
9B	´RJUST=´ FLOATING POINT
9D	ONLY IN VERS. 4.3 FOR CASSETTE
A6-FF	BEGIN INPUT LINE BUFFER

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)
01	FE 1CF2
FE 1CF2	39
39	09 (MAKES ARCTAN WORK PROPERLY)

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

CHANGES TO EDIT:
22CB 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)

CORRECTIONS FOR PIE 1K EPROM VERSION A1.0 (BY TOM CROSLEY)
NEW SPHERE 64 CHARACTER VIDEO MODIFICATION

	(OLD)	(NEW)
F4C8	8A 1F	8A 3F
F58A	84 E0	84 C0
F5CB	CE E1E0	CE E3C0
F5E9	8A 1F	8A 3F
F627	CE E1E0	CE E3C0
F631	E7 20	E7 40
F63F	D6 1C	96 1C
	96 1D	D6 1D
	54	58
	46	49
	44	58
	44	49
	44 44	84 0F
F658	84 1F	84 3F
F65D	CE E1E0	CE E3C0
F678	CE E200	CE E400
F6C4	CE E200	CE E400
F6D8	8C E1E0	8C E3C0
F734	CE E020	CE E040
F737	8C E200	8C E400
F746	8A 1F	8A 3F
F7B8	81 1F	81 3F
F7BC	86 1F	86 3F
F7C0	C4 E0	C4 C0
F7CA	C4 1F	C4 3F
F7DF	84 1F	84 3F

The Amateur Computer Club of New Jersey has several hundred programs in SWTP Basic which should run in CSS easily without any modification. Unfortunately, these are all traded on FLEX disks only. We have plans to make cassette copies and make this huge program library available to all cassette users. What we need are some original Basic programs to give in return. What have you got? I would arrange to put them on FLEX disks to donate to the library. Thanks. Programs which appeared in magazines and you have keyed in and made to run on SWTP or CSSBasics are OK if you indicate the article along with the program to give credit. Usually a REM at the beginning is sufficient to show where the donated software came from and also where instructions for its use may be found. The library is not interested in copyrighted software.

8/17/79

MOD: CHANGE CRT BOARD FROM 32 TO 64 CHARS/LINE

1) IF STEP 29 WAS PERFORMED DURING MEMORY UPGRADE, REMOVE WIRE FROM E5-14 TO E5-8.

2) CUT LAND FROM E10-11 TO E4-14 AT E4 END (ORIG A'8).

3) CUT LAND FROM E10-8 TO E4-5 AT E4 END (ORIG A'7).

4) CUT LAND FROM E10-9 TO E4-2 AT E4 END (ORIG A'6).

5) CUT LAND FROM E10-12 TO E3-11 AT E3 END (ORIG A'5).

6) CUT LAND FROM E6-8 TO E3-14 AT E3 END (ORIG A'4).

7) CUT LAND FROM E6-9 TO E3-5 AT E3 END (ORIG A'3).

8) CUT LAND FROM E11-11 TO E6-1 AT E6 END (CARRY BIT FROM E11).

9) CONNECT E10-11 TO E5-14 (NEW A'9).

10) CONNECT E10-8 TO E4-14 (NEW A'8).

11) CONNECT E10-9 TO E4-5 (NEW A'7).

12) CONNECT E10-12 TO E4-2 (NEW A'6).

13) CONNECT E6-8 TO E3-11 (NEW A'5).

14) CONNECT E6-9 TO E3-14 (NEW A'4).

15) CONNECT E6-1 TO E6-12 (ADD NEW COUNTER STAGE FOR A'3).

16) CONNECT E6-14 TO E11-11 (CARRY BIT FROM E11 TO NEW COUNTER STAGE).

17) CONNECT E6-1 TO E3-5 (NEW A'3).

18) REMOVE C37 AND REPLACE WITH A 4.7 PF CAPACITOR (THIS ALLOWS DENSITY TO BE INCREASED TO ACCOMMODATE 64 CHARS).

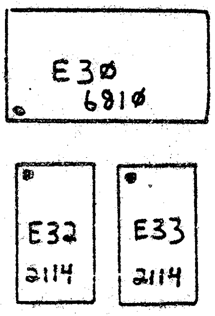
19) PLACE A 10K RESISTOR ACROSS R5 (MAKES CURSOR DISPLAY MORE RELIABLE).
NOTE: PERFORM STEP 19 ONLY IF YOUR CURSOR GIVES YOU PROBLEMS WHEN BACKED OVER A PREVIOUSLY TYPED CHARACTER. CHECK FOR THIS SYMPTOM BY GOING INTO EDIT MODE AND TYPING IN ABOUT 10 LETTER H'S. NOW BACK THE CUROSR OVER THE H'S AND IF WHEN THE CURSOR SHOULD BE ON, YOU INSTEAD SEE A 7, THEN DO STEP 19. WHAT YOU ARE ACTUALLY LOOKING AT IS THE COMPLEMENTED CHARACTER WITHOUT THE CURSOR FUNCTION, AND THIS PROBLEM WILL AFFECT ALL CHARACTERS.

20) AS THIS CHANGE WILL AFFECT ALL 3 ADJUSTMENT POTS, A DEFINITE ADJUSTMENT PROCEDURE CANNOT BE GIVEN. HOWEVER, I FOUND THE DENSITY POT (R21) TO REQUIRE THE MOST MOVEMENT, HENCE IT SHOULD BE THE FIRST ONE ADJUSTED.

8/17/79

MOD: Upgrade crt board from 512 to 1024 bytes.

1) Add 2 18-pin wire wrap sockets just below E30 on the CRT board and label them E32 and E33.



USE 300 ns OR LESS

2) Remove 4 6810's at E14, E20, E25 and E30.
3) Connect E32-18 to E33-18 to E30-24 (Vcc).
4) Connect E32-9 to E33-9 to E30-1 (Gnd).
5) Connect E32-5 to E33-5 to E30-23 (A0).
6) Connect E32-6 to E33-6 to E30-22 (A1).
7) Connect E32-7 to E33-7 to E30-21 (A2).
8) Connect E32-4 to E33-4 to E30-20 (A3).
9) Connect E32-3 to E33-3 to E30-19 (A4).
10) Connect E32-2 to E33-2 to E30-18 (A5).
11) Connect E32-1 to E33-1 to E30-17 (A6).
12) Connect E32-17 to E33-17 to E30-10 (A7).
13) Connect E32-16 to E33-16 to E30-13 (A8).
14) Connect X1-2 to E5-13 (A9).
15) Connect E32-15 to E33-15 to E5-12 (A9).
16) Connect E32-14 to E30-2 (D0).
17) Connect E32-13 to E30-3 (D1).
18) Connect E32-12 to E30-4 (D2).
19) Connect E32-11 to E30-5 (D3).
20) Connect E33-14 to E30-6 (D4).
21) Connect E33-13 to E30-7 (D5).
22) Connect E33-12 to E30-8 (D6).
23) Connect E33-11 to E30-9 (D7).
24) Connect E32-10 to E33-10 to E30-16 (R/W).
25) Connect E32-8 to E33-8 to E32-9 (CS always active).
26) Open land going from X1-2 to E13-13 at E13 end.
27) Connect E13-13 to E13-7 (this allows selection of board for addresses E000 - E3FF).
28) Install 2 2114 static rams at E32 and E33.
29) NOTE: If you are going to continue with the 64 char mod, then ignore this step, else, Connect E5-14 to E5-8 (this ties A'9 inactive so we can access the 1st 512 bytes).

V3A ROM changes for 64 char/line CRT MOD:

Address	Old data	New data
FB75	1E	3E
FB8F	1F	3F
FBAF	5F	BF
FBDA	3E	7E
FBDD	3F	7F
FC3B	E2	E4
FCC1	20	40
FCC9	20	40
FCE1	E2	E4
FD09,A	E1E0	E3C0
FD12	E0	C0
FD64	20	40
FD69,A	E1E0	E3C0
FD74,5	E1DF	E3BF
FD79	20	40
FD89	1F	3F

CHANGES TO THE V3N PROMS TO RUN WITH THE 64 CHARACTER MOD.

	FROM:	TO:
FC3F	CE E200	CE E400
FCCD	C6 20	C6 40
FCD5	C6 20	C6 40
FCE6	8C E200	8C E400
FD00	E0	C0
FD46	CE E1E0	CE E3C0
FD62	E6 20	E6 40
FD67	8C E1E0	8C E3C0
FD74	CE E1DF	CE E3BF
FD79	E7 20	E7 40
FD88	8C E01F	8C E03F

EDIT:

JOHN R. BAYLIS
ACCOUNTING SERVICES

1368 STELLARIA CIRCLE
BOUNTIFUL, UTAH 84010

TELEPHONE:
801/295-103

10 OCTOBER 1979

ROGER SPOTT AND READERS
13975 CONNECTICUT ROAD
WEATON, MARYLAND 20906

TO THOSE CONCERNED:

I AM CURIOUS AS TO THE NUMBER OF SPHERE SYSTEMS IN ACTUAL BUSINESS USE. I AM CURRENTLY USING MY COMPUTER SYSTEM IN AN ACCOUNTING PRACTICE, FOR GENERAL LEDGER WORK AND VARIOUS OTHER BUSINESS APPLICATIONS.

IF THERE ARE ANY OTHER SPHERE SYSTEM USERS APPLYING BUSINESS PROGRAMS, I WOULD ENJOY HEARING FROM YOU. PERHAPS WE COULD START A COLUMN IN THE NEWSLETTER FOR BUSINESS APPLICATIONS. I WOULD BE WILLING TO CONTRIBUTE.

RESPECTFULLY,

John R Baylis

JOHN R. BAYLIS C. P. A
PO BOX 137
N. S. L., UT 84054

[illegible]

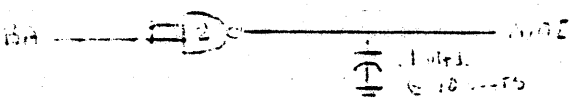
[illegible]

JB/JB

[illegible]

NOT A 6809 BUT CONSIDER THIS

I have managed to use the 3E instruction (WAI) to trigger an NMI. Whenever the 3E is executing, the BA pin of the 6800 goes high. It would be simple to invert this and generate a negative going edge to work the NMI except for the fact that the BA line also goes high on every refresh. The problem is to generate the interrupt only when BA stays high./



The 3E is being used with a supervisor program which does all kinds of nice things (many 16 bit manipulations) all on the stack. In other words the following pseudo instructions execute on my system without using any temporary locations in memory.

Push X onto stack
Pull X from stack
Transfer D accumulator to X register
Transfer X register to D accumulator
Swap D accumulator and X register
Swap X and stack ptr.
swap D and stack ptr.
Swap bytes of index register
Add X to contents of D
Add to X contents of stack ptr.
Add to D contents of X
Add to D contents of stack ptr.
Add to stack ptr. contents of X
Add to stack ptr. contents of D
Push X (MSB) onto stack
Push X (LSB) onto stack
Pull x (MSB) from stack
Pull X (LSB) from stack
Transfer X (LSB) to A acc.
Transfer X (MSB) to A acc.
Transfer X (1SB) to B acc.
Transfer X (MSB) to B acc.
Transfer A acc. to X (MSB)
Transfer A acc. to X (LSB)
Transfer B acc. to X (MSB)
Transfer B acc. to X (LSB)
Swap A and B accumulators
Swap A acc. and stack
Swap B acc. and stack
Swap A acc. and X (MSB)
Swap A acc. and X (LSB)
Swap B acc. and X (MSB)
Swap B acc. and X (LSB)
Add to A acc. contents of stk
add to B acc. contents of stk
Add to stk contents of A acc.
Add to stk contents of B acc.
Add to X(LSB) contents of A
Add to X(LSB) contents of B
Complement X Reg.
Swap nibbles of A acc.
Swap nibbles of B acc.
Mult. accums: result in AB
Mult. X by D: result in XD
Branch 16 bit PC relative
Pushall registers
Pullall registers
Move program block
Compare strings for match
Move message block
Index(mult.AxB and add to X)
Subtract X from AB
Subtract AB from X
Subtract A from X
Subtract B from X
16 bit unsigned divide
Hex to Ascii (one byte)
Ascii to binary (one byte)
Binary 16 bits to 5 Ascii
 digits pointed to by X

THESE ARE CALLED BY A PROGRAM AS 2 BYTE INSTRUCTIONS:

TO PUSH X	3E 01	
TO PULL X	3E 02	Etc.

Our CSS Basic will execute machine language on a line of the Basic program so these short cuts can greatly speed up the Basic/Machine code hybrid program.

Yes, I had to change the vector for NMI in the 1702 EPROM to an address where the supervisor resides but the SWI was left alone. The NMI still works from the keyboard if desired.

Jeff

SPHERE USER GROUP OCTOBER 1979

1	79	ASI MARKET RESEARCH	7655 SUNSET BLVD.	HOLLYWOOD CA.90046
3	79	ROBERT BAER	921 LINCOLN AVE.	PALO ALTO CA.94301
5	79	T.H. ANDERSON	215 BAYVIEW #110	SAN RAFAEL CA. 94901
6	79	BITS AN PIECEZ	P.O. BOX 23	WATERLOO 2017 AUSTRALIA
7	79	JEAN-FRANCOIS BOIVIN	1405 RUE IBERVILLE	MONTREAL CANADA H2K 3B2
9	79	SAM BRONSTEIN	9026 AUTOVILLE DR.	COLLEGE PARK MD. 20740
10	79	LANI BUCELLI	11941 WEIR CT.	CULVER CITY CA. 90230
11	79	CHARLES BURTON PHD.	1618 MARILYN AVE.	DAYTON OH.45420
13	79	PHILLIP CAMERON	43 OLIVE CT.	STOUGHTON MA.02072
16	79	TOM CROSLEY	1675 NEW BRUNSWICK AVE.	SUNNYVALE CA. 94087
18	79	SCOTT D'AMRON	2123 SUFFIELD DR.	WINTER PARK FL.32789
19	79	DARREL COLLINS	8638 E.SOLANDO DR.	SCOTTSDALE AZ. 85253
21	79	OWEN DAVIS	710 WAIKIKI DR.	DES PLAINES IL.60016
22	79	LEROY DANNER	3900(3900-45TH STS.)#23	KENOSHA WI. 53140
23	79	DAVID DEMOREST	12697 GRATON RD.	SEBASTOPOL CA. 95472
24	79	JOSEPH DAWES	2510 BROADWAY	BIG SPRING TX. 79720
25	79	JOHN DEPPE	1201 2ND ST.	DELANCO NJ. 08075
26	79	DONALD DORSON	GARDNER RD.	WEST KINGSTON RI. 02892
27	79	CARLYLE EASTMAN	6016 N.ARLINGTON	SAN PABLO CA. 94806
28	79	R.S. DOWNS	ROUTE 7 BOX 211-A	RALEIGH NC. 27614
29	79	ROBERT ENNIS	9322 LAUREL	FONTANA CA. 92335
31	79	HARRY FRIEDMAN	945 DUDLEY DR.	SHREVEPORT LA. 71104
33	79	DR. JOHN GEARHART	10430 GRAND PK. DR.	SAN ANTONIO TX. 78239
34	79	DAVID GHERSON	1745 RAVIZZA AVE.	SANTA CLARA CA. 95051
35	79	JOHN GIBBON	3 PUDDINGSTONE RD.	NORRIS PLAINS NJ. 07950
37	79	R.M. GRAINGER	RR NO.1 PRESCOTT	ONTARIO CANADA KOE 1TO
39	79	BYRON HALE	1190 GLENBLAIR WY.	CAMPBELL CA. 95008
40	79	G.K. HALE-LONG ENGINEERING	961 BURKE ST.	WINSTON-SALEM NC. 27101
41	79	WILLIAM HARTWEG	228 ST.MARKS PLACE	STATEN ISLAND NY. 10301
42	79	JOHN IRSIK	1017 MICHIGAN	BEAUMONT CA. 92223
45	79	MICHAEL KELLEHER	11004 27TH AVE. S.	SAVAGE MN. 55337
46	79	JOHN HEACOCK	4 STANFORD DR. #3A	BRIDGEWATER NJ. 08807
48	79	KIM KLAGES	435 SANTIAGO AVE.	ORLANDO FL. 32807
49	79	KLEPFER	76 LORNA LANE	TONAWANDA NY. 14150
51	79	DEAN LANCTON	207 S 2ND. ST.	CORNELL IL. 61319
52	79	G.H. LATTA	RT. 3 SMYRNA RD.	SEARCY AR. 72143
53	79	MICHAEL KOVIS	3810 MAIN STREET	STRATFORD CT. 06497
54	79	DAVID LAKE	243 W.SIRIUS	ANAHEIM CA. 92802
60	79	TOM MEIER	80 ALCOTT STREET	ACTON MA. 01720
62	79	JON MEIER	97 CENTER AVENUE	MIDDLETOWN RI. 02840
64	79	E.HUGH MELTON JR.	8314 UNIVERSITY DRIVE	RICHMOND VA. 23229
66	79	JEFFREY BROWNSTEIN D.D.S	2 TOR ROAD	WAPPINGERS NY. 12590
70	79	MELVIN NORELL-PROGRAMMA CONSULTA	P.O. BOX 70127	LOS ANGELES CA. 90070
71	79	MICHAEL MURPHY	2003 LAWRENCE AVE.	INDIANAPOLIS IN.46227
72	79	J.C. PIRTLE	P.O. BOX 537	AZLE TX. 76020
73	79	LOUIS NYERGES	25497 WOLF ROAD	BAY RIDGE OH. 44140
75	79	ARIE POLDERVAART	P.O. BOX 701	YERINGTON NV. 89447
76	79	JIM RAEHL	943 BEGONIA	ESCONDIDO CA. 92027
77	79	H.C. PURCELL	1634 STANFORD DRIVE	ANKORAGE AK. 99504
78	79	JOHN RIBLE	51 DAVENPORT ST.	CAMBRIDGE MA. 02141
79	79	WARREN REDDEN	RR 1 BOX 22	GYPSUM KS. 67448
80	79	W.J. RUTLEDGE	1201 PIERCE ST APT#305	ARLINGTON VA. 22209
81	79	MIKE SCHWARTZ	719 PATTERSON ST.WEST	LONG BEACH CA. 90806
82	79	LAWRENCE SAMBUCO	22 FREDRICK DR.	POUGHKEEPSIE NY. 12603
83	79	DR. ROGER J. SPOTT	13975 CONNECTICUT AVE.	WHEATON MD. 20906
85	79	FRANK VAUGHT	4996 SOMAN AVE.	SAN DIEGO CA. 92110
86	79	RICHARD THERIAULT	12120 'ARCHVEQUE	MONTREAL CANADA H1H 3C1
87	79	GENE WALLIS	1954R OLD MIDDLEFIELD WAY	MOUNTAIN VIEW CA. 94043
88	79	RONALD WALTNER	353 N.KENYON	INDIANAPOLIS IN. 46219
89	79	CHAN WAI YUNG	P.O. BOX K-2296	KOWLOON HONG KONG
90	79	LES J. FULLER	28 KIDMAN AVE.	S.GUILDFORD 6055 W.AUSTRALIA
91	79	WARREN WEIMER	23025 KINARD AVENUE	CARSON CA. 90745
92	79	DR. HARRY SPAIN	882 BLUE RIDGE RD. HOLIDAY PARK	PITTSBURGH PA. 15239
93	79	DAVE SIBLEY-DEPT. OF MATH	PENN STATE UNIVERSITY	UNIVERSITY PARK PA. 16802
94	79	JOHN MCLACHLAN	175 BRENTCLIFF RD.	TORONTO CANADA M4G 3Z1
96	79	DOUG CALLEY	RT1 BOX 219	FLAGSTAFF AZ. 86001
97	79	ROBERT LEIF	1030 MARIPOSA AVE.	CORAL GABLES FL. 33146
98	79	RONALD FISCHER	LPO 11247 LIVINGSTON COLLEGE	NEW BRUNSWICK NJ. 08903
99	79	JIM CHIN	33-34 70TH ST.	JACKSON HEIGHTS NY. 11372
101	79	SCHOLLS INC.-1020 RAYMOND AVE.	P.O. BOX 43116	ST PAUL MN. 55164
102	79	BILL RUTHERFORD	3201 N.450 W.	KOKOMO IN.46979
103	79	JOE D.TREGEAGLE B-35638	CTF-CENTRAL/F-336	SOLEDAD CA. 93960
106	79	DR.GEORGE HORNER	80 DELAMERE AVE.	STRATFORD ONT.CANADA N5A 425
108	79	STEPHEN BRADLEY	W/C 655 BLDG. K-490	NAS MIRAMAR CA. 92145
110	79	JOHN BAYLIS	P.O. BOX 137	NSL UTAH 84054
113	79	HARRIS G. ALLEN	1106 WYNBROOK LANE	MECHANICSVILLE VA. 23111