

SECTION 4

OPERATING, TESTING AND DEBUGGING PROGRAMS

	REFERENCE NUMBER	PROGRAM
1.	4-4 Paper Tape Edit Routine
2.	4-5 IOMEC Series 3 Cartrtape to Intel MCS-4
3.	4-7 Parity Checker/Generator
4.	4-8 Parity Generator, ASCII Character
5.	4-9 Delay Subroutines
6.	4-11 High Speed Printer Interface
7.	4-12 TOUCHTONE® Keyboard Scanner
8.	40-1 Bowmar TP 3100 Printer Routine
9.	40-2 TP 3100
10.	40-3 I/O Test
11.	40-4 8 Digit Register Display
12.	40-5 Mod 40/Silent 700 Interface
13.	40-6 PROM Utility Dump Program
14.	40-7 Pro Forma
15.	4-21 Peripheral Interface Routine for a Thermal Strip Printer
16.	40-8 MCS-4/40 Dissembler
17.	40-9 Right Jusitified Hexadecimal Data Shifter



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-4☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

Paper Tape Edit

Function

Permit operator to edit a program in paper tape. Lines may be added, corrected, or deleted to generate a new tape with having to manually control or jog the tape reader.
See attached writeup.

Required
Hardware

ASR 33 teletype
Switch inputs on ports 1, 2, 3

Required
Software

Input
Parameters

TTY Reader or Keyboard

Output
Results

TTY Printer/Punch
Character read displayed on ports 2, 3.

Registers Modified: ALL	Maximum Subroutine Nesting Level: 2
RAM Required: None for storage 1 for RAM output port	Assembler/Compiler Used: SIM4 Hardware Assembler
ROM Required: 233 locations, on 1 - PROM	Programmer: P. B. Close
	Company: HYDE PARK ELEC., INC. Dayton, Ohio 45440

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer and company
2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps should **not** be included.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

Paper Tape Edit Program

To edit programs punched in paper tape, it is necessary to read the incorrect tape, and to add, correct, or delete lines to generate a new tape. This program operates in conjunction with an ASR 33 Teletype machine to edit tape, without ever having to jog the tape reader. Since tape reading halts at the start of a new line, an incorrect line or lines may be skipped on the old tape, and new lines added as required. One other feature is the ability to read and punch tape until a preselected character is reached, and then halt.

To run the program, the options, or modes, are read in through ROM port 1, where one of four bits is set to a one or true (= 0 volts). If all are zero, the program will wait until one bit is set to a 1. The options are as follows:

- Bit 0 ON: Read paper tape and echo print/punch as long as this bit is on. If it is turned off, halt after reading and printing a line feed, and then check the other mode bits.
- Bit 1 ON: Momentarily turning on this bit will permit the operator to enter one line into the keyboard. Entry terminates after a line feed has been entered and Bit 1 is off. When off, check the other mode bits.
- Bit 2 ON: Momentarily turning on this bit will cause the reader to advance one line on the tape without printing or punching. Reading halts after the reading of a line feed character. Other mode bits are then checked.
- Bit 3 ON: Momentarily turning on this bit will initiate tape reading and printing/punching, which will continue until the detection and printing of a specific character (and bit 3 is off). The code for the character is set into the switches for ports 2 and 3, where port 2 is the MSW. For example, characters such as ?, !, *, etc., might be placed in the tape (comment area) to mark the end of subroutines, or break points.

As each character is read from tape or entered into the keyboard, the code is sent out on ports 2 and 3. The program takes one PROM.

9-30-74
P. B. Close
HYDE PARK ELECTRONICS, INC.
Dayton, Ohio 45440

6-27-74

Teletype Input Codes - Even Parity

<u>Input Character</u>	<u>Port 2</u>	<u>Port 3</u>
Ø	0011	0000
1	1011	0001
2	1011	0010
3	0011	0011
4	1011	0100
5	0011	0101
6	0011	0110
7	1011	0111
8	1011	1000
9	0011	1001
A	0100	0001
B	0100	0010
C	1100	0011
D	0100	0100
E	1100	0101
F	1100	0110
G	0100	0111
H	0100	1000
I	1100	1001
J	1100	1010
K	0100	1011
L	1100	1100
M	0100	1101
N	0100	1110
O	1100	1111
P	0101	0000
Q	1101	0001
R	1101	0010
S	0101	0011
T	1101	0100
U	0101	0101
V	0101	0110
W	1101	0111
X	1101	1000
Y	0101	1001
Z	0101	1010
;	1011	1011
,	1010	1100
.	0010	1110
/	1010	1111

Input Character	Port 2	Port 3
?	0011	1111
:	0011	1010
- (dash)	0010	1101
^ (over N)	1101	1110
!	0010	0001
"	0010	0010
#	1010	0011
\$	0010	0100
%	1010	0101
&	1010	0110
' (over 7)	0010	0111
(0010	1000
)	1010	1001
*	1010	1010
=	1011	1101
_ (under-score)	0101	1111
Return	1000	1101
Line Feed	0000	1010
Bell	1000	0111
[(over K)	1101	1011
\	0101	1100
+	0010	1011
]	1101	1101
<	0011	1100
>	1011	1110
"Here is" (Sprocket only)	0000	0000
Delete	1111	1111
Space	1010	0000

HYL - RADN TEL - 45440
45440 - 1011 E
DAYTON, OHIO 45440

PBC NOV 22 1974

/TAPE EDIT PROGRAM REVISED 6-30-74

0: NOP
1: NOP
2: FIM 0P 0
4: SRC 0P
5: LDM 1
6: WMP
7: FIM 6P 16
9:MSA SRC 6P
10: CLB
11: RDR
12: RAR
13: JOC MS1
15: RAR
16: JOC MS2
18: RAR
19: JOC MS3
21: RAR
22: JOC MS4
24: JUN MSA
26:MS1 JMS TEN
28: FIM 5P 10
30: JMS CFE
32: JNZ MS1
34: JUN MSA
36:MS2 JMS ST
38: FIM 5P 10
40: JMS CFE
42: JNZ MS2
44: JUN MSA
46:MS3 JMS TNP
48: FIM 5P 10
50: JMS CFE
52: JNZ MS3
54: JUN MSA
56:MS4 FIM 4P 32
58: SRC 4P
59: RDR
60: XCH 10
61: INC 8
62: SRC 4P
63: RDR
64: XCH 11
65:MS5 JMS TEN

67: JMS CFE
69: JNZ MS5
71: JUN MSA
73:CFE CLB
74: LD 3
75: SUB 11
76: JNZ CF1
78: CMC
79: LD 2
80: SUB 10
81: JNZ CF1
83: BBL 0
84:CF1 BBL 1
85:TEN FIM 2P 64
87: LDM 1
88: SRC 2P
89: WMP
90:ST CLB
91: FIM 2P 64
93: SRC 2P
94:ST2 JTZ ST2
96: WMP
97: JMS SBR1
99: FIM 0P 13
101:TEST ISZ 1 TEST
103: SRC 0P
104: RDR
105: CMA
106: WMP
107: JMS SBR2
109: FIM 0P 0
111: LDM 0
112: XCH 2
113: LDM 0
114: XCH 3
115: LDM 8
116: XCH 4
117:ST1 JMS SBR1
119: CLC
120: SRC 0P
121: RDR
122: CMA
123: WMP
124: RAR
125: LD 2
126: RAR
127: XCH 2
128: LD 3
129: RAR
130: XCH 3
131: JMS SBR2
133: ISZ 4 ST1
135: JMS SBR1
137: LDM 1
138: FIM 0P 0
140: SRC 0P
141: WMP
142: JMS SBR2
144: NOP
145:END FIM 7P 32
147: SRC 7P
148: LD 2

149: WRR
150: INC 14
151: SRC 7P
152: LD 3
153: WRR
154: BBL 0
155: SBR1 FIM 0P 0
157: L1 ISZ 0 L1
159: ISZ 1 L1
161: BBL 0
162: SBR2 FIM 0P 8
164: L2 ISZ 0 L2
166: ISZ 1 L2
168: BBL 0
169: TNP FIM 2P 64
171: LDM 1
172: SRC 2P
173: WMP
174: CLB
175: FIM 2P 64
177: SRC 2P
178: TT2 JTZ TT2
180: WMP
181: JMS SBR1
183: FIM 0P 13
185: TT3 ISZ 1 TT3
187: SRC 0P
188: RDR
189: CMA
190: NOP
191: JMS SBR2
193: FIM 0P 0
195: LDM 0
196: XCH 2
197: LDM 0
198: XCH 3
199: LDM 8
200: XCH 4
201: TT1 JMS SBR1
203: CLC
204: SRC 0P
205: RDR
206: CMA
207: NOP
208: RAR
209: LD 2
210: RAR
211: XCH 2
212: LD 3
213: RAR
214: XCH 3
215: JMS SBR2
217: ISZ 4 TT1
219: JMS SBR1
221: LDM 1
222: FIM 0P 0
224: SRC 0P
225: WMP
226: JMS SBR2
228: NOP
229: JON END
231: \$

0: BPPPPPPPPF 1: BPPPPPPPPF 2: BPPNPPPPPF BPPPPPPPPF
4: BPPNPPPPNF 5: BNNPNPPPPF 6: BNNNPPPPNF 7: BPPNPNPPPF
BPPNPPPPF 9: BPPNPNPPNF 10: BNNNNPPPPF 11: BNNNPNPPNF
12: BNNNNPNPPF 13: BPPPNPPPPF BPPNPNPPF 15: BNNNNPNPPF
16: BPPNPPPPNF BPPNPPPPF 18: BNNNNPNPPF 19: BPPNPPPPF
BPPNPNPPNF 21: BNNNNPNPPF 22: BPPNPPPPF BPPNPNPPF
24: BPNPPPPPPF BPPPNPPNF 26: BPNPNPPPPF BPNPNPNPF
28: BPPNPNPPNF BPPPNPNPF 30: BPNPNPPPPF BPNPNPNPF
32: BPPNPNPPNF BPPPNPNPF 34: BPNPPPPPPF BPPPNPNPF
36: BPNPNPPPPF BPNPNPNPF 38: BPPNPNPPNF BPPPNPNPF
40: BPNPNPPPPF BPNPNPNPF 42: BPPNPNPPF BPNPNPNPF
44: BPNPPPPPPF BPPPNPNPF 46: BPNPNPPPPF BPNPNPNPF
48: BPPNPNPPNF BPPPNPNPF 50: BPNPNPPPPF BPNPNPNPF
52: BPPNPNPPNF BPPNPNPPF 54: BPNPPPPPPF BPPPNPNPF
56: BPPNPNPPPF BPPNPPPPF 58: BPPNPNPPNF 59: BNNNPNPNPF
60: BPNNNPNPPF 61: BPNPNPPPPF 62: BPPNPNPPNF 63: BNNNPNPNPF
64: BPNNNPNPNF 65: BPNPNPPPPF BPNPNPNPF 67: BPNPNPPPPF
BPNPNPPNF 69: BPPNPNPPF BPNPPPPNF 71: BPNPPPPPF
BPPPNPNPF 73: BNNNNPPPPF 74: BPNPNPPPNF 75: BPNPNPNPF
76: BPPNPNPPF BPNPNPNPF 78: BNNNNPNPNF 79: BPNPNPPNF
80: BPNPNPNPF 81: BPPNPNPPF BPNPNPNPF 83: BNNPPPPPF
84: BNNPPPPNF 85: BPPNPNPPF BPNPPPPPF 87: BNNPNPPNF
88: BPPNPNPNF 89: BNNNPPPPNF 90: BNNNNPPPPF 91: BPPNPNPPF
BPNPPPPPF 93: BPPNPNPNF 94: BPPNPPPNF BPNNNNNPF
96: BNNNPPPPNF 97: BPNPNPPPPF BPNPNPNPF 99: BPPNPPPPF
BPPPNPNPF 101: BPNNNPPPNF BPNPNPNPF 103: BPPNPPPNF
104: BNNNPNPNPF 105: BNNNNPNPPF 106: BNNNPPPPNF 107: BPNPNPPPF
BPNPNPPNF 109: BPPNPPPPF BPPPPPPPF 111: BNNPNPPPF
112: BPNPNPNPF 113: BNNPNPPPF 114: BPNPNPNNF 115: BNNPNPNPF
116: BPNPNPNPF 117: BPNPNPPPF BPNPNPNNF 119: BNNNNPPPNF
120: BPPNPPPNF 121: BNNNPNPNF 122: BNNNNPNPPF 123: BNNNPPPNF
124: BNNNNPNPNF 125: BPNPNPPNF 126: BNNNNPNPNF 127: BPNPNPNPF
128: BPNPNPPNF 129: BNNNNPNPNF 130: BPNPNPNNF 131: BPNPNPPPF
BPNPNPNPF 133: BPNNNPNPPF BPNNNPNPNF 135: BPNPNPPPF
BPNPNPNNF 137: BNNPNPPNF 138: BPPNPPPPF BPPPPPPPF
140: BPPNPPPNF 141: BNNNPPPNF 142: BPNPNPPPF BPNPNPNPF
144: BPPPPPPPF 145: BPPNPNNNF BPPNPPPPF 147: BPPNPNNNF
148: BPNPNPPNF 149: BNNNPPPNF 150: BPNPNNNPF 151: BPPNPNNNF
152: BPNPNPPNF 153: BNNNPPPNF 154: BNNPPPPPF 155: BPPNPPPPF
BPPPPPPPF 157: BPNNNPPPF BPNNNPNPF 159: BPNNNPPNF
BPNNNPNPF 161: BNNPPPPPF 162: BPPNPPPPF BPPPNPPPF
164: BPNNNPPPF BPNPNPNPF 166: BPNNNPPNF BPNPNPNPF
168: BNNPPPPPF 169: BPPNPNPPF BPNPPPPPF 171: BNNPNPPNF
172: BPPNPNPNF 173: BNNNPPPNF 174: BNNNNPPPF 175: BPPNPNPPF
BPNPPPPPF 177: BPPNPNPNF 178: BPPNPPPNF BPNPNPNPF
180: BNNNPPPNF 181: BPNPNPPPF BPNPNPNNF 183: BPPNPPPPF
BPPPNPNPF 185: BPNNNPPNF BPNNNPNPF 187: BPPNPPPNF
188: BNNNPNPNF 189: BNNNNPNPPF 190: BPPPPPPPF 191: BPNPNPPPF
BPNPNPNPF 193: BPPNPPPPF BPPPPPPPF 195: BNNPNPPPF
196: BPNPNPNPF 197: BNNPNPPPF 198: BPNPNPNNF 199: BNNPNPNPF
200: BPNPNPNPF 201: BPNPNPPPF BPNPNPNNF 203: BNNNNPPNF
204: BPPNPPPNF 205: BNNNPNPNF 206: BNNNNPNPPF 207: BPPPPPPPF
208: BNNNNPNPNF 209: BPNPNPPNF 210: BNNNNPNPNF 211: BPNPNPNPF
212: BPNPNPPNF 213: BNNNNPNPNF 214: BPNPNPNNF 215: BPNPNPPPF
BPNPNPNPF 217: BPNNNPNPF BPNPNPNF 219: BPNPNPPPF
BPNPNPNNF 221: BNNPNPPNF 222: BPPNPPPPF BPPPPPPPF
224: BPPNPPPNF 225: BNNNPPPNF 226: BPNPNPPPF BPNPNPNPF
228: BPPPPPPPF 229: BPNPPPPPF BPNPNPNF 231:



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-5☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

IOMEC SERIES THREE (S-3) CARTRITAPE TO INTEL Mcs-4

Function

This package of firm ware routines allow full control of the Iomec S-3 by the MCS-4.

Required
Hardware

Reference FIG #1 & FIG #2

Required
Software

EXECUTIVE

Input
Parameters

SRWAL	SRDEL
SRTWD	SRDL1
SRTRD	SRWEF
SRBOT	SRRWD
SREOT	SRBSK
SRTRE	SREOF

Output
Results

SEE INDIVIDUAL ROUTINES

Registers Modified:	Maximum Subroutine Nesting Level:
RAM Required:	Assembler/Compiler Used: 2
ROM Required: 256 BYTES	Programmer: CARL R. BOEHME
	Company: IOMEC



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

SRTRE

Function

This routine samples the CRC-Error (TRE) status of the S-3. The TRE status is returned via R4.

C(R4) = 15 A Tape Read Error was detected.
C(R4) = 0 No Tape Read Error detected.

Required
Hardware

Required
Software

Input
Parameters

P6 ← 0

Output
Results

P6 = 0 Selected
C ← TRE
R4 ← TRE
MuX(1,0)

Registers Modified:	Maximum Subroutine Nesting Level: 1
RAM Required:	Assembler/Compiler Used: TYMSHARE V2.2
ROM Required:	Programmer: CARL R. BOEHME
	Company: IOMEC



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

SRSBK

Function

This routine causes the tape to backspace one record

SPECIAL CASES

1. BOT

A. Tape at BOT - operation is never initiated

B. BOT detected - operation is terminated

Required
Hardware

2. File Mark (FM) - During the back space operation a
FM is treated as a record.

Required
Software

SRBOT

SRDL1

SRDEL

Input
Parameters

Output
Results

C(C) = BOT c(C) = 1
P7←0 P1←200 Selected
P6←0 A←0
P5←U/D
P1←200
A←0

Registers Modified:	Maximum Subroutine Nesting Level: 2
RAM Required:	Assembler/Compiler Used: TYMSHARE V2.2
ROM Required:	Programmer: CARL R. BOEHME
	Company: IOMEC



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

☒ 8080 ☐ 8085 ☐ 8088

(use additional sheets if necessary)

Program
Title

SRRWD

Function

This routine causes the series three to rewind the tape to BOT at high speed (21IPs).

Required
Hardware

Required
Software

SRBOT
SRDEL

Input
Parameters

P7←0
P6←0
P5←U/D
P1←40Ø
C←BOT
A←0
Mux (1,0)

Output
Results

Registers Modified:	Maximum Subroutine Nesting Level: 2
RAM Required:	Assembler/Compiler Used: TYMSHARE V2.2
ROM Required:	Programmer: CARL R. BOEHME
	Company: IOMEC



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

SRWAL

Function

This routine samples the file protected switch (WALO) on the S-3. The WALO status is returned via the C bit.

c(C) = 1 FILE PROTECTED

c(C) = 0 FILE UNPROTECTED; OK TO WRITE.

Required
Hardware

Required
Software

Input
Parameters

Output
Results

P7←0 Selected

A←0

C←WALO'

Registers Modified:	Maximum Subroutine Nesting Level: 1
RAM Required:	Assembler/Compiler Used: TYMSHARE V2.2
ROM Required: 2	Programmer: CARL R. BOEHME
	Company: IOMEC



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

SRDEL
SRDL1

Function

A general purpose timer routine which is used to effect delays in the normal instruction stream. SRDL1 Provides a 89.55 MSEC delay.

Required
Hardware

Required
Software

The 1's compliment of the desired delay time is passed to the subroutine Via P7 (MSB) and P6 (LSB).

Input
Parameters

The minimum delay increment is 84.80 MSEC. The maximum delay which may be caused, with one JMS, is 5.535 seconds.

When large delays are required it is often possible to initialize only P7. The average resultant error is 10.85 sec. The maximum is 21.708 MSEC and the minimum is zero.

Additional processing time is absorbed by the BBL instruction.

Output
Results

P7←0
P6←0
P5←U/D
A←0

Registers Modified:	Maximum Subroutine Nesting Level: 1
RAM Required:	Assembler/Compiler Used: TYMSHARE V2.2
ROM Required:	Programmer: CARL R. BOEHME
	Company: IOMEC



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

SREOF

Function

This routine causes the S-3 to search at high speed (21IPS), for the next file mark (FM). Detection of a FM causes the operation to terminate with the tape head positioned after the FM.

In the event a FM does not occur before EOT, the detection of EOT terminates the operation.

Required
Hardware

Required
Software

SRBOT
SREOT
SRDEL

Input
Parameters

Output
Results

P1←400
P6←0
P5←U/D

C←EOT
A←0
MUX (1,0)

Registers Modified:	Maximum Subroutine Nesting Level:
RAM Required:	Assembler/Compiler Used: TYMSHARE V2.2
ROM Required:	Programmer: CARL R. BOEHME
	Company: IOMEC



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

SREOT

Function

This routine samples the End of Tape (EOT) status of the S-3. The EOT status is returned Via the C bit.

c(C) = 1 EOT was encountered.

c(C) = 0 EOT foil has not been detected.

Required
Hardware

Required
Software

Input
Parameters

Output
Results

P7←0 Selected
C←EOT
MUX (1,0)
A←0

Registers Modified:	Maximum Subroutine Nesting Level: 1
RAM Required:	Assembler/Compiler Used: TYMSHARE V2.2
ROM Required:	Programmer: CARL R. BOEHME
	Company: IOMEC



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

SRTRD

Function

SEE ATTACHED SHEET

Required
Hardware

Iomec Data Formatter and Data Buffer.

Required
Software

SRBOT
SRDL1
SRDEL
SRTRE

Input
Parameters

Output
Results

P7←0
P6←0
P5←U/D
P3←(R4←TRE; R5←FM)
P1←200
P0←(R0←0; R₁←EOT)

Registers Modified:	Maximum Subroutine Nesting Level: 2
RAM Required:	Assembler/Compiler Used: TYMSHARE V2.2
ROM Required:	Programmer: CARL R. BOEHME
	Company: IOMEC

SRTRD CONT.

FUNCTION

This routine is used to recover a record of data from tape.

SPECIAL CASES

1. EOT - EOT status is sensed only if the read operation started with the tape positioned at BOT. If EOT is encountered before data is encountered, (perfectly clean tape), the operation is terminated.
2. FILEMARK - If a FM is encountered the operation is terminated and R5←FM status.

c (R1) = 15 EOT Detected
c (R1) = 0 EOT not detected
c (R5) = 15 FM detected
c (R5) = 0 Record read and FM not detected
c (R4) = 15 Read Error detected
c (R4) = 0 No Read Error detected



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

SRTWD

Function

SEE ATTACHED SHEET

Required
Hardware

Iomec Data formatter, Write Control logic and Data Buffer.

Required
Software

SRDEL
SRDL1
SRBOT
SREOT
SRTRE

Input
Parameters

Output
Results

P7←0
P6←0
P5←U/D
P4←TRE
P1←20Ø
A←0
C←EOT
MUX (1,0)

Registers Modified:	Maximum Subroutine Nesting Level: 2
RAM Required:	Assembler/Compiler Used: TYMSHARE V2.2
ROM Required:	Programmer: CARL R. BOEHME
	Company: IOMEC

SRTWD CONT.

FUNCTION:

This routine is used to write a record of data on tape.

SPECIAL CASES

1. BOT - A section of tape, 9.2" in length, is erased before the recording process is begun. This BOT gap eliminates the possibility of read errors occurring in the vicinity of the BOT splice.

2. READ AFTER WRITE

If the S-3 includes Read after Write capability R4 will contain the valid Write Error status.

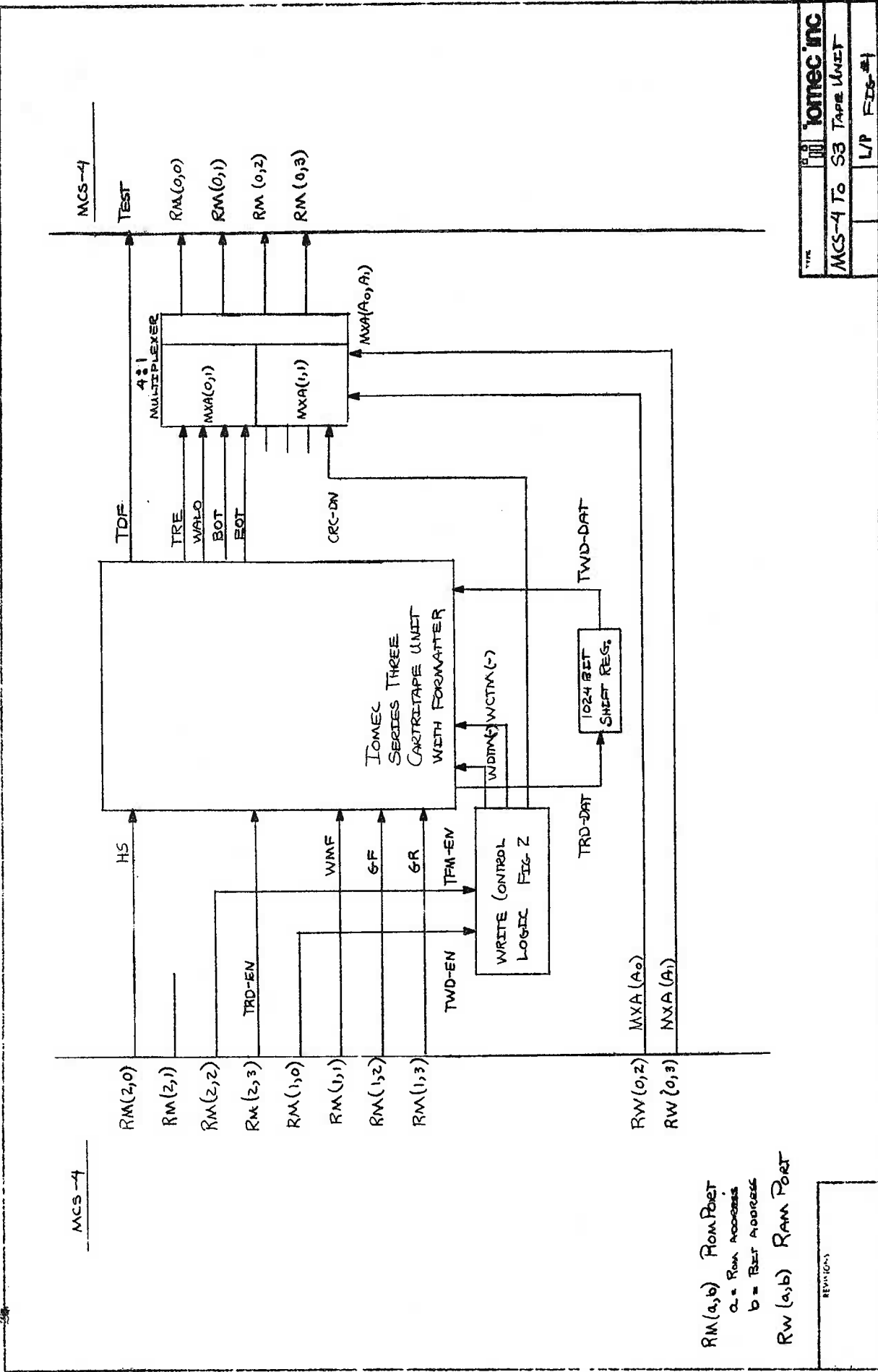
3. EOT - EOT status is returned Via the C bit.

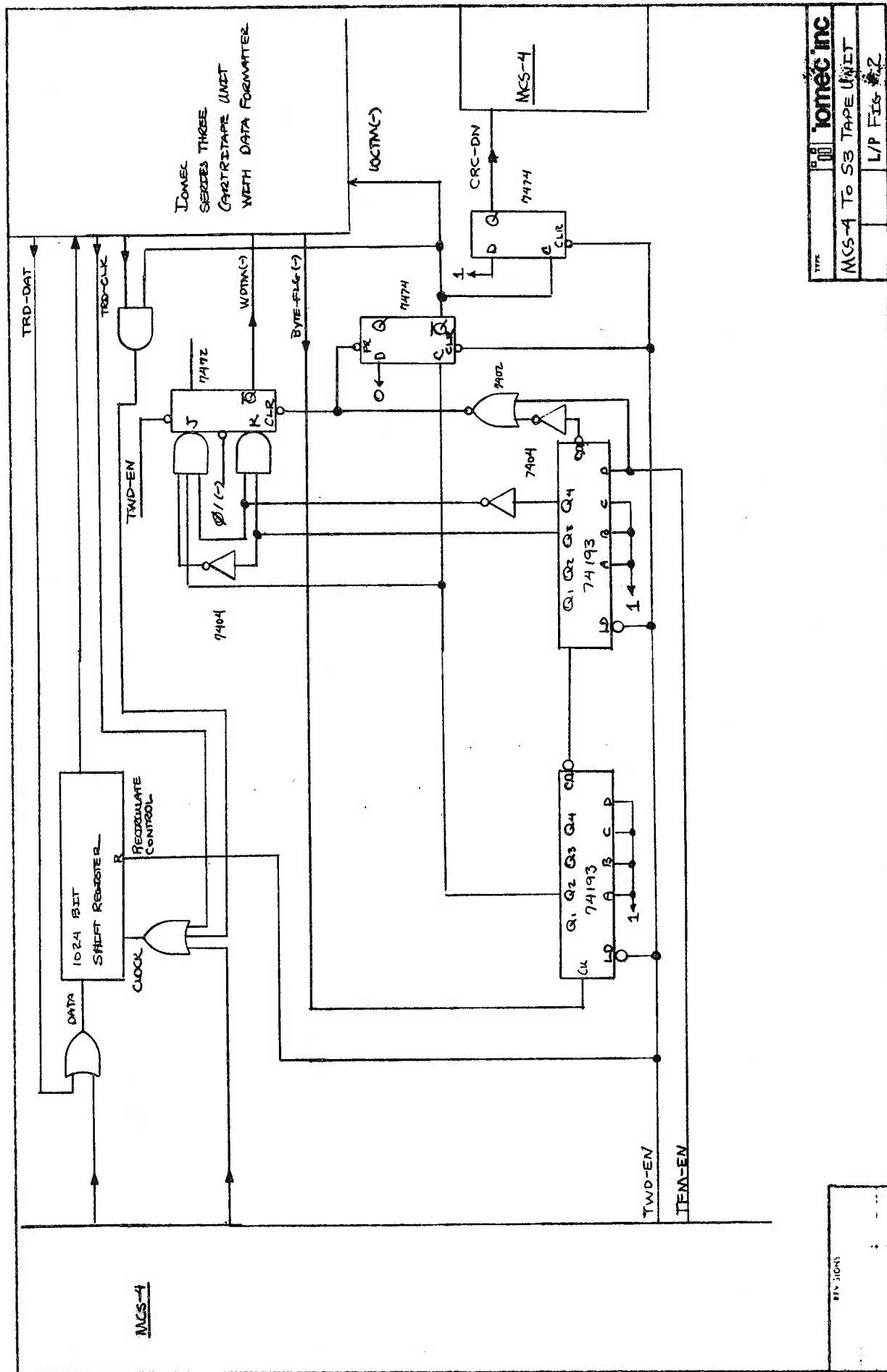
c (R4) = 15 Write Error detected
c (R4) = 0 No Write Error detected
c (C) = 1 EOT detected
c (C) = 0 EOT not detected

CONSIDERATIONS

1. Upon detection of EOT it is good practice to consider the tape full.
2. Editing of records should not be allowed. If editing is required consider all of the records after the edited record unrecoverable.

BOT	True when the tape is positioned at the Beginning Of Tape.
CRC-DN	True after the Cyclic Redundancy Check (CRC) characters have been written on tape.
EOT	True when the End Of Tape foil is detected.
GF	Go Forward; When true, moves tape in the forward direction.
GR	Go Reverse; When true, moves tape in the reverse direction.
HS	High Speed; When true, primes the tape unit to move tape at 21 ips instead of 7 ips. May be used in conjunction with GF or GR.
MXA(Ao,A1)	Multiplexer Address Lines; selects one of 4 possible sources of data for Rom Port 0. Syn MUX (Ao,A1)
TDF	Tape Data Field; true when the tape head is sensing a flux reversal.
TFM-EN	Tape File Mark Enable; causes A 16 bit File mark to be written on tape.
TRD-CLK	Tape Read Clock; read clock generated by the formatter and used to shift the read data into the 1024 bit Buffer Register.
TRD-DAT	Tape Read Data; data recovered from tape during a read operation.
TRD-EN	Tape Read Enable; when true enables the read data decoders
TRE	Tape Read Error; true when the contents of the CRC register is not zero.
TWD-CLK	Tape Write Clock; when true requests the next bit of data to be written.
TWD-DAT	Tape Write Data; data to be written on tape.
TWD-EN	Tape Write Enable; when true enables write control logic to write a record or five on tape and enables the Recirculate logic for the Buffer Register.
WALO	Write Allow; true if Write Allow pin is installed in tape cartridge.
WCTM	Write CRC Time; when true writes the contents of the CRC feed back shift register on tape.
WDTM	Write Data Time; When true writes the contents of the 1024 bit Buffer Register on tape.
WMF	Write Mode Function; when true turns on S-3 Write Amplifier.





TYPE	00	IOMEC inc
MCS-4 To S8 TAPE UNIT		
L/P FIB #2		

* 256D

0256 00046 SR01, FIM P7 0
00000
0258 00047 SRC P7
0259 00216 LLM 8
0260 00225 WMP /MUX 1,0
0261 00234 RDR /EOT, WALE,THE
0262 00245 RAL
0263 00192 BBL 0
0264 00032 SRRWD, FIM PQ 400
00032
0266 00033 SRC P0
0267 00209 LLM 1
0268 00226 WRR /HS=1
0269 00081 RWD1, JMS SR001 /C=B0T
00131
0271 00035 SRC P1
0272 00018 JCN C1 RWD2
00022
0274 00216 LDM 8D
0275 00226 WRR /GR=1
0276 00065 JUN RWD1
00013
0278 00046 RWD2, FIM P7 3670
00247
0280 00081 JMS SRDEL /190MS
00071
0282 00226 WRR /GR=0
0283 00192 BBL 0

```

0284 00046 SEWAL, FIM P7 0
      00000
0286 00047 SRC P7
0287 00216 LDM 8D
0288 00225 WMP /MUX 1,0
0289 00234 RDR /E0T,B0T,WAL0',TRE
0290 00246 RAR
0291 00246 RAR
0292 00192 BBL 0 /C+WAL0'
0293 00081 SRSBK, JMS SRB0T
      00131
0295 00018 JCN C1 SBKX0 /IF B0T THEN DONE
      00066
0297 00035 SRC P1
0298 00216 LDM 8D
0299 00226 WRR /GR=1
0300 00081 JMS SRDL1 /80MS
      00067
0302 00025 SBK2, JCN T1 SBK1 /IF TDF THEN SBK1
      00054
0304 00081 JMS SRB0T
      00131
0306 00018 JCN C1 SBKX1 /IF B0T THEN DONE
      00056
0308 00065 JUN SBK2
      00046
0310 00025 SBK1, JCN T1 SBK1 /IF TDF THEN WAIT
      00054
0312 00046 SBKX1, FIM P7 3760
      00254
0314 00044 FIM P6 1770
      00127
0316 00081 JMS SRDEL /30MS
      00071
0318 00035 SRC P1
0319 00226 WRR /GR=0
0320 00081 JMS SRDL1 /80MS
      00067
0322 00192 SBKX0, BBL 0
0323 00046 SRDL1, FIM P7 3730
      00251

```

0325	00044	FIM P6 3376	/86MS
	00223		
0327	00125	SRDEL, ISZ R13 DEL1	
	00080		
0329	00124	ISZ R12 DEL2	
	00081		
0331	00127	ISZ R15 DEL3	
	00082		
0333	00126	ISZ R14 SRDEL	
	00071		
0335	00192	BEL 0	
0336	00058	DEL1, FIN P5	
0337	00058	DEL2, FIN P5	
0338	00065	DEL3, JUN SRDEL	
	00071		
0340	00081	SRTWD, JMS SRB0T	
	00131		
0342	00035	SRC P1	
0343	00214	LDM 6	
0344	00226	WRR /WMP,GF=1	
0345	00026	JCN C0 TWD1 /IF B0T THEN CONT	
	00099		
0347	00081	TWD2, JMS SRB0T	
	00131		
0349	00018	JCN C1 TWD2 /IF B0T THEN WAIT	
	00091		
0351	00046	FIM P7 2770 /1.3S LDR	
	00191		
0353	00081	JMS SRDEL	
	00071		
0355	00046	TWD1, FIM P7 3720	
	00250		
0357	00044	FIM P6 2770	
	00191		
0359	00081	JMS SRDEL /110MS	
	00071		
0361	00035	SRC P1	
0362	00215	LDM 7 /GF,WMP,TWD-EN=1	
0363	00226	WRR	
0364	00220	LDM 12D	
0365	00047	SRC P7	
0366	00225	WMP /MUX=1,1	
0367	00234	TWD3, RDR	
0368	00245	RAL /C=CRC-DN	
0369	00026	JCN C0 TWD3 /IF CRC-DN THEN CONT	
	00111		
0371	00046	FIM P7 3760	
	00254		
0373	00044	FIM P6 1470	
	00103		
0375	00081	JMS SRDEL /33MS	
	00071		
0377	00035	SRC P1	
0378	00210	LDM 2	
0379	00226	WRR /GF,TWD-EN=0	
0380	00081	JMS SRDL1 /80MS	
	00067		

0382	00081	JMS SRTRE	/IF TRE THEN R4=15
	00142		
0384	00081	JMS SREOT	
	00000		
0386	00192	BBL 0	
0387	00034	SRBOT,	FIM P1 200
	00016		
0389	00046	FIM P7 0	
	00000		
0391	00047	SRC P7	
0392	00216	LDM 8D	
0393	00225	WMP	/MUX=1.0
0394	00234	RDR	/EOT,BOT,WALC',TRE
0395	00245	RAL	
0396	00245	RAL	/C=BOT
0397	00192	BBL 0	
0398	00045	SRTRE,	SRC P6 /INSURE P6=0
0399	00216	LDM 8D	/MUX 1.0
0400	00225	WMP	
0401	00234	RDR	
0402	00246	RAR	/C=TRE
0403	00208	LDM 0	/A=0
0404	00180	XCH R4	
0405	00026	JCN CO TREX	/IF TRE THEN CONT
	00153		
0407	00223	LDM 15D	
0408	00180	XCH R4	
0409	00192	TREX,	BBL 0
0410	00032	SRTRE,	FIM P0 0
	00000		
0412	00038	FIM P3 0	
	00000		
0414	00081	JMS SRBOT	
	00131		
0416	00035	SRC P1	
0417	00212	LDM 4	
0418	00226	WRR	/GF=1
0419	00040	FIM P4 400	/P4(RDM 2)
	00032		
0421	00018	JCN C1 TRDL1	/IF BOT' THEN CONT
	00174		
0423	00081	JMS SRDL1	/80NS
	00067		
0425	00041	SRC P4	
0426	00218	LDM 10D	
0427	00226	WRR	/TRD-EN=1
0428	00065	JUN TRD5	
	00198		
0430	00081	TRDL1,	JMS SRBOT /C=BOT
	00131		
0432	00018	JCN C1 TRDL1,	/IF BOT' THEN CONT
	00174		
0434	00046	FIM P7 337L	
	00223		
0436	00081	JMS SRDEL	/0.75
	00071		
0438	00041	SRC P4	

0439	00218	LDM 10D	
0440	00226	WRR	/TRD-EN=1
0441	00045	SRC P6	
0442	00216	LDM 8D	/MUX(1,0)
0443	00225	WMP	
0444	00234	RDR	/E01,E01,WALL',THE
0445	00245	KAL	
0446	00026	JCN CO TRD3	/IF E01 THEN R1=15
	00196		
0448	00223	LDM 15D	
0449	00177	XCH R1	/R1-E01 FLG
0450	00065	JUN TRDX	
	00214		
0452	00017	TRD3, JCN TO TRD1	/IF TDF THEN CON1
	00188		
0454	00017	TRD5, JCN TO TRD5	/IF TDF THEN CONT
	00198		
0456	00046	FIM P7 3770	
	00255		
0458	00044	FIM P6 1770	
	00127		
0460	00081	JMS SRDEL	/10MS
	00071		
0462	00025	JCN T1 TRD2	/IF TDF THEN TRD2
	00212		
0464	00223	LDM 15D	
0465	00181	XCH R5	
0466	00065	JUN TRDX	/FM FLG SET
	00214		
0468	00025	TRD2, JCN T1 TRD2	/IF TDF THEN WAIT
	00212		
0470	00046	TRDX, FIM P7 3770	
	00255		
0472	00081	JMS SRDEL	/23MS
	00071		
0474	00035	SRC P1	
0475	00240	CLB	
0476	00226	WRR	/GF=0
0477	00041	SRC P4	/P4(R0M 2)
0478	00210	LDM 2	
0479	00226	WRR	/TRD EN=0
0480	00081	JMS SRTRE	/R4=1RE
	00142		
0482	00081	JMS SRDL1	/80MS
	00067		
0484	00192	BBL 0	
0485	00034	SEERS, FIM P1 200	
	00016		
0487	00035	SRC P1	
0488	00214	LDM 6	
0489	00226	WRR	/WMP=1, GF=1
0490	00046	FIM P7 3520	
	00234		
0492	00081	JMS SRDEL	/430MS
	00071		
0494	00210	LDM 2	
0495	00226	WRR	/GF=0
0496	00081	JMS SRDL1	/80MS
	00067		
0498	00081	JMS SRE01	
	00000		
0500	00192	BBL 0	



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-7☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title	Parity Checker/Generator
Function	Routine to check or generate parity for 8-bit byte in P0. Utilized modulo-2 counting technique.
Required Hardware	Any MCS-4 system
Required Software	None: Routine for use in larger program
Input Parameters	Any 8-bit byte in P0
Output Results	Carry bit = 0 for even parity Carry bit = 1 for odd parity (use CMC instruction for complement of above) Also R13 indicates # of ones in byte

Registers Modified: R13, R14, R15, ACC, C	Maximum Subroutine Nesting Level: None
RAM Required: None	Assembler/Compiler Used: Intel ASM4 Assembler on G.E. Timesharing
ROM Required: 20	Programmer: J. Parchesky
Execution time: 48 to 56 cycles (depends on data)	Company: Datatype Corporation Miami, Florida

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer and company
2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps should not be included.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

FILE10 05/14/74

ASSEMBLY OF FILE1 ON 05/14/74 AT 10:00EDT

/PARITY CHECKER/GENERATOR

/ROUTINE TO CHECK OR GENERATE PARITY FOR 8 BIT BYTE IN P0

/RESULT: CARRY BIT = 0 FOR EVEN PARITY; = 1 FOR ODD PARITY

/P0 IS UNDISTURBED; R13, R14, R15, AND ACC ARE CHANGED

/

0000 00046 A1,	FIM P7 204	/SET R15 TO -4; R14 TO -4
00204		
0002 00240	CLB	/CLEAR R13
0003 00189	XCH R13	
0004 00161	LD R1 3	/GET LOWER 4 BITS
0005 00246 A2,	RAR	/SHIFT RIGHT
0006 00026	JCN C2 A3	/INCREMENT R13 IF A ONE BIT
00009		
0008 00109	INC R13	
0009 00127 A3,	ISZ R15 A2	/REPEAT 4 TIMES
00005		
0011 00160	LD R0 2	/GET HIGHER 4 BITS
0012 00246 A4,	RAR	/SHIFT RIGHT
0013 00026	JCN C2 A5	/INCREMENT R13 IF A ONE BIT
00016		
0015 00109	INC R13	
0016 00126 A5,	ISZ R14 A4	/REPEAT 4 TIMES
00012		
0018 00173	LD R13	/SHIFT PARITY BIT INTO CARRY
0019 00246	RAR	



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-8☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title	Parity Generator, ASCII Character
Function	Routine to add even parity bit to 7-bit ASCII character in P0. Utilizes modulo-2 counting technique.
Required Hardware	Any MCS-4 system
Required Software	None: Routine for use in larger program
Input Parameters	7-bit ASCII character in P0; MSB of R0 is "don't care"
Output Results	8-bit ASCII character in P0 with MSB of R0 as even parity bit.

Registers Modified: R0, R13, R14, R15, ACC, C	Maximum Subroutine Nesting Level: None
RAM Required: None	Assembler/Compiler Used: Intel ASM4 Assembler on G.E. Timesharing
ROM Required: 25 bytes	Programmer: J. Parchesky
Execution Time: 48-55 cycles (depends on data)	Company: Datatype Corporation Miami, Florida

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer and company
2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps should not be included.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

FILE20

05/14/74

ASSEMBLY OF FILE2 ON 05/14/74 AT 10:10EDT

/PARITY GENERATOR: ASCII CHARACTER

/ROUTINE TO ADD EVEN PARITY BIT TO 7 BIT ASCII CHARACTER

/IN P0. MSB OF P0 (R0) IS A "DON'T CARE".

/RESULT: 8 BIT ASCII CHARACTER IN P0 WITH MSB = EVEN PARITY

/

0000 00046 B1,	FIM P7 220	/SET R15 TO -4; R14 TO -3
00220		
0002 00240	CLB	/CLEAR R13
0003 00189	XCH R13	
0004 00161	LD R1	/GET LOWER 4 BITS
0005 00246 B2,	RAR	/SHIFT RIGHT
0006 00026	JCN C2 B3	/INCREMENT R13 IF A ONE BIT
00009		
0008 00109	INC R13	
0009 00127 B3,	ISZ R15 B2	/REPEAT 4 TIMES
00005		
0011 00160	LD R0	/GET HIGHER 3 BITS
0012 00246 B4,	RAR	/SHIFT RIGHT
0013 00026	JCN C2 B5	/INCREMENT R13 IF A ONE BIT
00016		
0015 00109	INC R13	
0016 00126 B5,	ISZ R14 B4	/REPEAT 3 TIMES
00012		
0018 00246	RAR	/ELIMINATE THE "DON'T CARE"
0019 00176	XCH R0	/TEMP SAVE HIGHER 3 BITS
0020 00173	LD R13	/GET PARITY BIT
0021 00246	RAR	/SHIFT TO CARRY
0022 00160	LD R0	/RETRIEVE HIGHER 3 BITS
0023 00246	RAR	/ROTATE WITH CARRY TO PROPER P0
SITION		
0024 00176	XCH R0	/SAVE IN R0



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-9☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title	Delay Subroutines
Function	To conditionally generate a selectable time delay or <ol style="list-style-type: none">1. 1 thru 256 MS in one MS increments2. 1 thru 256 quarterseconds in one quartersecond increments3. 1 thru 15 minutes in one minute increments
Required Hardware	N/A <i>can only do mins or secs or ms</i>
Required Software	N/A <i>if want mins & secs have to jump to delay routine 2 times</i>
Input Parameters	P7, P5, and R9 hold the two's complement of the number specifying the desired delay in milliseconds, quartersecond, or minutes respectively. The carry is either set or reset depending on the type of delay desired; unconditional or conditional. The test signal input to the processor unit is used to terminate a conditional delay prematurely.
Output Results	The content of index registers R9 thru R15 is altered. The subroutines always return with a zero in the accumulator for an unconditional delay (CY=1). They return with a fifteen in the accumulator for a conditional delay (CY=0) that was terminated prematurely, or with a zero in the accumulator for a conditional delay that timed out completely. <i>I am still not sure whether Reg. used for secs has to be cleared when making to min</i>

Registers Modified: CY, ACC, R9 thru R15	Maximum Subroutine Nesting Level: Two <i>1 - JMS, TE Delay 1 - Internal</i>
RAM Required: None	Assembler/Compiler Used: Tymshare
ROM Required: One	Programmer: Jobst E. Jesse
	Company: IOMEC

I don't think it matters.

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer and company
2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps should **not** be included.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

everywhere you see TEST
substitute carry, because of the
one change that was made.

/DELAY SUBROUTINE

/THIS ROUTINE GENERATES A

/ 1) 1-256 MILLISECOND DELAY:

W/ ARGUMENTS: (P7) = 2'S COMPLIMENT OF THE
NUMBER SPECIFYING THE DESIRED DELAY IN
MILLISECONDS

~~CY = 0~~ CARRY = 1
~~AND TEST = 1~~ FOR EARLY EXIT TO
BE ACTIVE

W/ RETURN: (ACC), (P6), (P7) ALTERED

W/ CALL: JMS MSDLY

/DELAY = ((92) X (2'S COMPL. OF (P7)) + 5) X (10.8)
MICROSECONDS

/TWO SUBROUTINE ADDRESS LEVELS REMAINING

/ 2) 1/4-64 SECONDS DELAY:

W/ ARGUMENTS: (P5) = 2'S COMPLIMENT OF THE
NUMBER SPECIFYING THE DESIRED DELAY IN
QUARTER SECONDS

~~CY = 0~~ CARRY = 1
~~AND TEST = 1~~ FOR EARLY EXIT TO
BE ACTIVE

W/ RETURN: (ACC), (R9), (P5), (P6), (P7) ALTERED

W/ CALL: JMS QSDLY

/ONE SUBROUTINE ADDRESS LEVEL REMAINING

/ 3) 1-15 MINUTES DELAY:

W/ ARGUMENTS: (R9) = 2'S COMPLIMENT OF THE
NUMBER SPECIFYING THE DESIRED DELAY IN
MINUTES

~~CY = 0~~ C = 1
~~AND TEST = 1~~ FOR EARLY EXIT TO BE
ACTIVE

W/ RETURN: (ACC), (R9), (P5), (P6), (P7), ALTERED

W/ CALL: JMS MIDLY

2.2
2.2
7
2.2
10
2.2
54

/ONE SUBROUTINE ADDRESS LEVEL REMAINING

/-

/NOTE: LESS THAN ONE PERCENT ERROR FOR DELAYS OF MORE
/ THAN TEN MILLISECONDS AND 10.8US INSTRUCTION
/ CYCLES

/-

MIDLY, FIM P5 16

JUN QSDLY+2

QSDLY, LDM 15

XCH R9

FIM P7 5

JMS MSDLY

JCN NZA EXIT

ISZ R11 QSDLY+2

ISZ R10 QSDLY+2

ISZ R9 MIDLY

BBL 0

/TIME OUT

MSDLY, JCN 130 EXIT

RTR12, LDM 13

XCH R12

NCP

NCP

RTR13, LDM 4

XCH R13

ISZ R13 *

ISZ R12 RTR13

ISZ R15 MSDLY

ISZ R14 RTR12

BBL 0

/TIME OUT

EXIT, BBL 15

/EARLY EXIT

/-

/OPTIONS PROVIDED:

/

A. UNCONDITIONAL DELAY:

/

/

/

/

/

/

/

/

/

/

/

/

/

/

/

/

/

/

/

/

/

/

/

/

/

/

/

/

/

/

Minutes Delay (1-15)

Quarter Second Delay (1-256)

Half Second Delay (1-256)

SET THE CARRY (CY=1) BEFORE
ENTERING ANY ONE OF THE THREE TIME DELAY SUBROUTINES.
WHEN THE DESIRED DELAY EXPIRES THE PROGRAM BRANCHES
BACK WITH A ZERO IN THE ACCUMULATOR.

B. CONDITIONAL DELAY:

RESET THE CARRY (CY=0) BEFORE
ENTERING ANY ONE OF THE THREE TIME DELAY SUBROUTINES.
THE PROGRAM BRANCHES BACK EITHER WITH A FIFTEEN IN
THE ACCUMULATOR AS SOON AS THE TEST SIGNAL TO THE
CPU CHIP (4004) IS ACTIVATED, OR WITH A ZERO IN THE
ACCUMULATOR IF THE DESIRED DELAY EXPIRES BEFORE THE
TEST SIGNAL IS ACTIVATED.

NOTE: THE TEST SIGNAL IS ASSUMED TO BE A LEVEL OR A
PULSE OF MORE THAN ONE MILLISECOND DURATION.

```

/ TEST PROGRAM TO ASSURE THE VALIDITY OF THE THREE
/ DELAY SUBROUTINES
/-

```

```

0000 00000 START, NOP
0001 00040 FIM P0 0
      00000
0003 00360 CLB
0004 00041 CLEAR, SRC P0 /CLEAR ALL
0005 00342 WRK /RAM AND ROM
0006 003 WMP /OUTPUT
0007 001 ISZ R0 CLEAR /PRTS
      00004

```

```

/-
0011 00321 LDM 1 /TEST 1: 100MS CONDITIONAL DELAY
0012 00120 JMS DSPLA /((0001) TO RAM1 PORT
      00115

```

```

/-
0014 00140 RPT1, INC R0 /INCREMENT FOUR BIT
0015 00240 LD R0 /BINARY OUTPUT CODE
0016 00341 WMP /AT RAM0 PORT EVERY
0017 00056 FIM P7 156 /100MS, AND
      00234
0021 00120 JMS MSPLY /CONTINUE AS SOON AS
      00151
0023 00342 WRK /"TEST" IS ACTIVATED
0024 00024 JCN AZ RPT1
      00014
0026 00031 JCN TN *
      00026

```

```

/-
0030 00322 LDM 2 /TEST 2: TEN SECONDS COND. DELAY
0031 00120 JMS DSPLA /((0010) TO RAM1 PORT
      00115

```

```

/-
0033 00140 RPT2, INC R0 /INCREMENT FOUR BIT
0034 00240 LD R0 /BINARY OUTPUT CODE
0035 00341 WMP /AT RAM0 PORT EVERY
0036 00052 FIM P5 216 /TEN SECONDS, AND
      00330
0040 00120 JMS MSPLY /CONTINUE AS SOON AS
      00132
0042 00342 WRK /"TEST" IS ACTIVATED
0043 00024 JCN AZ RPT2
      00033
0045 00031 JCN TN *
      00045

```

```

/-
0047 00323 LDM 3 /TEST 3: ONE MINUTE COND. DELAY
0050 00120 JMS DSPLA /((0011) TO RAM1 PORT
      00115

```

```

/-
0052 00140 RPT3, INC R0 /INCREMENT FOUR BIT
0053 00240 LD R0 /BINARY OUTPUT CODE

```



```

0054 00341      WMP          /AT RAMO PORT EVERY
0055 00337      LDM 15       /MINUTE,          AND
0056 00271      XCH R9       /CONTINUE AS SOON AS
0057 00120      JMS MIDLY    /"TEST" IS ACTIVATED
      00126
0061 00342      WRK
0062 00024      JCN A7 RPT3
      00052
0064 00031      JCN IN *
      00064
/-
0066      00024      LDM 4 /TEST 4: ONE SECOND UNCOND. DELAY
0067 00120      JMS DSPLA    / (0100) TO RAM1 PORT
      00115
/-
0071 00372      STC
0072 00040      RPT4, LD R0    /UNCONDITIONALLY
0073 00341      WMP          /INCREMENT FOUR BIT
0074 00052      FIM P5 252    /BINARY OUTPUT CODE
      00374
0076 00120      JMS GSDLY    /AT RAMO PORT EVERY
      00132
0100 00342      WRK          /SECOND,          ANI
0101 00034      JCN NZA ERROR /RETURN TO START UP
      00107
0103 00160      ISZ R0 RPT4   /PROGRAM AFTER
      00072
0105 00100      JUN START    /SIXTEEN SECONDS
      00000
/-
0107 00337      ERROR, LDM 15 /ERROR: (1111) TO RAMO,
0110 00120      JMS DSPLA    /RAM1, AND ROMO PORTS
      00115
0112 00341      WMP
0113 00100      JUN *        /HALT
      00113
/-
0115 00040      DSPLA, FIM P0 1000 / (ACC) TO RAM1 OUTPUT
      00100
0117 00041      SRC P0       /PORT TO DISPLAY THE
0120 00341      WMP          /TEST NUMBER
0121 00040      FIM P0 0     /SELECT RAMO AND ROMO
      00000
0123 00041      SRC P0
0124 00361      CLC          /CY=0 FOR COND. DELAY
0125 00317      BBL 15
/-
/XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
/-
0126 00052      MIDLY, FIM P5 16
      00020
0130 00100      JUN GSDLY+2
      00134
0132 00337      GSDLY, LDM 15

```

```

0133 00271      XCH R9
0134 00056      FIM F7 5
          00005
0136 00120      JMS MSDLY
          00151
0140 00034      JCN NZA EXIT
          00172
0142 00173      ISZ R11 GSDLY+2
          00134
0144 00172      ISZ R10 GSDLY+2
          00134
0146 00171      ISZ R9 MIDLY
          00126
0150 00300      BBL 0          /TIME OUT
0151 00033      MSDLY, JCN 130 EXIT
          00172
0153 00335      RTR12, LDM 13
0154 00274      XCH R12
0155 00100      NOP
0156 00000      NOP
0157 00324      RTR13, LDM 4
0160 00275      XCH R13
0161 00175      ISZ R13 *
          00161
0163 00174      ISZ R12 RTR13
          00157
0165 00177      ISZ R15 MSDLY
          00151
0167 00176      ISZ R14 RTR12
          00153
0171 00300      BBL 0          /TIME OUT
0172 00317      EXIT, BBL 15    /EARLY EXIT
/-

```

/XX

SYMBOL TABLE:

CLEAR 000004	MIDLY 000151	RPT4 000072
DSPLA 000115	GSDLY 000132	RTR12 000153
ERR0R 000107	RPT1 000014	RTR13 000157
EXIT 000172	RPT2 000033	STAR1 000000
MIDLY 000126	RPT3 000052	



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-4

4-11

☒ 4004 ☐ 4040 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

HSP (High Speed Printer Interface)

Function

HSP is a subroutine which writes the ASCII character, stored in IR2 and IR3 to a Centronics 101 High Speed Printer. The routine waits for the Printer to acknowledge receipt of the character before existing.

(Note: using this high speed printer with the Intellec 4 Assembler Version 3.0 Modified, will speed up pass 2 listing by more than 10 times)

Required
Hardware

4004, 4008, 4009, ROM output Ports 6, 7, 8
ROM input port 8

(see attached logic diagram)

Required
Software

None

Input
Parameters

The ASCII value of the character to be printed should be in index registers 2 and 3 with the most significant nibble in R2 and least significant in R3.

Output
Results

N/A

Registers Modified: IR4, IR5	Assembler/Compiler Used: Intellec 4 Resident Assembler version 3.0
RAM Required: None	Programmer: Daniel D. Hammond
ROM Required: 17 locations	Company: Automation Associates, Inc.
Maximum Subroutine Nesting Level: One	Address: P. O. Drawer I Cape Canaveral, FL 32920

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

- 1 Complete Submittal Form as follows (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps is not necessary.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

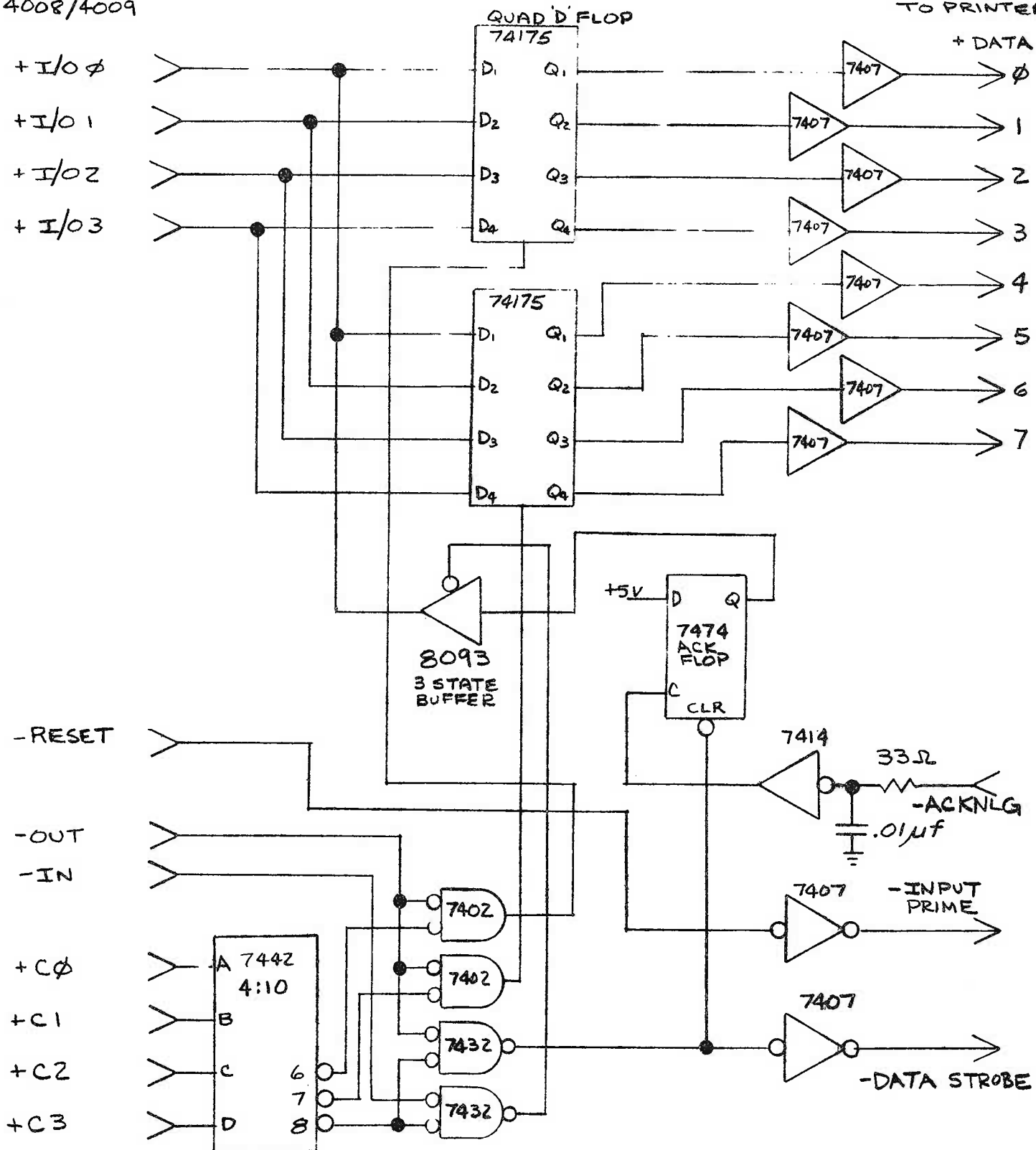
Your program will be photo-copied for publication in the User's Library. Please send an original, clear, un-marked copy.

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

FROM 4008/4009

TO PRINTER



DSN 4/2/75

INTEL 4 INTERFACE TO CENTRONICS CI PRINTER

A

SHEET 1 OF 1

```

;*****
;
;       HSP TEST ROUTINE (PRINT ASCII COLUMNS 3,4 & 5 ON H.S. PRINTER)
;
0002      R2      EQU      2          ; ASSIGN INDEX REGISTERS
0003      R3      EQU      3          ; R2 AND R3 TO BE USED FOR OUTPUT BUFFER
0004      R4      EQU      4          ; R4 TO BE USED AS COLUMN COUNTER
0006      R6      EQU      6          ; R6,R7 TO BE USED IN PORT SELECTION
000D      CR      EQU      0DH        ; ASCII VALUE OF CARRIAGE RETURN
000A      LF      EQU      0AH        ; ASCII VALUE OF LINE FEED
;
;
0000      2230    TSTHSP:FIM      R2,30H      ; START PRINTING WITH ASCII 0
0002      24D0          FIM      R4,0DOH      ; SET COLUMN COUNTER = -3
0004      5015          JMS      HSP          ; PRINT CHARACTER
0006      7304          ISZ      R3,$-2        ; INCREMENT ASCII ROW
0008      62          INC      R2            ; INCREMENT ASCII COLUMN
0009      7404          ISZ      R4,$-5        ; INCREMENT COLUMN COUNTER, SKIP IF ZERO
000B      220D          FIM      R2,CR        ; LOAD CARRIAGE RETURN
000D      5015          JMS      HSP          ; SEND CR TO PRINTER
000F      220A          FIM      R2,LF        ; LOAD LINE FEED
0011      5015          JMS      HSP          ; SEND LF TO PRINTER
0013      4000          JUN      TSTHSP       ; LOOP
;
;
;*****
;
;       WRITE ASCII CHARACTER IN R2,R3 TO HIGH SPEED PRINTER
;
;
0015      2660    HSP:  FIM      R6,60H      ; ACCESS ROM PORTS.
0017      27          SRC      R6          ; SELECT LS PORT
0018      A3          LD      R3
0019      E2          WRR          ; WRITE LS NIBBLE TO ROM6 OUTPUT PORT
001A      66          INC      R6          ; INCREMENT PORT SELECTION
001B      27          SRC      R6          ; SELECT MS PORT
001C      A2          LD      R2
001D      E2          WRR          ; WRITE MS NIBBLE TO ROM7 OUTPUT PORT
001E      66          INC      R6
001F      27          SRC      R6          ; ACCESS PORT 8 (CONTROL PORT)
0020      E2          WRR          ; STROBE DATA TO PRINTER & RESET ACK FLOP
0021      EA          RDR          ; READ STATE OF ACK FLOP
0022      F6          RAR          ; ROTATE ACK BIT INTO CARRY/LINK
0023      1A21        JNC      $-2        ; WAIT FOR ACK
0025      C0          BBL      0          ; EXIT
END

```




MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 40-2☒ 4004 ☒ 4040 ☐ 8008 ☐ 8080 ☐ 3000

(use additional sheets if necessary)

Program
Title

TP3100

Function

This is a Subroutine which provides a Software Interface to a Bowmar TP3100 Thermal Printer. The Subroutine provides all timing as well as control and character print functions.

Required
Hardware

Bowmar TP3100 Thermal Printer
Nortec 4881 Character Generator Chip (Sold with above printer) wired as shown on attached schematic.

Required
Software

None

Input
Parameters

Stack Registers 2 & 3 contain the 7-Bit ASCII Equivalent of the character to be printed.
The Test Line is connected to End-of-Travel Switch in Printer (See attached schematic)
Stack Register 14 is used to store the current step position of the Print Head stepping motor.

Output
Results

Ram Port 3 outputs stepper motor commands
Rom output Ports 2 & 3 output the 7-Bit ASCII Character to the Printer.
Ram Port 2 Bit 0 is Print Command

Registers Modified: 0,1	Assembler/Compiler Used: Dow Intel 4 Batch/Time Share Assembler
RAM Required: None	Programmer: Richard J. Brod
ROM Required: 129 Words	Company: Dow Chemical U.S.A.
Maximum Subroutine Nesting Level: 1	Address: Freeport, Tx 77541

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
 2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
 3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.
 4. A source paper tape of the contributed program is required. This insures that a clear, original copy of the program is available to photo-copy for publication in a User's Library update publication.
-

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

1.000	0000 0000 0020	FWDSTEP,	FIM OP BTL	/ LOAD BRANCH TABLE ADDRESS FOR PRINT HEAD FORWARD STEP
	0001 0001 000C			
2.000	0002 0002 00F0	FRSTEP,	CLB	
3.000	0003 0003 00FD		DCL	
4.000	0004 0004 00AE		LD 14	
5.000	0005 0005 0081		ADD 1	
6.000	0006 0006 00B1		XCH 1	
7.000	0007 0007 001A		JCN 10 NC	
	0008 0008 000A			
8.000	0009 0009 0060		INC 0	/ ADD CURRENT STEP POSITION TO BRANCH TABLE ADDRESS
9.000	0010 000A 0030	NC,	FIN OP	
10.000	0011 000B 0031		JIN OP	/ BRANCH TO NEXT STEP POSITION INSTRUCTIONS
11.000	0012 000C 0015	BTL,	STEP2	
12.000	0013 000D 001A		STEP3	
13.000	0014 000E 001F		STEP4	BRANCH TABLE FOR FORWARD STEPPING OF PRINT HEAD.
14.000	0015 000F 0010		STEP1	
15.000	0016 0010 00D0	STEP1,	LDM 0	
16.000	0017 0011 00BE		XCH 14	SET UP ACCUMULATOR AND CURRENT STEP POSITION REGISTER (R14)
17.000	0018 0012 00D9		LDM 9	
18.000	0019 0013 0040		JUN STEP	
	0020 0014 0035			
19.000	0021 0015 00D1	STEP2,	LDM 1	
20.000	0022 0016 00BE		XCH 14	
21.000	0023 0017 00DA		LDM 0AH	
22.000	0024 0018 0040		JUN STEP	
	0025 0019 0035			
23.000	0026 001A 00D2	STEP3,	LDM 2	
24.000	0027 001B 00BE		XCH 14	
25.000	0028 001C 00D6		LDM 6	
26.000	0029 001D 0040		JUN STEP	
	0030 001E 0035			
27.000	0031 001F 00D3	STEP4,	LDM 3	
28.000	0032 0020 00BE		XCH 14	
29.000	0033 0021 00D5		LDM 5	
30.000	0034 0022 0040		JUN STEP	
	0035 0023 0035			
31.000	0036 0024 001F	BTL2,	0+STEP4	
32.000	0037 0025 0010		0+STEP1	BRANCH TABLE FOR RETRACE OF PRINT HEAD
33.000	0038 0026 0015		0+STEP2	
34.000	0039 0027 001A		0+STEP3	
35.000	0040 0028 0020	RETRACE,	FIM OP BTL2	/ LOAD BRANCH TABLE ADDRESS FOR PRINT HEAD RETRACE
	0041 0029 0024			
36.000	0042 002A 0050		JMS FRSTEP	/ RETRACE ONE STEP
	0043 002B 0002			
37.000	0044 002C 0011		JCN 1 RETRACE	/ KEEP RETRACING UNTIL PRINT HEAD END OF TRAVEL CONTACTS CLOSE.
	0045 002D 0028			
38.000	0046 002E 0050		JMS FWDSTEP	
	0047 002F 0000			
39.000	0048 0030 0050		JMS FWDSTEP	FORWARD STEP THREE TIMES TO GET TO PRINT HEAD HOME POSITION.
	0049 0031 0000			
40.000	0050 0032 0050		JMS FWDSTEP	
	0051 0033 0000			
41.000	0052 0034 00C0		BBL 0	
42.000	0053 0035 0020	STEP,	FIM OP 0COH	/ NOW WE'LL OUTPUT STEP COMMANDS. SELECT RAM PORT 3
	0054 0036 00C0			
43.000	0055 0037 0021		SRC OP	
44.000	0056 0038 00E1		WMP	/ ISSUE STEP COMMAND
45.000	0057 0039 0020		FIM OP 0	
	0058 003A 0000			
46.000	0059 003B 0071	DL1,	ISZ 1 DL1	
	0060 003C 003B			
47.000	0061 003D 0070		ISZ 0 DL1	
	0062 003E 003B			
48.000	0063 003F 0071	DL2,	ISZ 1 DL2	
	0064 0040 003F			

FOR ~17ms

49.000	0065	0041	0070
	0066	0042	003F
50.000	0067	0043	0020
	0068	0044	001F
51.000	0069	0045	0071
	0070	0046	0045
52.000	0071	0047	0070
	0072	0048	0045
53.000	0073	0049	0020
	0074	004A	00C0
54.000	0075	004B	0021
55.000	0076	004C	00F0
56.000	0077	004D	00E1
57.000	0078	004E	00C0
58.000	0079	004F	00F0
59.000	0080	0050	00FD
60.000	0081	0051	0020
	0082	0052	0020
61.000	0083	0053	0021
62.000	0084	0054	00A2
63.000	0085	0055	00E2
64.000	0086	0056	0060
65.000	0087	0057	0021
66.000	0088	0058	00A3
67.000	0089	0059	00E2
68.000	0090	005A	0020
	0091	005B	0080
69.000	0092	005C	0021
70.000	0093	005D	00D1
71.000	0094	005E	00E1
72.000	0095	005F	0020
	0096	0060	0000
73.000	0097	0061	0071
	0098	0062	0061
74.000	0099	0063	0070
	0100	0064	0061
75.000	0101	0065	0071
	0102	0066	0065
76.000	0103	0067	0070
	0104	0068	0065
77.000	0105	0069	0020
	0106	006A	00F8
78.000	0107	006B	0071
	0108	006C	006B
79.000	0109	006D	0070
	0110	006E	006B
80.000	0111	006F	0020
	0112	0070	0080
81.000	0113	0071	0021
82.000	0114	0072	00F0
83.000	0115	0073	00E1
84.000	0116	0074	0020
	0117	0075	0000
85.000	0118	0076	0071
	0119	0077	0076
86.000	0120	0078	0070
	0121	0079	0076
87.000	0122	007A	0020
	0123	007B	00D1
88.000	0124	007C	0071
	0125	007D	007C
89.000	0126	007E	0070
	0127	007F	007C
90.000	0128	0080	00C0
91.000	0129	0081	0001

	ISZ 0 DL2
	FIM OP 1FH
DL3,	ISZ 1 DL3
	ISZ 0 DL3
	FIM OP 0C0H
	SRC OP
	CLB
	WMP
	BBL 0
PRINT,	CLB
	DCL
	FIM OP 20H
	SRC OP
	LD 2
	WRR
	INC 0
	SRC OP
	LD 3
	WRR
	FIM OP 80H
	SRC OP
	LDM 1
	WMP
	FIM OP 0
PDL1,	ISZ 1 PDL1
	ISZ 0 PDL1
PDL2,	ISZ 1 PDL2
	ISZ 0 PDL2
	FIM OP 0F8H
PDL3,	ISZ 1 PDL3
	ISZ 0 PDL3
	FIM OP 80H
	SRC OP
	CLB
	WMP
	FIM OP 0
CDL1,	ISZ 1 CDL1
	ISZ 0 CDL1
	FIM OP 0D1H
CDL2,	ISZ 1 CDL2
	ISZ 0 CDL2
	BBL 0
	.END

THEN CLEAR STEP COMMAND.
PRINT HEAD HAS MOVED
ONE STEP.

/ PRINT ROUTINE. ASCII EQUIV. OF
CHARACTER TO BE PRINTED IS IN
STACK REGISTERS 2 AND 3.
/ SELECT Rom Port 2

/ WRITE UPPER 3 BITS OF CHARACTER

/ WRITE LOWER 4 BITS OF CHARACTER TO
Rom Port 3.

/ SELECT RAM Port 2

/ ISSUE PRINT COMMAND

FOR
~ 12 mS

/ THEN CLEAR PRINT COMMAND

AND LET PRINT HEAD
COOL FOR ~ 7 mS.

TEST PROGRAM USING TP3100 SUBROUTINE

THIS PROGRAM INPUTS A CHARACTER FROM THE
TELETYPE AND ECHOES IT ON THE TP3100 PRINTER.

1.000	0000 0000 0050	JMS SR0	
	0001 0001 00BC		
2.000	0002 0002 00D1	LDM 1	SILENCE TTY
3.000	0003 0003 00E1	WMP	
4.000	0004 0004 00BE	XCH 14	LOAD 1 INTO PRINT HEAD STEP POSITION REGISTER
5.000	0005 0005 0050	JMS RETRACE	SEND PRINT HEAD TO HOME POSITION
	0006 0006 0063		
6.000	0007 0007 00F0	CLB	
7.000	0008 0008 00D8	LDM 8	
8.000	0009 0009 00B7	XCH 7	
8.500	0010 000A 0050	JMS SR0	
	0011 000B 00BC		
9.000	0012 000C 00EA	RDR	
10.000	0013 000D 00F6	RAR	
11.000	0014 000E 001A	JCN 10 T1	
	0015 000F 000C		
12.000	0016 0010 0050	JMS SBR1	
	0017 0011 00C6		
13.000	0018 0012 00E1	WMP	
14.000	0019 0013 0050	JMS SBR2	
	0020 0014 00C0		
15.000	0021 0015 00EA	RDR	
16.000	0022 0016 00F4	CMA	
17.000	0023 0017 00E1	WMP	
18.000	0024 0018 00F6	RAR	
19.000	0025 0019 00A2	LD 2	
20.000	0026 001A 00F6	RAR	
21.000	0027 001B 00B2	XCH 2	
22.000	0028 001C 00A3	LD 3	
23.000	0029 001D 00F6	RAR	
24.000	0030 001E 00B3	XCH 3	
25.000	0031 001F 0077	ISZ 7 T2	
	0032 0020 0013		
26.000	0033 0021 0050	JMS SBR2	
	0034 0022 00C0		
27.000	0035 0023 00D1	LDM 1	
28.000	0036 0024 00E1	WMP	
29.000	0037 0025 0050	JMS SBR2	
	0038 0026 00C0		
30.000	0039 0027 0050	JMS SBR1	
	0040 0028 00C6		
31.000	0041 0029 00B2	XCH 2	
32.000	0042 002A 00F5	RAL	
33.000	0043 002B 00F1	CLC	REMOVE PARITY BIT
34.000	0044 002C 00F6	RAR	
35.000	0045 002D 00B2	XCH 2	
36.000	0046 002E 00A2	LD 2	
37.000	0047 002F 001C	JCN 12 NORTN	
	0048 0030 0035		
38.000	0049 0031 00D3	LDM 3	CHECK CHARACTER
39.000	0050 0032 0083	ADD 3	TO SEE IF IT'S A
40.000	0051 0033 0014	JCN 4 ST	CARRIAGE RETURN
	0052 0034 0005		IF IT IS, JUMP BACK TO ST

TTY INPUT
ROUTINE

41.000	0053 0035 0050	NØRTN,	JMS PRINT	/IF CHARACTER IS NOT A CR, PRINT IT.
48.000	0054 0036 008A		JMS FWDSTEP	/STEP PRINT HEAD FORWARD
49.000	0055 0037 0050			
	0056 0038 003B			
49.000	0057 0039 0040		JUN TTI	/GET ANOTHER CHARACTER FROM TTY.
	0058 003A 0007			
50.000	0059 003B 0020	FWDSTEP,	FIM OP BTL	
	0060 003C 0047			
51.000	0061 003D 00F0	FRSTEP,	CLB	
52.000	0062 003E 00FD		DCL	
53.000	0063 003F 00AE		LD 14	TP3100
54.000	0064 0040 0081		ADD 1	
55.000	0065 0041 00B1		XCH 1	SUBROUTINE
56.000	0066 0042 001A		JCN 10 NC	
	0067 0043 0045			
57.000	0068 0044 0060		INC 0	
58.000	0069 0045 0030	NC,	FIN OP	
59.000	0070 0046 0031		JIN OP	
60.000	0071 0047 0050	BTL,	STEP2	
61.000	0072 0048 0055		STEP3	
62.000	0073 0049 005A		STEP4	
63.000	0074 004A 004B		STEP1	
64.000	0075 004B 00D0	STEP1,	LDM 0	
64.100	0076 004C 00BE		XCH 14	
64.200	0077 004D 00D9		LDM 9	
65.000	0078 004E 0040		JUN STEP	
	0079 004F 0070			
66.000	0080 0050 00D1	STEP2,	LDM 1	
66.100	0081 0051 00BE		XCH 14	
66.200	0082 0052 00DA		LDM 0AH	
67.000	0083 0053 0040		JUN STEP	
	0084 0054 0070			
100.000	0085 0055 00D2	STEP3,	LDM 2	
100.100	0086 0056 00BE		XCH 14	
100.200	0087 0057 00D6		LDM 6	
101.000	0088 0058 0040		JUN STEP	
	0089 0059 0070			
102.000	0090 005A 00D3	STEP4,	LDM 3	
102.100	0091 005B 00BE		XCH 14	
102.200	0092 005C 00D5		LDM 5	
103.000	0093 005D 0040		JUN STEP	
	0094 005E 0070			
104.000	0095 005F 005A	BTL2,	0+STEP4	
105.000	0096 0060 004B		0+STEP1	
106.000	0097 0061 0050		0+STEP2	
107.000	0098 0062 0055		0+STEP3	
108.000	0099 0063 0020	RETRACE,	FIM OP BTL2	
	0100 0064 005F			
109.000	0101 0065 0050		JMS FRSTEP	
	0102 0066 003D			
110.000	0103 0067 0011		JCN 1 RETRACE	
	0104 0068 0063			
111.000	0105 0069 0050		JMS FWDSTEP	
	0106 006A 003B			
112.000	0107 006B 0050		JMS FWDSTEP	
	0108 006C 003B			
112.500	0109 006D 0050		JMS FWDSTEP	
	0110 006E 003B			
113.000	0111 006F 00C0		BBL 0	
114.000	0112 0070 0020	STEP,	FIM OP 0C0H	
	0113 0071 00C0			
115.000	0114 0072 0021		SRC OP	
118.000	0115 0073 00E1		WMP	
119.000	0116 0074 0020		FIM OP 0	

120.000	0117 0075 0000		
	0118 0076 0071	DL1,	ISZ 1 DL1
	0119 0077 0076		
121.000	0120 0078 0070		ISZ 0 DL1
	0121 0079 0076		
122.000	0122 007A 0071	DL2,	ISZ 1 DL2
	0123 007B 007A		
123.000	0124 007C 0070		ISZ 0 DL2
	0125 007D 007A		
124.000	0126 007E 0020		FIM OP 1FH
	0127 007F 001F		
125.000	0128 0080 0071	DL3,	ISZ 1 DL3
	0129 0081 0080		
126.000	0130 0082 0070		ISZ 0 DL3
	0131 0083 0080		
127.000	0132 0084 0020		FIM OP 0C0H
	0133 0085 00C0		
128.000	0134 0086 0021		SRC OP
129.000	0135 0087 00F0		CLB
130.000	0136 0088 00E1		WMP
131.000	0137 0089 00C0		BBL 0
132.000	0138 008A 00F0	PRINT,	CLB
133.000	0139 008B 00FD		DCL
134.000	0140 008C 0020		FIM OP 20H
	0141 008D 0020		
135.000	0142 008E 0021		SRC OP
136.000	0143 008F 00A2		LD 2
138.000	0144 0090 00E2		WRR
139.000	0145 0091 0060		INC 0
140.000	0146 0092 0021		SRC OP
141.000	0147 0093 00A3		LD 3
143.000	0148 0094 00E2		WRR
144.000	0149 0095 0020		FIM OP 80H
	0150 0096 0080		
145.000	0151 0097 0021		SRC OP
146.000	0152 0098 00D1		LDM 1
147.000	0153 0099 00E1		WMP
148.000	0154 009A 0020		FIM OP 0
	0155 009B 0000		
149.000	0156 009C 0071	PDL1,	ISZ 1 PDL1
	0157 009D 009C		
150.000	0158 009E 0070		ISZ 0 PDL1
	0159 009F 009C		
151.000	0160 00A0 0071	PDL2,	ISZ 1 PDL2
	0161 00A1 00A0		
152.000	0162 00A2 0070		ISZ 0 PDL2
	0163 00A3 00A0		
153.000	0164 00A4 0020		FIM OP 0F8H
	0165 00A5 00F8		
154.000	0166 00A6 0071	PDL3,	ISZ 1 PDL3
	0167 00A7 00A6		
155.000	0168 00A8 0070		ISZ 0 PDL3
	0169 00A9 00A6		
156.000	0170 00AA 0020		FIM OP 80H
	0171 00AB 0080		
157.000	0172 00AC 0021		SRC OP
158.000	0173 00AD 00F0		CLB
159.000	0174 00AE 00E1		WMP
160.000	0175 00AF 0020		FIM OP 0
	0176 00B0 0000		
161.000	0177 00B1 0071	CDL1,	ISZ 1 CDL1
	0178 00B2 00B1		
162.000	0179 00B3 0070		ISZ 0 CDL1

	0180 00B4 00B1		
163.000	0181 00B5 0020		FIM OP OD1H
	0182 00B6 00D1		
164.000	0183 00B7 0071	CDL2,	ISZ 1 CDL2
	0184 00B8 00B7		
165.000	0185 00B9 0070		ISZ 0 CDL2
	0186 00BA 00B7		
166.000	0187 00BB 00C0		BBL 0
167.000	0188 00BC 0020	SR0,	FIM OP 0
	0189 00BD 0000		
168.000	0190 00BE 0021		SRC OP
169.000	0191 00BF 00C0		BBL 0
170.000	0192 00C0 0020	SBR2,	FIM OP 60
	0193 00C1 003C		
171.000	0194 00C2 0071	L2,	ISZ 1 L2
	0195 00C3 00C2		
172.000	0196 00C4 0070		ISZ 0 L2
	0197 00C5 00C2		
173.000	0198 00C6 0020	SBR1,	FIM OP 60
	0199 00C7 003C		
174.000	0200 00C8 0071	L1,	ISZ 1 L1
	0201 00C9 00C8		
175.000	0202 00CA 0070		ISZ 0 L1
	0203 00CB 00C8		
176.000	0204 00CC 00C0		BBL 0
177.000	0205 00CD 0001		.END

TTY

TIMING ROUTINES

--HEX-- --DEC--

	****	****
BTL	0047	0071
BTL2	005F	0095
CDL1	00B1	0177
CDL2	00B7	0183
DL1	0076	0118
DL2	007A	0122
DL3	0080	0128
FRSTEP	003D	0061
FWDSTEP	003B	0059
L1	00C8	0200
L2	00C2	0194
NC	0045	0069
NØRTN	0035	0053
PDL1	009C	0156
PDL2	00A0	0160
PDL3	00A6	0166
PRINT	008A	0138
RETRACE	0063	0099
SBR1	00C6	0198
SBR2	00C0	0192
SR0	00BC	0188
ST	0005	0005
STEP	0070	0112
STEP1	004B	0075
STEP2	0050	0080
STEP3	0055	0085
STEP4	005A	0090
TT1	0007	0007
T1	000C	0012
T2	0013	0019

0 ERRORS



DOW CHEMICAL U.S.A.
TEXAS DIVISION

JOB NO. _____

AUTH. NO. _____

CHARGE NO. _____

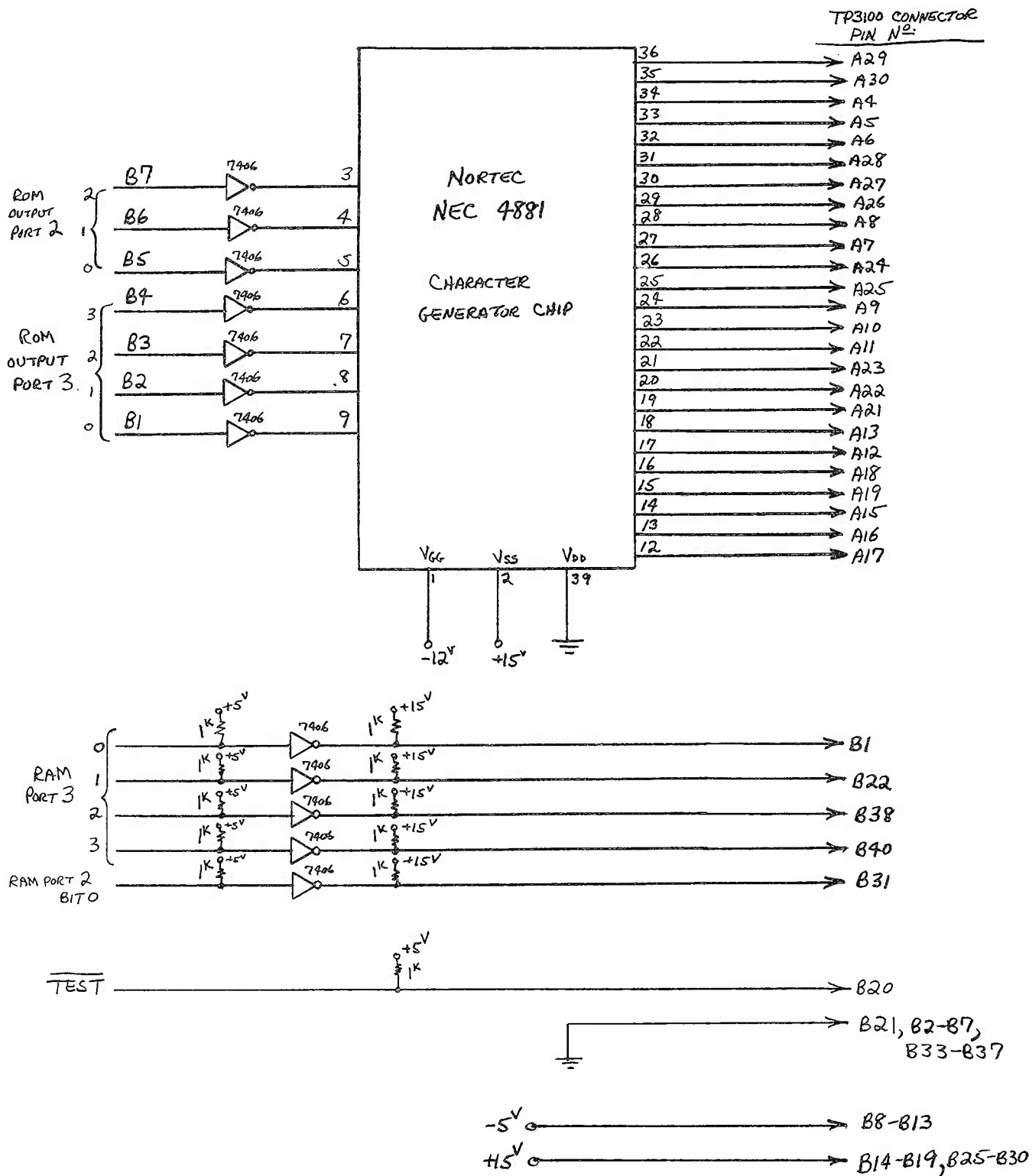
FILE NO. _____

SUBJECT 4004/4040 — Bowmar TP3100 INTERFACEBY R. J. BROD

DATE _____

SHEET 1 OF 1

CHECKED BY _____





MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 40-3☐ 4004 ☒ 4040 ☐ 8008 ☐ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title I/O Test

Function To exercise all I/O lines to allow for troubleshooting of system design, wiring errors, and chip malfunctions.

Required Hardware User's system

Required Software No additional software

Input Parameters No initialization required

Output Results All RAM and ROM output ports shall appear to be 4 bit synchronous up counters updating in order of their designation from 0-15. Input port strobes will appear following the output port strobe of the same address.

Registers Modified: 0, 1, 2, 3	Assembler/Compiler Used: 4004 Macro Assembler, Ver. 2.4
RAM Required: -0-	Programmer: Lee Jay Mandell
ROM Required: 21 bytes	Company: Waugh Controls Corp.
Maximum Subroutine Nesting Level: 0	Address: 9001 Fullbright Ave. Chatsworth, CA 91311

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
 2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
 3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.
 4. A source paper tape of the contributed program is required. This insures that a clear, original copy of the program is available to photo-copy for publication in a User's Library update publication.
-

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

```

;I/O TEST PROGRAM
;BY LEE JAY MANDELL
;WAUGH CONTROLS CORP.
;
;

```

```

0000          ORG 0
0000    2000    TEST: FIM 0,00H
0002    A1      LOOP: LD 1
0003    21      SRC 0
0004    E2      WRR                      ;WRITE ROM OUTPUT PORT
0005    221C    FIM 2,1CH
0007    B2      RAM: XCH 2
0008    FD      DCL
0009    F1      CLC
000A    F5      RAL
000B    B2      XCH 2
000C    E1      WMP                      ;WRITE RAM OUTPUT PORT
000D    7307    ISZ 3,RAM
000F    EA      RDR                      ;READ ROM INPUT PORT
0010    7002    ISZ 0,LOOP
0012    61      INC 1
0013    4002    JUN LOOP
                END

```

NO PROGRAM ERRORS



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 40-4☒ 4004 ☒ 4040 ☐ 8008 ☐ 8080 ☐ 3000

(use additional sheets if necessary)

Program
Title

8 DIGIT REGISTER DISPLAY

Function

Display 8 digits of data

Required
Hardware

2 output ports, MC 14511CP 7 segment decoder, MC 14028CP
octal decoder, MC 75491P segment driver (2), MC 75492P digit driver
(2), FNA-37 LED display, 1.5K resistor pack

Required
Software

See attachment

Input
Parameters

See attachment

Output
Results

8 digit octal realtime display

Registers Modified: 0,1,2,3,4,5	Assembler/Compiler Used: 4004 Macro Assembler, Ver. 2.4
RAM Required: None	Programmer: G. Barrowcliff
ROM Required: 28 words	Company: LTV Aerospace
Maximum Subroutine Nesting Level: 1	Address: 306 Shadow Lane Euless, Texas 76039

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
 2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
 3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.
 4. A source paper tape of the contributed program is required. This insures that a clear, original copy of the program is available to photo-copy for publication in a User's Library update publication.
-

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

The display is a Fairchild FNA 37 and is designed for use in a calculator. The display is a seven segment, 9 digit, LED, decimal model but, in this application, it is used as an 8 digit octal display.

The display requires two output ports, one to select the digit and one for the data to be displayed.

The hardware is connected for +15V and ground logic levels; however, if the system has a +5V, -9V power supply, the hardware will also operate at those levels providing the current limiting resistor pack is changed to ~ 680 to 820Ω . The display is intended for diagnostic work and draws approximately 4 MA when on. Any value which can be loaded in the accumulator can be displayed.

The program included has three segments. The section labeled Start is simply a jump to the Test routine. The Load routine sets the registers for masking the MSB of the data, sets the output port numbers (Port 4 = digit port number, Port 5 is the data port), loads the display data in the accumulator and calls the display routine. The display routine is general purpose and displays the accum data on the digit specified in the upper register of the data port pair. The loop at the end is the 4 MS delay and can be called as a general purpose delay from anywhere.

While this display is designed for displaying octal data, by using a National MM74C48 decoder, all 16 states could be displayed uniquely thus providing a hex display.

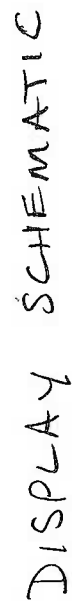
```

/
/
;8 DIGIT REGISTER DISPLAY
/
/

```

0000	00	START:	NOP	
0001	5005		JMS LOAD	
0003	4000		JUN START	
0005	00	LOAD:	NOP	
0006	2048		FIM 0,048H	;DIGIT PORT IN R0
0008	2250		FIM 2,050H	;DATA PORT IN R2
000A	A1	RLOOP:	LD 1	
000B	5011		JMS DISPLAY	
000D	710A		ISZ 1,RLOOP	
000F	C0		BBL 0	
0010	C0		BBL 0	
0011	Z1	DISPLAY:	SRC 0	;SELECT DIGIT PORT
0012	B3		XCH 3	;DATA TO R3, DIGIT # TO ACCUM
0013	E2		WRR	;SELECT DIGIT
0014	F2		IAC	;SET FOR NEXT CALL
0015	B3		XCH 3	;DIG+1 TO R3,DATA TO ACCUM
0016	23		SRC 2	;SELECT DATA PORT
0017	E2		WRR	
0018	2407	DELENT:	FIM 4,007H	
001A	741A	DISLOOP:	ISZ 4,DISLOOP	
001C	751A		ISZ 5,DISLOOP	
			END	

NO PROGRAM ERRORS



DISPLAY SCHEMATIC



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 40-5☐ 4004 ☒ 4040 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

Intellec 4/MOD 40-Silent 700 Interface

Function

Required
HardwareRequired
SoftwareInput
ParametersOutput
Results

SEE ATTACHED NOTE

Registers Modified:	Assembler/Compiler Used:
RAM Required:	Programmer: Phil Jensen
ROM Required:	Company:
Maximum Subroutine Nesting Level:	Address: 6350 LBJ Freeway #178 Dallas, Texas 75240

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
 2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
 3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.
-

Your program will be photo-copied for publication in the User's Library. Please send an original, clear, un-marked copy.

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

Intellec 4/Mod 40 - Silent 700 Interface

Introduction

This note describes an interface between the Intellec 4/Mod 40 Microcomputer Development System and the Texas Instruments "Silent 700" Electronic Data Terminal.

The interface consists of an Intellec/Silent 700 adapter. In addition, the "Silent 700" terminal and Intellec 4/Mod 40 must be modified. The adapter and equipment modifications are described in this note.

The interface was designed for use with the basic Intellec configuration (imm 4-44A) and a Model 733ASR terminal equipped with the 1200 Baud Transmission option and either the Automatic Device Control (ADC) option or Remote Device Control (RDC) option.

Intellec/Silent 700 Adapter

The Intellec/Silent 700 adapter connects the Intellec to the terminal interface cable and translates the current loop signals from the Intellec to RS232C signals for the terminal. A schematic of the adapter is shown in figure 1.

Silent 700 Modifications

1. Raise the cover on the terminal, loosen the two retaining screws on the rear card cage cover, and remove the cover.
2. Remove the 1200 baud transmitter card and locate networks Z17 and Z18.
3. Cut the etch between Z17 pin 2 and the VCC bus. This bus connects Z17 pin 2, Z17 pin 4 and Z13 pin 14.
4. Cut the etch between Z18 pin 1 and the feedthru located next to Z17 pin 1.
5. Connect a jumper wire from Z17 pin 2 to the feedthru located next to Z17 pin 1 (this feedthru was disconnected from Z18 pin 1).
6. Connect a jumper wire from Z18 pin 1 to Z18 pin 14.
7. Replace the 1200 baud transmitter card in the card cage.

TERMINAL MATING CONNECTOR

INTELLEC MATING CONNECTOR

P1- CINCH P-308-CCT

4 CONDUCTOR CABLE

J1-CINCH DB25S

CINCH DB-51226-1 SHELL

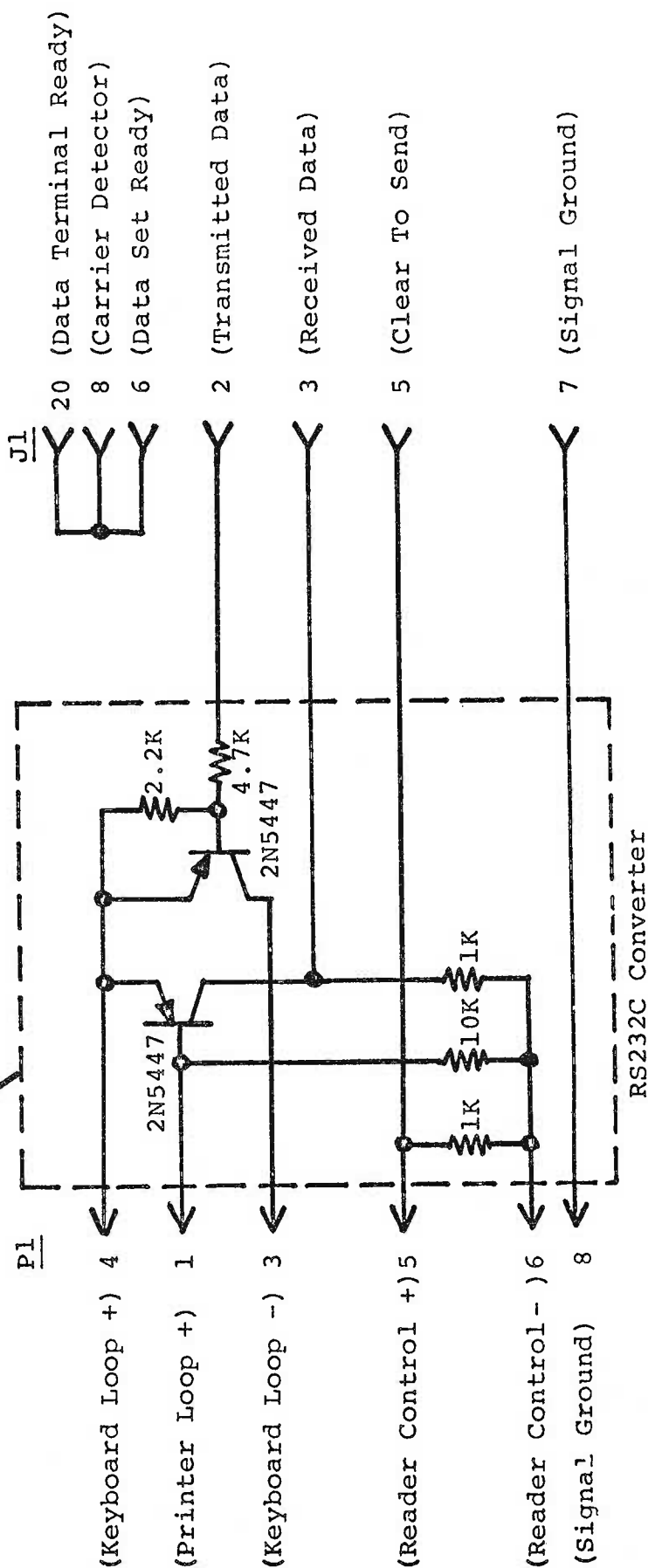


Figure 1. Intellec/Silent 700 Adapter

8. Remove the 1200 baud receiver card from the card cage and remove resistor R10 from the card.

9. Replace the 1200 baud receiver card in the card cage, replace the card cage cover, and secure with the two retaining screws. This completes the "Silent 700" modifications.

Intellec Hardware Modifications

Refer to figure 2 for the following modifications.

1. Place the Intellec upside-down on a flat surface, remove the five bottom cover retaining screws and remove the bottom cover.
2. Connect a jumper wire between J43 pin 8 and the ground bus located on the left side of the mother board.
3. Reverse the wires on J43 pins 3 and 4.
4. Connect a jumper wire accross the 220 ohm resistor located at the rear-left corner of the mother board.
5. Replace the bottom cover on the Intellec and secure with the five retaining screws. This completes the Intellec hardware modifications.

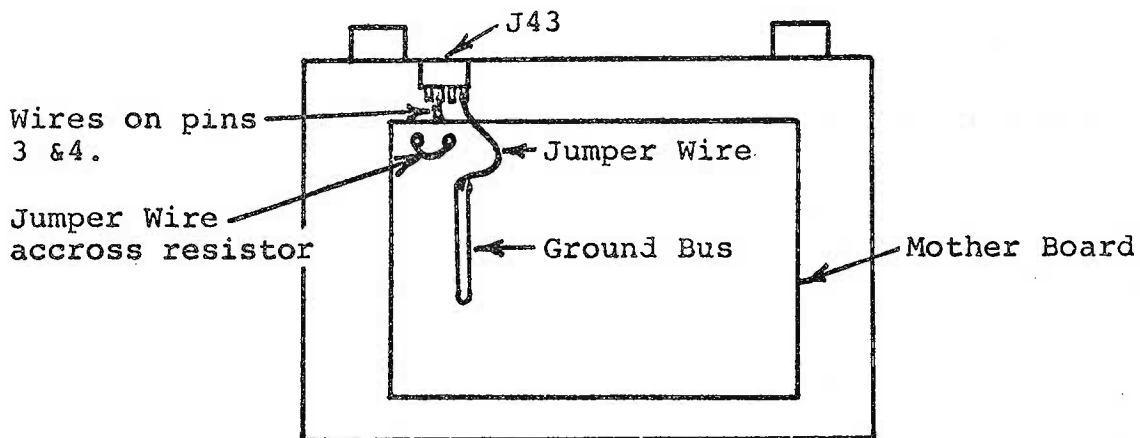


Figure 2. Intellec 4/Mod 40 Bottom View

Intellec Software Modifications

Refer to the Intellec 4/Mod 40 Operators Manual for the following modifications.

1. Using a Teletype, transfer the Intellec 4 monitor program from paper tape to RAM memory and modify according to figure 3.
2. Transfer the modified monitor program from RAM memory to four 4702A EPROMs.
3. Connect a jumper wire between J1 pins 5 and 6 on the Intellec/Silent 700 adapter.
4. Using a Teletype, transfer the Intellec 4 assembler program from paper tape to RAM memory and modify according to figure 4.
5. Disconnect the Teletype from the Intellec and connect the "Silent 700" terminal via the adapter.
6. Set the data rate switch on the "Silent 700" terminal to 10 characters per second.
7. Using the "Silent 700" terminal, transfer the modified assembler program from RAM memory to cassette.
8. Turn off the Intellec, open the top cover and remove the CPU module.
9. Replace the four monitor EPROMs on the CPU module with the four EPROMs that were programmed.
10. Replace the CPU module in the Intellec and close the top cover.
11. Remove the jumper wire between J1 pins 5 and 6 on the Intellec/Silent 700 adapter. This completes the interface modifications.

Operational Considerations

The modifications described in this note do not effect the basic operation of either the Intellec or the "Silent 700" terminal. Refer to the Intellec 4/Mod 40 Operators Manual and the "Silent 700" Operators Manual to perform the desired operations.

The following points should be remembered when using the Intellec with the terminal:

1. A higher storage efficiency will be realized if data is recorded on cassette in the CONTINUOUS mode. Hex memory dumps and object programs produced by the assembler should be recorded in this mode. When recording in hex format, over 50,000 bytes of data can be stored on each side of a cassette.

2. Source programs should be recorded in the LINE mode to simplify editing.

3. Data recorded on cassette should be followed with an X-OFF character. The X-OFF character will automatically stop the cassette after data has been read from the playback transport.

4. If a read error occurs while loading an object program, the Inteltec will automatically stop the cassette. To restart, depress the BLOCK REVERSE switch twice to reposition the record in error and enter an "R" command. The tape will restart from this point.

```

;TI 733ASR TERMINAL CONTROL PROGRAM
;
;MODIFICATIONS TO INTELLEC 4 MONITOR, VERSION 2.1
;
0007      START EQU      7          ;MONITOR RESTART POINT
00FA      LER      EQU      0FAH    ;ERROR TRAP
002A      ORG      2AH              ;B COMMAND TRAP
002A      40FA      JUN      LER
003A      ORG      3AH              ;J COMMAND TRAP
003A      40FA      JUN      LER
003C      40FA      JUN      LER    ;K COMMAND TRAP
003E      40FA      JUN      LER    ;L COMMAND TRAP
;
;CASSETTE/PRINTER OUTPUT ROUTINE (300 BAUD)
;
0081      ORG      81H
0081      2030      COI      FIM      0,30H
0083      21        SRC      0
0084      A2        LD      2
0085      E4        WR0
0086      A3        LD      3
0087      E5        WR1
;
;CHARACTER DISASSEMBLY ROUTINE
;
0088      2000      FIM      0,0
008A      21        SRC      0
008B      D0        LDM      0
008C      E1        WMP
008D      08        LDM      8
008E      B4        XCH      4
008F      5393      JMS      DLY2
0091      4286      JUN      COI
0093      4108      CIX:     JUN      CI
;
;KEYBOARD INPUT ROUTINE (300 BAUD)
;
009D      ORG      9DH
009D      50B8      KI:     JMS      CKS
009F      1C9D      JNZ      KI
00A1      EA        RDR
00A2      F6        RAR
00A3      1AA1      JNC      5-2
00A5      5108      JMS      CI
00A7      4081      JUN      CO
;
;CASSETTE INPUT ROUTINE (1200 BAUD)
;
0049      50B8      RI:     JMS      CKS
004B      1C93      JNZ      CIX

```


00AD	2640	FIM	0,40H
00AF	27	SRC	0
00B0	00	LDM	0
00B1	E1	WFP	
00B2	2211	FIM	2,11H
00B4	50B1	JMS	CO
00B5	40A9	JUN	RT

```

: INPUT BIT STATUS CHECK

```

00BB	2640	CK5:	FIN	6,40H
00BA	27		SRC	6
00BB	D1		LDM	1
00BC	E1		WAP	
00BD	2000		FIN	0,0
00BE	21		SRC	0
00C0	EA	CK1:	RDR	
00C1	F6		RAR	
00C2	1AC5		JNC	DCR
00C4	C1		BRL	1
00C5	70C0	DCR:	ISZ	0,CK1
00C7	71C0		ISZ	1,CK1
00C9	C0		BRL	0

120 MS TIME OUT

00FE		JRG	UFEM
00FE	439D	JUN	STOP

ERROR TRAP

0106		ORG	100H
0106	40A9	JUN	RI

;CASSETTE INPUT TRAP

CHARACTER ASSEMBLY ROUTINE

0108	5397	CI:	JMS	DLX1
010A	2640		FIM	6,40H
010C	27		SRC	6
010D	E1		WMP	
010E	2000		FIM	0,0
0110	21		SPC	0
0111	D8		LDM	8
0112	B4		XCH	4
0113	5393	CI1:	JMS	DLX2
0115	EA		RDM	
0116	F4		CMA	
0117	F6		RAR	
0118	A2		LD	2
0119	F6		RAR	
011A	B2		XCH	2
011B	A3		LD	3
011C	F6		RAR	
011D	B3		XCH	3
011E	7413		ISZ	4,CI1

CI:	JMS	DLX1
	FIM	0,40H
	SRC	6
	WMP	
	FIM	0,0
	SFC	0
	LDM	8
	XCH	4
CI1:	JMS	DLX2
	RDR	
	CMA	
	RAR	
	LD	2
	RAR	
	XCH	2
	LD	3
	RAR	
	XCH	3
	ISZ	4,CI1

```

0120 5393      JMS      DLY2
0122 B2        XCH      2
0123 F5        RAL
0124 F1        CLC
0125 F6        RAR
0126 B2        XCH      2
0127 C0        BBL      0

;
0243          ORG      243H      ;FAST PROGRAM MOD
0243 00        NOP
0244 00        NOP

;
0286          ORG      286H
0286 F1        COL: CLC
0287 B2        XCH      2
0288 F6        RAR
0289 B2        XCH      2
028A B3        XCH      3
028B F6        RAR
028C B3        XCH      3
028D F7        ICC
028E E1        WMP
028F 5393      JMS      DLY2
0291 7486      ISZ      4,C01
0293 D1        LDM      1
0294 E1        WMP
0295 5393      JMS      DLY2
0297 2030      FIM      0,30H
0299 21        SPC      0
029A EC        RDO
029B B2        XCH      2
029C ED        RDI
029D B3        XCH      3
029E A4        LD       4
029F E4        WRO
02A0 A5        LD       5
02A1 E5        WRI
02A2 2000      FIM      0,0
02A4 24CF      FIM      4,0CFH      ;25 MS PRINTER SYNC DELAY
02A6 4375      JUM      CRC

;
0310          ORG      310H      ;EOF TRAP
0310 439D      JUM      STOP

;
0375          ORG      375H
0375 F0        CRC: CLB
0376 92        SUB      2
0377 1C80      JNZ      DLY
0379 DD        LDM      0DH
037A F1        CLC
037B 93        SUB      3

```

```

037C 1C80      JNZ      DLY
037E 244E      FIM      4,4EH      ;165 MS CARRIAGE RTN SYNC
0380 7080      DLY:    ISZ      0,DLY
0382 7180      ISZ      1,DLY
0384 7480      ISZ      4,DLY
0386 7580      ISZ      5,DLY
0388 2030      FIM      0,30H
038A 21        SRC      0
038B EC        R00
038C B4        XCH      4
038D ED        R01
038F H5        XCH      5
038F 2000      FIM      0,0
0391 21        SRC      0
0392 C0        BBL      0

;
;BIT TIMING ROUTINE
;
0393 2020      DLY2: FIM      0,40H      ;691 US DELAY
0395 7095      ISZ      0,S
0397 2010      DLY1: FIM      0,10H      ;367 US DELAY
0399 7099      ISZ      0,S
039B 00        R0P
039C C0        BBL      0

;
;PLAYBACK STOP ROUTINE
;
039D 2213      STOP: FIM      2,13H
039F 5081      JMS      C0
03A1 4007      JDB      START
                     END

```

NO PROGRAM ERRORS

```

;
; TI 733 ASR TERMINAL CONTROL PROGRAM
;

```

```

; MODIFICATIONS TO INTELLEC 4 ASSEMBLER, VERSION 3.0
;

```

```

0040          ORG      40H
0040          DB       0,0,0,0,"ASM4,V3.0X"
0044          00000000
0044          41534D34
0048          2C56332E
004C          3058

```

```

; CASSETTE/PRINTER OUTPUT ROUTINE (300 BAUD)
;

```

```

020E          ORG      20EH
020E          CO:     FIM      4,40H
0210          25      SRC      4
0211          A2      LD       2
0212          E4      WRD
0213          A3      LD       3
0214          E5      WR1

```

```

; CHARACTER DISASSEMBLY ROUTINE (1200 BAUD)
;

```

```

0215          2400          FIM      4,0
0217          25          SRC      4

0218          E1          WMP
0219          D0          LDM      0
021A          D8          LDM      8
021B          B4          XCH      4
021C          5287        JMS      DLY2
021E          F1          CO1:    CLC
021F          B2          XCH      2
0220          F6          RAR
0221          B2          XCH      2
0222          B3          XCH      3
0223          F6          RAR
0224          B3          XCH      3

0225          F7          TCC
0226          E1          WMP
0227          5287        JMS      DLY2
0229          741E        ISZ      4,CO1
022B          D1          LDM      1
022C          E1          WMP
022D          5287        JMS      DLY2

022F          2440          FIM      4,40H
0231          25          SRC      4
0232          EC          RDO

```

0233	B2	XCH	2	
0234	ED	RD1		
0235	B3	XCH	3	
0236	2600	FIM	6,0	
0238	24CF	FIM	4,0CFH	;25 MS PRINTER SYNC DELAY
023A	F0	CLB		
023B	92	SUB	2	
023C	1C45	JNZ	DLY	
023E	DD	LDM	0DH	
023F	F1	CLC		
0240	93	SUB	3	
0241	1C45	JNZ	DLY	
0243	244E	FIM	4,4FH	;165 MS CARRIAGE RETURN DELAY
0245	7645	DLY:	ISZ	6,DLY
0247	7745		ISZ	7,DLY
0249	7445		ISZ	4,DLY
024B	7545		ISZ	5,DLY
024D	C0	BBL	0	

; KEYBOARD INPUT ROUTINE (300 BAUD)

0254		ORG	254H	
0254	5275	KI:	JMS	CKS
0256	1C54		JNZ	KI
0258	EA		RDR	
0259	F6		RAR	
025A	1A58		JNC	6-2
025C	5291		JMS	CI
025E	420E		JUN	CD

; CASSETTE INPUT ROUTINE (1200 BAUD)

0260	5275	RI:	JMS	CKS
0262	146A		JZ	XDN
0264	5291		JMS	CI
0266	EE		RD2	
0267	140E		JZ	CD
0269	C0		BBL	0
026A	2640	XDN:	FIM	6,40H
026C	27		SRC	6
026D	D0		LDM	0
026F	F1		WMP	
026F	2211		FIM	2,11H
0271	520E		JMS	CD
0273	4260		JUN	RI

; INPUT BIT STATUS CHECK

0275	2640	CKS:	FIM	6,40H
0277	27		SRC	6

```

0278      D1          LDM          1
0279      E1          WMP
027A      2400        FIM          4,0
027C      25          SRC          4
027D      EA          CK1: RDR
027E      F6          RAR
027F      1A82        JNC          DCR
0281      C1          BBL          1
0282      747D        DCR: ISZ      4,CK1      ;20 MS TIME OUT
0284      757D        ISZ          5,CK1
0286      C0          BBL          0

```

```

;
;BIT TIMING ROUTINE
;

```

```

0287      2620        DLY2: FIM      6,20H      ;691 US DELAY
0289      7689        ISZ          6,5
028B      2610        DLY1: FIM      6,10H      ;367 US DELAY
028D      768D        ISZ          6,5
028F      00          NOP
0290      C0          BBL          0

```

```

;
;CHARACTER ASSEMBLY ROUTINE (1200 BAUD)
;

```

```

0291      528B        C1:   JMS      DLY1
0293      2640        FIM      6,40H
0295      27          SRC      6
0296      E1          WMP
0297      2400        FIM      4,0
0299      25          SRC      4
029A      D8          LDM      8
029B      B4          XCH      4
029C      5287        C11:  JMS      DLY2
029E      EA          RDR
029F      F4          CMA
02A0      F6          RAR
02A1      A2          LD       2
02A2      F6          RAR
02A3      B2          XCH      2
02A4      A3          LD       3
02A5      F6          RAR
02A6      B3          XCH      3
02A7      749C        ISZ      4,C11
02A9      5287        JMS      DLY2
02AB      B2          XCH      2
02AC      F5          RAL
02AD      F1          CLC
02AE      F6          RAR
02AF      B2          XCH      2
02B0      C0          BBL      0

```

02D3		ORG	203H	
02D3	4260	JUN	RI	;RI TRAP

		:		
		:		
OCF2		ORG	OCF2H	
OCF2	2230	FIM	2,30H	
OCF4	520E	JMS	CO	
OCF6	520E	JMS	CO	
OCF8	2213	FIM	2,13H	
OCFA	520E	JMS	CO	
OCFC	CO	BBL	0	
		END		

NO PROGRAM ERRORS



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 40-6

☒ 4004 ☐ 4040 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

**Program
Title**

PDUP: PROM Dump Utility Program

Function

This is a program to dump the contents of a PROM in the front panel socket onto the teletype printer. The first address and first word is always printed out in the form

"00-00"

as address-contents. All subsequent address-contents listing are printed out only if the contents of the respective location are different from the contents of the previous location.

**Required
Hardware**

The Intellec 4 and ASR-33 are required.

**Required
Software**

There is no additional required software.

**Input
Parameters**

No input is required.

**Output
Results**

The output results are

00-00 A1-C1 A2-C2 ... An-Cn

in the form address-contents.

See the program listing for further description.

Registers Modified:

0-5, E, F

Assembler/Compiler Used:

Intellec 4 V3.0

RAM Required:

none

Programmer:

Mark Rothstein

ROM Required:

77(Hex) words

Company:

G & R Associates

Maximum Subroutine Nesting Level:

1

Address:20716
15401 Pegg Court, Bowie, MD

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

Your program will be photo-copied for publication in the User's Library. Please send an original, clear, un-marked copy.

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

```

C *****
C
C   THIS STAND-ALONE UTILITY PROGRAM WILL ALLOW A USER WITH AN INTELLEC
C   4MOD4 OR 4MOD40 AND TELETYPEWRITER TO VERY RAPIDLY CHECK THE CONTENTS
C   OF A PROM, PROVIDED THAT ALL, OR NEARLY ALL, OF THE WORDS STORED IN
C   THE PROM ARE THE SAME
C
C   CHECKING A PROM FOR COMPLETE ERASURE TAKES LESS THAN 1 SEC AFTER
C   THE TTY PRINTS OUT "00-00"
C
C   [ADDRESS-CONTENTS]
C   ALL CHANGES ARE ALSO PRINTED OUT IN THIS FORMAT.
C   THIS IS COMPARED TO THE TWO MINUTES REQUIRED FOR THE MONITOR SYSTEM
C   PROM PROGRAMMING ROUTINE (WITH THE PROGRAMMER POWER OFF, OF COURSE)
C
C   THIS PROGRAM IS ALSO USEFUL FOR CHECKING TO SEE THAT ALL BITS OF A
C   PROM CAN BE PROGRAMMED. A USER MAY PROGRAM 0A5H INTO ALL THE WORDS
C   OF A PROM, AND THEN CHECK IT, ERASE THE PROM, PROGRAM 05AH AND CHECK
C   IT AGAIN.
C
C   INPUT: NONE
C   OUTPUT: 00-00 A1-C1 A2-C2 ... AN-CN^
C           WHERE A1, A2,...,AN ARE THE ADDRESSES WHERE CHANGES OCCUR
C           AND C1, C2,...,CN ARE THE CONTENTS AT THOSE ADDRESSES.
C           ^ INDICATES THAT THE PROGRAM HAS REACHED PROM
C           LOCATION FF AND HAS FINISHED.
C
C   THIS PROGRAM CONTAINS NO CARRIAGE RETURN.
C   IT SHOULD BE USED FOR DUMPING PROMS WITH FEWER THAN 12 WORD
C
C   CHANGE BOUNDARIES.
C
C *****
C   <THIS PROGRAM WAS ASSEMBLED AND TESTED ON AN INTELLEC 4MOD40>
C *****
C
C   NOP
C   FIM 0 00      INITIAL PROM ADDRESS
C   FIM 2 0F      "PREVIOUS" DATA
C *****
C
C   PROM ACCESS ROUTINE.
C   FIM 4 00      ADDRESS THE PROM
C   SRC 5
C   LD 0          LOWER 4 BITS
C   WRR
C   INC 4          REG 4=1
C   SRC 5
C   LD 1          UPPER 4 BITS
C   WRR
C   INC 4          REG 4=2
C   SRC 5          GET READY TO READ

```

```

000 00
001 20 00
003 22 0F

```

```

005 24 00
007 25
008 A0
009 E2
00A 64
00B 25
00C A1
00D E2
00E 64
00F 25

```

*PR 1

PAGE 5

PROM DUMP UTILITY PROGRAM V1.0 5/16/75

```

010 EA      RDR      READ LOWER 4 BITS
011 B6      XCH 6    STORE TEMP IN REG 6
012 A6      LD 6
013 F1      CLC
014 92      SUB 2     COMPARE WITH PREVIOUS
015 B5      XCH 5     DATA
016 64      INC 4     REG 4=3
017 25      SRC 5
018 EA      RDR      READ UPPER 4 DATA BITS
019 B7      XCH 7     STORE TEMP IN REG 7

01A A7      LD 7
01B F1      CLC
01C 93      SUB 3     COMPARE
01D 1C 2C   JCN A1 E2 IF NOT ZERO, PRINT PROM CONTENTS AT THIS ADDRESS
01F A5      LD 5
020 1C 2C   JCN A1 E2 IF NOT ZERO, PRINT PROM CONTENTS AT THIS ADDRESS
022 70 05   ISZ 0 PR1 GO FROM ADDRESS 00 TO FF
024 71 05   ISZ 1 PR1
026 2E 5E   FIM E 5E      'A'

C      END OF LOOP
C
C *****
028 50 4E   JMS PRINT    ALL DONE.
02A 1C 2A   JCN A1 END    .
C *****
C
C PRINT OUT PROM ADDRESS-CONTENTS. THEN UPDATE REG 2,3
02C 2E 20   FIM E 20     SPACE
02E 50 4E   JMS PRINT

030 A1      LD 1      FIRST ADDRESS DIGIT
031 50 46   JMS PRNT    PRINT.
033 A0      LD 0      SECOND ADDRESS DIGIT
034 50 46   JMS PRNT    PRINT.
036 2E 2D   FIM E 2D     '-'
038 50 4E   JMS PRINT
03A A6      LD 6      NOW FIRST DATA DIGIT
03B 50 46   JMS PRNT
03D A7      LD 7      NOW SECOND DATA DIGIT
03E 50 46   JMS PRNT

040 A6      LD 6      NOW UPDATE REG 2
041 B2      XCH 2
042 A7      LD 7      AND REG 3
043 B3      XCH 3
044 40 22   JUN PR3     RETURN TO THE LOOP
C
C *****
C
C PRINT ROUTINES:

```

PAGE

6

PROM DUMP UTILITY PROGRAM V1.0 5/16/75

			C	REGISTERS E & F CONTAIN THE CHARACTER TO BE PRINTED.
			DAA	PRINT OUT A NUMBER IF A NUMBER
046	FR	*PRNT	XCH F	
047	RF		LDM 3	PUT 3 IN REGISTER E FOR A NUMBER
048	D3		XCH F	
049	BE		JCN CO PRINT	
04A	1A 4E		INC E	OTHERWISE A LETTER
04C	6E		INC F	ADD 1 TO REG'S E & F
04D	6F		FIM 8 08	TTY OUTPUT WITH RETURN THRU DELAY SUBROUTINE
04E	28 08	*PRINT	SRC 9	
050	29		CLB	
051	F0			
052	E1		WMP	
053	50 68	*DLOOP	JMS DLY	
055	F1		CLC	
056	BE		XCH E	
057	F6		RAR	
058	BE		XCH F	
059	BF		XCH F	
05A	F6		RAR	
05B	BF		XCH F	
05C	F7		TCC	
05D	E1		WMP	
05E	79 53		ISZ 9 DLOOP	
060	50 68		JMS DLY	
062	D1		LDM 1	
063	E1		WMP	
064	50 68		JMS DLY	
066	50 68		JMS DLY	RUNS INTO DLY
068	2C 3C	*DLY	FIM C 3C	
06A	7D 6A	*L2	ISZ D L2	
06C	7C 6A		ISZ C L2	
06E	2C 3C		FIM C 3C	
070	7D 70	*L1	ISZ D L1	
072	7C 70		ISZ C L1	
074	C0		BBL 0	
075			END	

OUTPUT AND OPERATION OF THE MONITOR
SYSTEM CHANGE AND PDUP. 6/3/75.

.J

.R

.

<PDUP NOW LOADED INTO RAM.>

<CLEAR PROM MOUNTED IN SOCKET.>

. 00-00:

<THAT'S ALL THERE IS TO IT.>



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 40-7☐ 4004 ☐ 8008 ☐ 8080 ☒ 4040

(use additional sheets if necessary)

Program Title	"PRO FORMA"
Function	<p>This programme assists in the compiling of source code tapes by eliminating errors and typing mistakes.</p> <p>In the keyboard mode it will only transmit characters to the paper tape punch that are valid in the context of the system assembly language, and automatically formats individual lines and pages to suit.</p> <p>It also provides editing facilities similar to the system assembler but includes extra functions for error removal and tape interleaving.</p>
Required Hardware	<p>TTY on Ports 0 and 1</p> <p>High Speed Reader on Ports Optional.</p>
Required Software	None.
Input Parameters	As described from TTY and High Speed Reader.
Output Results	"Clean" programme tapes in an acceptable format are obtained from the paper tape punch.

Registers Modified: Bank 0:- Nos. 0 to 15 Bank 1:- Nos. 0 and 1	Maximum Subroutine Nesting Level: 7th Level
RAM Required: Up to 108 Bytes	Assembler/Compiler Used: Assembler Version 3.0
ROM Required: 1255 Bytes	Programmer: A. G. Treves.
	Company: St. Peter Street Maidstone - Kent ME16 0SP England

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only):
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer and company
2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps should **not** be included.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.


```

;
;   TREBOR SHARPS LIMITED  INTELLEC PROGRAMME
;
;PROGRAMME:- TS 001  REV:- 004
;
;   TITLE:-  PRO FORMA   COMPILED 2ND JULY 1975
;
;.....
;LABEL: CODE OPERAND: COMMENT:-----

```

```

;
*J
*7C

```

```

0000          ORG 0
0000      0A          SB0
0001      54BA      JMS LEAD
0003      54C5      JMS ULINE
0005      54E3      JMS SCLN
0007      2CFD      FIM 12,-3 AND 255 ;PRINT 3 SPACES

*S
*33C

0009      52E0      JMS SPACE
000B      540F      JMS ST1          ;PRINT PROGRAMME HEADER
000D      52A6      JMS CRLF2
000F      5440      JMS NEXT          ;PRINT 'PROGRAMME:- TS
0011      2C40      FIM 12,40H        ;SRC FOR PAM C1,P0
0013      52C0      JMS ST2          ;READ AND STORE 3 NUMBERS
0015      5457      JMS NEXT1
0017      2C50      FIM 12,50H        ;SRC FOR PAM C1,P1 - REV NO.
0019      52C0      JMS ST2
001B      52A6      JMS CRLF2
001D      54E3      JMS SCLN
001F      2CFE      FIM 12,-2 AND 255 ;PRINT 2 SPACES
0021      52E0      JMS SPACE
0023      2CF5      FIM 12,-11 AND 255
0025      2071      FIM 0,HDG3 AND 255
0027      5400      JMS BLKOT
0029      2CD3      FIM 12,-40 AND 255 ;40 LETTER TITLE
002B      5370      ST4: JMS DCODE      ;JUMP TO DECODE KED CHAP
002D      1435      JZ ST5          ;JUMP OUT IF CLPF
002F      5344      JMS WRITE        ;OTHERWISE PRINT IT
0031      7D2E      ISZ 13,ST4      ;READ OTHER LETTERS
0033      7C2E      ISZ 12,ST4
0035      5466      ST5: JMS NEXT2      ;PRINT LINE HEADER
0037      2C60      FIM 12,60H        ;SRC FOR PAM C1,P2
0039      2D          STC 12
003A      D0          LDM 0
003B      E4          WFO
003C      D1          LDM 1
003D      E5          WPI          ;CONTAINS PAGE NO.
003E      0B          SB1
003F      20C5      FIM 0,-59 AND 255 ;LINE COUNT 1ST. TIME
0041      0A          SB0
0042      4101      JUN EDIT

*11C

0044      52AA      NEVLN: JMS CRLF
0046      53AE      JMS PAG
0048      2CFB      NEXT3: FIM 12,-5 AND 255 ;SET LABEL COUNT
004A      5370      JMS DCODE      ;JUMP TO READ 1ST CHAP
004C      1444      JZ NEVLN        ;IS CRLF
004E      F8          DAC
004F      1474      JZ LAB3          ;IS SPACE
0051      FE          DAC
0052      1462      JZ LABEL        ;IS ALPHA
0054      F8          DAC

*S
*SC

0055      1462      JZ LABEL        ;IS ? OF ?

```

;
;.....
;LABEL: CODE OPERAND: COMMENT:.....

```

;
0057 F8 DAC
0058 14F5 JZ COM1 ;IS S/ COLON
005A 2404 FIM 4,04H ;EOT CODE
005C 529A JMS MATCH
005E 1C48 JNZ NEXT3
0060 4101 JUN EDIT
0062 5344 LABEL: JMS WRITE ;PRINT CHAR
0064 7D68 ISZ 13,LAB1 ;INC COUNT
0066 4C72 JUN LAB2
0068 5370 LAB1: JMS DCODE
006A F1 CLC
006E F6 PAR
006C F6 PAR
006D 1262 JC LABEL ;IS ALPHA,NUMB.? OR @
006F F5 PAL
0070 1A68 JNC LAB1 ;JUMP IF INVALID
0072 223A LAB2: FIM 2,': ' ;COLON REQD.
0074 5344 LAB3: JMS WRITE ;WRITE COLON OR SPACE
0076 AD LD 13
0077 F8 DAC
0078 ED MCH 13 ;ADD EXTRA SPACE
0079 52E8 JMS SPACE
007B 2CEE FIM 12,0EEH ;SET COUNT OF 2 FOR CODE CHECK

007D 5370 CODE: JMS DCODE
007F 1444 JZ NEWLN ;CRLF FOUND
0081 F8 DAC
0082 F8 DAC
0083 1C7D JNZ CODE ;JUMP IF NOT ALPHA
0085 5344 JMS WRITE ;PRINT ALPHA
0087 7D7D ISZ 13,CODE ;BACK FOR ANOTHER LETTER
0089 5370 CODE2: JMS DCODE
008A 1444 JZ NEWLN ;CRLF FOUND
008D F8 DAC
008E 14A1 JZ OPP ;SPACE
0090 F8 DAC
0091 14A1 JZ OPP ;ALPHA
0093 F8 DAC
0094 1489 JZ CODE2 ;? OF @ - TRY AGAIN
0096 F8 DAC
0097 1489 JZ CODE2 ;S/ COLON - TRY AGAIN
0099 F8 DAC
009A 1489 JZ CODE2 ;COLON - TRY AGAIN
009C F8 DAC
009D 14A1 JZ OPP ;NUMB
009F 4089 JUN CODE2

00A1 5344 OPP: JMS WRITE ;WRITE LAST SYMBOL
00A3 2220 FIM 2,20H ;SPACE
00A5 5344 JMS WRITE
00A7 2AD6 FIM 10,-42 AND 255 ;CHARACTER COUNT
00A9 5370 JMS DCODE
00AB 1444 JZ NEWLN ;CRLF
00AD F8 DAC
00AE 14FE JZ COM3 ;SPACE
00B0 5344 JMS WRITE
00B2 5370 JMS DCODE
00B4 1444 JZ NEWLN
00B6 F8 DAC
00B7 14C1 JZ $+10
00B9 5344 JMS WRITE ;RE-ENTER HERE IF NOT SPACE
00BE 7EB2 ISZ 11,$-9
00BD 7AB2 ISZ 10,$-11

```

.....
;LABEL: CODE OPERAND: COMMENT
.....

```

00BF 4044      JUN NEWLN
00C1 5344      JMS WRITE      ;PRINT 1ST SPACE
00C3 7BC9      ISZ 11,S+6
00C5 7AC9      ISZ 10,S+4
00C7 4044      JUN NEWLN
00C9 5370      JMS DCODE
00CB 1444      JZ NEWLN
00CD F8        DAC
00CE 1CB9      JNZ OPR+24      ;JUMP IF NOT SPACE
00D0 5344      JMS WRITE
00D2 7BD3      ISZ 11,S+6
00D4 7AD3      ISZ 10,S+4
00D6 4044      JUN NEWLN
00D8 F1        CLC
00D9 D3        LDM 3
00DA 8A        ADD 10
00DB 1CE5      JNZ COM2+2
00DD AF        LD 11
00DE BD        XCH 13
00DF 2AEC      FIM 10,-32 AND 255
00E1 DF        LDM 15
00E2 BC        XCH 12
00E3 52E0      COM2: JMS SPACE
00E5 54E3      JMS SCLN
00E7 7BED      ISZ 11,S+6
00E9 7AED      ISZ 10,S+4
00EB 4044      JUN NEWLN
00ED 5370      JMS DCODE
00EF 1444      JZ NEWLN
00F1 5344      JMS WRITE
00F3 40E7      JIN S-12

00F5 2CFA      COM1: FIM 12,-6 AND 255
00F7 2AD0      FIM 10,-48 AND 255
00F9 40E3      JIN COM2
00FB 2CF5      COM3: FIM 12,-11 AND 255
00FD 2AE0      FIM 10,-32 AND 255
00FF 40E3      JIN COM2

0101 53AF      EDIT: JMS PAC
0103 222A      FIM 2,2AM      ;THIS IS START OF PROG CYCLE
0105 5344      JMS WRITE      ;PRINT ASTERISK
0107 5293      JMS FCT10
0109 5303      JMS FCT1
010B 2C82      FIM 0,ED2 AND 255
010D F1        ED1: CLC
010E DD        LDM -3 AND 15   ;MSB OF ASCII NUMB CHAP
010F 32        ADD 2          ;MSB OF CHAP
0110 1413      JZ S+3
0112 31        JIN 0          ;JUMP IF NOT NUMB
0113 F1        CLC
0114 A3        LD 3
0115 FE        DAA
0116 1A19      JNC S + 3
0118 31        JIN 0          ;JUMP IF NOT NUMB
0119 2C00      FIM 12,00
011B 2E0A      FIM 14,10
011D 5200      JMS FCTN      ;PROCESS NUMB
011F 523F      JMS FCT5      ;CLEARS RP.S 12 & 14
0121 A3        LD 3
0122 EF        XCH 15
0123 5244      JMS FCT6      ;ADD TO PREVIOUS NUMB

```

.....
;

;
;.....
;LABEL: CODE OPERAND: COMMENT:.....

```

;
0125 5303 JMS RED1 ;WAIT FOR NEXT CHAR
0127 206F FIM 0,ED4 AND 255 ;MODIFIED ADDR FOR JUMP
0129 410D JUN ED1
012B 243A ED2: FIM 4,' ': ;JUMP HERE IF NO NIME
012D 529A JMS MATCH
012F 1C33 JNZ S + 4
0131 42B3 JUN FINAL
0133 244B FIM 4,'K' ;TEST FOR PTR OFF
0135 529A JMS MATCH
0137 1C41 JNZ S+10
0139 5303 JMS RED1 ;TEST FOR CARR. RET
013B 529B JMS CRD
013D 144E JZ S+17
013F 4180 JUN ER
0141 244A FIM 4,'J' ;TEST FOR PTR ON
0143 529A JMS MATCH
0145 1C56 JNZ S+17
0147 5303 JMS RED1
0149 529B JMS CRD
014B 1C80 JNZ ER
014D DF LDM 0FH
014E 2000 FIM 0,00
0150 21 SPC 0
0151 E7 WR3
0152 52AE JMS LF
0154 4101 JUN EDIT
0156 2424 FIM 4,'S' ;TEST FOR S
0158 529A JMS MATCH
015A 1C64 JNZ ED3
015C 283F FIM 8,3FH ;SET COUNT
015E 2AFF FIM 10,0FFH
0160 5303 JMS RED1
0162 416F JUN ED4
0164 2441 ED3: FIM 4,'A'
0166 529A JMS MATCH
0168 1C6C JNZ S + 4
016A 4044 JUN NEWLN
016C 5293 JMS R810
016E 6B INC 11
016F 523F ED4: JMS SR1
0171 2453 FIM 4,'S'
0173 529A JMS MATCH
0175 1C7A JNZ S + 5
0177 D1 LDM 1
0178 4183 JUN ED5
017A 2443 FIM 4,'C'
017C 529A JMS MATCH
017E 1483 JZ ED5
0180 223F ER: FIM 2,'?'
0182 5344 JMS WRITE
0184 52AA JMS CRLF
0186 4101 JUN EDIT
0188 E4 ED5: WR0 ;RAM C0,P1,ST0 IS SET AS
0189 F0 CLB ;1 = SKIP 0 = COPY
018A E5 WR1 ;RAM C0,R1,ST1 SET ZERO
018B 5303 JMS RED1
018D 529B JMS CRD
018F 1C80 JNZ ER
0191 52AE JMS LF
0193 41F1 JUN ED13
0195 525A ED6: JMS PTPD
0197 2400 FIM 4,00
0199 529A JMS MATCH
;.....

```

.....
; LABEL: CODE OPERAND: COMMENT

```

;
019B 1CA4          JNZ ED7          ;JUMP IF NOT NULL
019D 528F          JMS SR1
019F ED           PD1             ;READ RAM CO, R1, ST1
01A0 1495          JZ ED6
01A2 4101          JUN EDIT
01A4 243B          ED7: FIM 4, 3
01A6 529A          JMS MATCH
01A8 2000          FIM 0, 00
01AA 1CB8          JNZ ED9
01AC 525A          ED8: JMS PTRD
01AE 240A          FIM 4, 0AH
01B0 529A          JMS MATCH
01B2 1CAC          JNZ ED8
01B4 4195          JUN ED6
01B6 525A          JMS PTRD
01B8 242A          ED9: FIM 4, 2AH      ;LOOK FOR ASTERISK
01BA 529A          JMS MATCH
01BC 14AC          JZ ED8
01BE 21           SFC 0
01BF A2           LD 2
01C0 EC           UPN
01C1 61           INC 1
01C2 21           SFC 0
01C3 A3           LD 3
01C4 E0           UPN
01C5 71C3         ISZ 1, 3+3
01C7 60           INC 0
01C8 240A          FIM 4, 0AH
01CA 529A          JMS MATCH
01CC 1CB6          JNZ ED9-2
01CE 528F          JMS SR1
01D0 EC           RD0
01D1 1CED          JNZ ED12
01D3 2000          FIM 0, 00
01D5 21           ED10: SFC 0
01D6 EC           PDM
01D7 E2           MCH 2
01D8 61           INC 1
01D9 21           SFC 0
01DA E9           PDM
01DB E3           MCH 3
01DC 240A          FIM 4, 0AH
01DE 529A          JMS MATCH
01E0 14E9          JZ ED11
01E2 5344          JMS WRITE
01E4 71E7          ISZ 1, 3+3
01E6 60           INC 0
01E7 41D5          JUN ED10
01E9 5344          ED11: JMS WRITE
01EB 53AB          JMS PAG
01ED 528F          ED12: JMS SR1
01EF D1           LDM 1
01F0 E5           UP1             ;WRITE RAM CO, R1, ST1
01F1 2CFF          ED13: FIM 12, 0FFH
01F3 2EFF          FIM 14, 0FFH
01F5 5244          JMS FCT6
01F7 1295          JC ED6
01F9 4101          JUN EDIT

```

;
;PROGRAMME:- TS 001 REV:- 004 PAGE NUMBER:- 06

;
;LABEL:LCODELQSSANDL:CONCEPT

;
0200 FCTN EQU 200H
023F FCT5 EQU 23FH
0244 FCT6 EQU 244H
025A PTRD EQU 25AH
028F SF1 EQU 28FH
0293 F810 EQU 293H
0298 CPD EQU 298H
029A MATCH EQU 29AH
02A6 CPLF2 EQU 2A6H
02AA CPLF EQU 2AAH
02AB LF EQU 2ABH
02E3 FINAL EQU 2E3H
02C0 ST2 EQU 2C0H
02E0 SPACE EQU 2E0H
0303 PED1 EQU 303H
0344 WHITE EQU 344H
0370 DCODE EQU 370H
03AB PAG EQU 3ABH
0400 BLKOT EQU 400H
040F ST1 EQU 40FH
0440 NEXT EQU 440H
0457 NEXT1 EQU 457H
0466 NEXT2 EQU 466H
0471 HDG3 EQU 471H
04BA LEAD EQU 4BAH
04C5 WLINE EQU 4C5H
04E3 SCLN EQU 4E3H

;0000

END

:1

:2

;
;

; TREEOP SHARPS LIMITED INTELLEC PROGRAMME

;PROGRAMME:- TS 002 REV:- 104

; TITLE:- PRO FORMA PT. 2. 2.JUL.1975

.....
; LABEL: CODE OPERAND: COMMENT

; *J
*S
*S
*S
*S
*S
*C

*C

0200

ORG 200H

*174C

0200	2000	FCTN:	FIM 0,00
0202	2400		FIM 4,00
0204	2600		FIM 6,00
0206	5224	FCT1:	JMS FCT3
0208	1A17		JNC FCT2
020A	F1		CLC
020B	A7		LD 7
020C	8F		ADD 15
020D	B7		XCH 7
020E	A6		LD 6
020F	8E		ADD 14
0210	B6		XCH 6
0211	A5		LD 5
0212	8D		ADD 13
0213	E5		XCH 5
0214	A4		LD 4
0215	8C		ADD 12
0216	B4		XCH 4
0217	5231	FCT2:	JMS FCT4
0219	7006		ISZ 0, FCT1
021B	A4		LD 4
021C	B8		XCH 8
021D	A5		LD 5
021E	B9		XCH 9
021F	A6		LD 6
0220	BA		XCH 10
0221	A7		LD 7
0222	BE		XCH 11
0223	C0		RBL 0
0224	A8	FCT3:	LD 8
0225	F6		RAP
0226	B8		XCH 8
0227	A9		LD 9
0228	F6		RAP
0229	B9		XCH 9
022A	AA		LD 10
022B	F6		RAP
022C	BA		XCH 10
022D	AE		LD 11
022E	F6		RAP
022F	BE		XCH 11
0230	C0		RBL 0
0231	F1	FCT4:	CLC
0232	AF		LD 15
0233	F5		RAL
0234	EF		XCH 15

;P7 + P15 = P7

;P6 + P14 = P6

;P5 + P13 = P5

;P4 + P12 = P4

;THIS S/R SHIFTS 1 PLACE

;RIGHT REGP.S 8 TO 11

;
;LABEL: CODE OPERAND: COMMENT

```

0235 AE LD 14
0236 F5 RAL
0237 EE XCH 14
0238 AD LD 13
0239 F5 PAL

023A BD XCH 13
023B AC LD 12
023C F5 PAL
023D EC XCH 12
023E C0 BEL 0
023F 2C00 FCT5: FIM 12,00
0241 2E00 FIM 14,00
0243 C0 BEL 0
0244 F1 FCT6: CLC
0245 AE LD 11
0246 8F ADD 15
0247 EE XCH 11
0248 AA LD 10
0249 8E ADD 14
024A BA XCH 10
024B A9 LD 9
024C 8D ADD 13
024D B9 XCH 9
024E A8 LD 8
024F 8C ADD 12
0250 B8 XCH 8
0251 C0 BEL 0
0252 535F PTRD1: JMS SBR0
0254 EC PDC
0255 F6 RAR
0256 F7 TCC
0257 E6 WR2 ;RAM C0,P0,ST2 SET TO ST0
0258 425E JUN PT1 ; BIT 0 ONLY
025A 535F PTRD: JMS SBR0
025C D1 LDM 1
025D E6 WR2 ;RAM C0,P0,ST2 SET 1
025E EF PT1: RD3 ;RAM C0,P0,ST3
025F 1C64 JNZ PT2 ; SET = PTR RESET = TY
0261 D1 LDM 1
0262 4300 JUN PEAD
0264 2240 PT2: FIM 2,40H
0266 23 SPC 2
0267 D0 LDM 0
0268 E2 WFR
0269 D8 LDM 8
026A E2 WFR
026E 2600 FIM 6,00
026D EA PT3: PDP
026E F5 RAL
026F 1A79 JNC PT4
0271 766D ISZ 6,PT3
0273 776D ISZ 7,PT3
0275 736D ISZ 3,PT3
0277 01 HLT ;BUT JUMP ER IF FOR 4004
0278 00 NOP
0279 D6 PT4: LDM 6
027A B4 XCH 4
027B 25 SPC 4
027C EA PDR
027D F4 CMA
027E B3 XCH 3
027F 64 INC 4

```


;.....
;LABEL: CODE OPERAND: COMMENT

```

0280 25          SPC 4
0281 EA          RDP
0282 F4          CMA
0283 F5          PAL
0284 F1          CLC
0285 F6          PAR
0286 B2          XCH 2
0287 535F        JMS SBP0
0289 EE          RD2
028A 1C3E        JNZ S+4
028C 4344        JUN WPRTE
028E C0          BBL 0

```

```

028F 2410        SR1: FIM 4,10H
0291 25          SPC 4
0292 C0          BBL 0

```

```

0293 2800        RS10: FIM 8,00H
0295 2A09        FIM 10,00H
0297 C0          BBL 0
0298 240D        GPD: FIM 4,0DH

```

```

029A F1          MATCH: CLC          ;THIS S/R MATCHES PP 2 TO
029B A3          LD 3              ;REG. PP. 4
029C 95          SUB 5

```

```

029D 14A0        JZ S+3
029F C1          M1: BBL 1          ;UNMATCHED EXIT
02A0 F1          CLC
02A1 A2          LD 2
02A2 94          SUB 4
02A3 1C9F        JNZ M1
02A5 C0          BBL 0

```

```

02A6 52AA        CPLF2: JMS CPLF
02A8 54E3        JMS SCLN
02AA 220D        CPLF: FIM 2,0DH
02AC 5344        JMS WPRTE
02AE 220A        LF: FIM 2,0AH
02B0 5344        JMS WPRTE
02B2 C0          BBL 0

```

```

02B3 C0          FINAL: SR1
02B4 52AA        JMS CPLF
02B6 71B4        ISZ 1,S-2
02B8 70B4        ISZ 0,S-4
02BA 0A          SBC
02BB 54C5        JMS ILINE
02BD 54FA        JMS LEAD
02BF 01          HLT

```

```

02C0 54D6        ST2: JMS NUMPD
02C2 A3          LD 3
02C3 E4          WPO
02C4 5344        JMS WPRTE
02C6 54B6        JMS WPRPD
02C8 A3          LD 3
02C9 E5          WPR1
02CA 5344        JMS WPRTE
02CC 54D6        JMS NUMPD
02CE A3          LD 3
02CF E6          WPR2
02D0 5344        JMS WPRTE
02D2 C0          BBL 0

```

;
; LABEL CODE OPERAND COMMENT

```

;
;A
02E0                                ORG 2E0H

02E0    2220    SPACE: FIM 2,20H
02E2    5344    JMS WRITE
02E4    7DE0    ISZ 13,SPACE
02E6    7CE0    ISZ 12,SPACE
02E8    C0      BBL 0
;SC

0300                                ORG 300H

0300    DF      READ: LDM 15
0301    4304    JUN $+3
0303    D0      RED1: LDM 0
0304    2600    FIM 6,00
0306    27      SPC 6
0307    E6      WR2
0308    2640    FIM 6,40H
030A    27      SRC 6
030B    E1      WMP ;ADVANCE READER
030C    D8      LDM 8
030D    B4      XCH 4 ;REG4 - BIT COUNTER
030E    535F    JMS SBR0
0310    EA      T10: RDR
0311    F6      RAR ;SHIFT TO CARRY
0312    1A10    JNC T10 ;LOOP WAITING FOR START BIT
0314    5369    JMS SBR1 ;4.55MSEC DELAY
0316    27      SRC 6
0317    E1      WMP ;TURN READER OFF
0318    2F      SPC 14
0319    E5      WR1
031A    EE      PD2
031B    1C1F    JNZ T11
031D    ED      PD1
031E    E1      WMP
031F    5363    T11: JMS SBR2
0321    EA      RDR ;INPUT DATA
0322    F4      CMA ;COMP ACC
0323    E5      WR1
0324    EE      PD2
0325    1C29    JNZ T12
0327    ED      PD1
0328    E1      WMP ;ECHO DATA BIT
0329    ED      T12: RDR
032A    F6      RAR ;BIT TO LINK
032B    A2      LD 2 ;GET UPPER NIBBLE
032C    F6      RAR ;SHIFT IN CARRY
032D    B2      XCH 2 ;SAVE UPPER NIBBLE
032E    A3      LD 3 ;GET LOWER NIBBLE
032F    F6      RAR ;SHIFT IN CARRY
0330    B3      XCH 3 ;SAVE LOWER NIBBLE
0331    741F    ISZ 4,T11 ;GET ALL 8 BITS
0333    5363    JMS SBR2 ;STOP BIT 1
0335    EE      RD2
0336    1C3A    JNZ T13
0338    D1      LDM 1
0339    E1      WMP ;ECHO STOP BIT 1
033A    5363    T13: JMS SBR2
033C    5369    JMS SBR1 ;4.55MSEC DELAY
033E    B2      XCH 2 ;ELIMINATE PARITY
033F    F5      PAL
;
; .....

```

.....
 ; LABEL: CODE OPERAND: COMMENT:

```

;
0340 F1 CLC
0341 F6 RAR
0342 B2 XCH 2
0343 C0 BBL 0

0344 535F WRITE: JMS SBR0
0346 E1 WMP
0347 D8 LDM 8
0348 B4 XCH 4 ;REG 4 = BIT COUNT
0349 5363 T01: JMS SBR2 ;9.09 MESEC DELAT
034B F1 CLC
034C B2 XCH 2 ;SHIFT 8 BITS RT
034D F6 RAR
034E B2 XCH 2
034F B3 XCH 3
0350 F6 RAR
0351 B3 XCH 3
0352 F7 TCC ;CARRY TO ACC
0353 E1 WMP
0354 7449 ISZ 4,T01 ;CONTINUE LOOPING
0356 5363 JMS SBR2 ;9.09 MS DELAY
0358 D1 LDM 1 ;STOP BIT 1
0359 E1 WMP
035A 5363 JMS SBR2
035C 5363 JMS SBR2 ; MORE DELAYS!
035E C0 BBL 0 ;RETURN
035F 2E00 SBR0: FIM 14,0
0361 2F SRC 14
0362 C0 BBL 0
0363 2E3C SBR2: FIM 14,60 ;9.09 MSEC DELAY
0365 7F65 L2: ISZ 15,L2
0367 7E65 ISZ 14,L2
0369 2E3C SBR1: FIM 14,60
036B 7F6B L1: ISZ 15,L1
036D 7E6B ISZ 14,L1
036F C0 BBL 0

0370 5300 DCODE: JMS PEAD
0372 A2 LD 2 ;M.S.B. OF CHAP TO ACC
0373 1C80 JNZ $ + 13 ;TEST FOR CP AND LF IF ZERO
0375 F1 CLC
0376 DA LDM 0AH ;VALUE OF LINE FFED
0377 93 SUB 3
0378 147F JZ $ + 7 ;JUMP IF LF FOUND
037A F1 CLC
037B DD LDM 0DH ;VAL OF CP
037C 93 SUB 3
037D 1C83 JNZ $+11 ;INVALID CHAT IF JUMP
037F C0 BBL 0 ;CRLF EXIT
0380 F8 DAC
0381 F8 DAC
0382 1C89 JNZ $ + 7 ;TEST MSB FOR SP OR OTHER SGV
0384 A3 LD 3
0385 1C83 JNZ $ + 3
0387 C1 BBL 1 ;SPACE FOUND
0388 C3 BBL 3 ;OTHER SIGN
0389 F8 DAC
038A 1C9C JNZ $ + 13
038C D5 LDM 5 ;TESTING FOR NUMB OF SGV
038D 83 ADD 3 ;ADD 5+1 TO CAUSE O/FLOW
038E 1291 JC $ + 3 ;IF NOT NUMBERED
0390 C6 BBL 6
0391 1C94 JNZ $ + 3 ; TEST FOR :

```

.....
 ; LABEL: CODE OPERAND: COMMENT

```

0393 C5 BBL 5 ;COLON FOUND
0394 F8 DAC
0395 1C98 JNZ $ + 3 ;TEST S/COLON
0397 C4 BBL 4 ;S/COLON FOUND
0398 92 SUB 2
0399 1C88 JNZ $ - 17 ;JUMP IF NOT ?
039B C3 BBL 3
039C F8 DAC
039D 1CA3 JNZ $ + 6 ;TRY MSB'S FOR 6 OF ALPHA
039F A3 LD 3 ;LSB'S
03A0 149B JZ $ - 5 ;JUMP IF 0
03A2 C2 BBL 2
03A3 F8 DAC ;MSB'S AGAIN
03A4 1C70 JNZ DCODE ;TRY ANOTHER IF INVALID
03A6 D4 LDM 4 ;Z IS AT ADDR. 5A
03A7 83 ADD 3
03A8 1235 JC $ - 35 ;OTHER SIGN
03AA C2 BBL 2 ;WAS ALPHABETICAL

```

```

03AE 02 PAG: SB1
03AC 71B2 ISZ 1,$+6
03AE 70B2 ISZ 0,$+4
03E0 11E4 JNT $+4
03E2 0A SB0
03E3 C0 BBL 0
03E4 20C1 FIM 0,-63 AND 255
03E6 0A SB0
03E7 54C5 JMS ULINE
03E9 5440 JMS NEXT ;CALL PROG NO
03EE 2C40 FIM 12,40H ;FOR SPC IN S/P
03ED 53E9 JMS PAGE2
03EF 5457 JMS NEXT1 ;CALL REV
03C1 2C50 FIM 12,50H
03C3 53E9 JMS PAGE2
03C5 2CEB FIM 12,-21 AND 255
03C7 20A5 FIM 0,HDS AND 255
03C9 5400 JMS BLKOT
03CE 2C60 FIM 12,60H
03CD 2D SRC 12
03CE 2230 FIM 2,30H
03D0 F1 CLC
03D1 ED PDI
03D2 F2 IAC
03D3 FB DAA
03D4 E5 WPI
03D5 1ADB JNC PAGE1
03D7 EC RDO
03D8 F2 IAC
03D9 FB DAA
03DA E4 WPI
03DE EC PAGE1: RDO
03DC B3 XCH 3
03DD 5344 JMS WRITE
03DF 2230 FIM 2,30H
03E1 2D SPC 12
03E2 ED PDI
03E3 E3 XCH 3
03E4 5344 JMS WRITE
03E6 5466 JMS NEXT2
03E8 C0 BBL 0

```

```

03E9 2230 PAGE2: FIM 2,30H
03EB 2D SPC 12

```

.....
 ;LABEL: CODE OPERAND: COMMENT:

```

03EC EC PD0
03ED B3 XCH 3
03EE 5344 JMS WRITE
03F0 2230 FIM 2,30H
03F2 2D SPC 12
03F3 ED PD1
03F4 B3 XCH 3
03F5 5344 JMS WRITE
03F7 2230 FIM 2,30H
03F9 2D SPC 12
03FA EE PD2
03FB B3 XCH 3
03FC 5344 JMS WRITE
03FE C0 BBL 0

```

```

0400 ORG 400H

```

```

0400 32 BLKOT: FIM 2 ;THIS S/R BLOCKS OUT ASCII CHAPS STOPED
0401 5344 JMS WRITE ;AT ADDRESS STARTING AT CONT-
0403 F1 CLC ;TENTS OF PP 0, NO OF WORDS TO
0404 D1 LDM 1 ;BE IN BP 12.
0405 81 ADD 1
0406 B1 XCH 1
0407 1A0A JNC S+3
0409 60 INC 0
040A 7D00 ISZ 13,BLKOT
040C 7C00 ISZ 12,BLKOT
040E C0 BBL 0

```

```

040F 2CD6 ST1: FIM 12,-42 AND 255 ;PRINT 42 CHAPS
0411 2016 FIM 0,HDG AND 255
0413 5400 JMS BLKOT
0415 C0 BBL 0
0416 20545245 HDG: DB 'TREBOP '
      424F5220
041E 53484152 DB 'SHAPPS '
      505320
0425 40494D49 DB 'LIMITED '
      54454420
      20
042E 494E5445 DB 'INTELLEC '
      40404543
      20
0437 50524F47 DB 'PROGRAMME'
      52414D4D
      45

```

```

0440 2CF0 NEXT: FIM 12,-16 AND 255
0442 2047 FIM 0,HDG1 AND 255
0444 5400 JMS BLKOT
0446 C0 BBL 0
0447 3B50524F HDG1: DB ';PROGRAMME:- TS
      4752414D
      4D453A2D
      20545320

```

```

0457 2CF8 NEXT1: FIM 12,-8 AND 255
0459 205E FIM 0,HDG2 AND 255
045E 5400 JMS BLKOT
045D C0 BBL 0
045E 20205245 HDG2: DB ' REV:-
      563A2D20

```

0466 52A6 NEXT2: JMS CPLF2
0468 2CD7 FIM 12,-41 AND 255
046A 207C FIM 0,HDG4 AND 255
046C 5400 JMS BLKOT
046E 54C9 JMS UL0
0470 C0 BEL 0

0471 20205449 HDG3: DE ' TITLE:-
544C453A
2D2020

047C 3D4C4142 HDG4: DE '3 LABEL: CODE OPE'
454C3A20
434F4445
204F5045

;
:

;.....
;LABEL: CODE OPERAND: COMMENT

```

;
048C 52414E44 DE 'FAND ;
      203B2020
      20202020
      20202020
049C 434F4D4D DE 'COMMENT'
      454E54
04A3 0D DE 0DH ; CARRIAGE RETURN
04A4 0D DE 0DH
04A5 20202020 HDG5: DE ' PAGE NUMBER'
      20205041
      4745204E
      554D4245
04E5 523A2D20 DE 'R:- '
      20

```

;SUBROUTINES

```

04BA 20C4 LEAD: FIM 12,-60 AND 255 ;PINCHES 60 NIL&S
04EC 2200 FIM 2,0
04EE 5344 JMS WRITE
04C0 7DEE ISZ 13,S-2
04C2 7CBE ISZ 12,S-4
04C4 C0 BBL 0

04C5 52A8 ULINE: JMS CRLF2+2 ;PUNCH ;CRLF; 54 U/LINE
04C7 54E3 JMS SCLN
04C9 2CCB UL0: FIM 12,-53 AND 255 ;ENTRY IF 2CRLF NOT REQD
04CB 225F FIM 2, ' '
04CD 5344 JMS WRITE
04CF 7DCB ISZ 13,UL0+2
04D1 7CCE ISZ 12,UL0+2
04D3 52A6 JMS CRLF2
04D5 C0 BBL 0

04D6 5370 NUMPD: JMS DCODE ;THIS S/R TESTS FOR NUMERAL
04D8 F8 DAC
04D9 2D SRC 12
04DA 14E2 JZ S+8
04DC F1 CLC
04DD 240B FIM 4,11 ;COMP -1 OF NUMB FLAG
04DF 85 ADD 5
04E0 1CD6 JNZ NUMPD
04E2 C0 BBL 0

04E3 223B SCLN: FIM 2,3BH ;ASCII VALUE OF COLON
04E5 5344 JMS WRITE
04E7 C0 BBL 0

```

END

```

;
;   TREBOR SHARPS LIMITED   INTELLEC PROGRAMME
;
;PROGRAMME:- TS 123   REV:- 456
;
;   TITLE:-   PRO FORMA - DESCRIPTION AND TEST
;.....
;LABEL: CODE_OPEBAND:-----COMMENT-----
;
;J
;$C

;--TITLE AND HEADING--
;
;ONCE THE PROGRAMME HAS BEEN LOADED AND RESET
;IT WILL PUNCH 60 NULLS FOR A LEAD TAPE AND
;START TYPING THE ABOVE HEADING. AFTER TYPING
;'PROGRAMME :- TS ' IT WILL WAIT FOR 3 NUMBERS
;TO BE TYPED IN AND AGAIN WAIT AFTER 'REV :- '
;THESE NUMBERS WILL BE REPEATED ON SUCCESSIVE
;PAGE HEADERS.
;THE PROGRAMME WILL PUNCH 'TITLE' WHEN A 40
;CHARACTER STING CAN BE TYPED. A CARRIAGE
;RETURN (CR) ENDING THIS SEQUENCE PREMATURELY
;THE REMAINDER OF THE HEADING WILL THEN BE
;PUNCHED AND THE PROGRAMME ENTER THE EDIT
;PHASE AND TYPE AN ASTERISK
;PAGE HEADERS, UNLESS SUPPRESSED BY TEST
;'ON', WILL BE PUNCHED EVERY 70 LINES THUS
;GIVING A4 SIZED SHEETS AND AS THEY START
;WITH ';' WILL BE IGNORED IN TAPE EDITS
;
;--EDIT PHASE--
;
;THERE ARE 10 COMMANDS WHICH CAN BE GIVEN
;ALL WITH THE EXCEPTION OF 'A' AND ': '
;COMMANDS END WITH (CR).
;INVALID COMMANDS
;CAUSE ?, (CR), (LF), ASTERISK. TO BE PRINTED
;THE COMMANDS ARE
;
; C   WHICH COPIES 1 LINE OF TAPE
; NC  WHICH COPIES N LINES OF TAPE
; $C  WHICH COPIES ALL VALID LINES UNTIL A NULL
; J   WHICH ASSIGNS HIGH SPEED TAPE READER
; K   WHICH ASSIGNS TTY TAPE READER
; S   WHICH SKIPS 1 LINE OF TAPE
; NS  WHICH SKIPS N VALID LINES OF TAPE
; $S  WHICH SKIPS ALL VALID LINES UNTIL A NULL
; :   WHICH COMPLETES LAST PAGE, PUNCHES 60
;     NULLS AND HALTS
; A   PROGRAMME ENTERS KEYBOARD PHASE
;
;     NOTE 1. 'N' IS A MULTI DIGIT DECIMAL
;             NUMBER
;     NOTE 2. LINES WILL BE IGNORED FOR
;             COPYING AND SKIPPING IF THEY
;             CONTAIN AN ASTERISK OR IF THEY
;             START WITH ';' OR EOT
;
;
;--KEYBOARD PHASE--
;
;ALL SUBSEQUENT LINES TYPED IN AT THE KEYBOARD
;WILL BE COPIED TO THE PUNCH IF VALID, (CR)
;ENDING A LINE
;-AUTOMATIC LINE DELETE-
;IF A MISTAKE IS MADE AT ANY POINT IN A LINE
;THEN THE TYPING OF AN ASTERISK WILL CAUSE THE

```


;.....
;LABEL: CODE_OPERAND: COMMENT

; LINE TO BE IGNORED IN SUBSEQUENT COPYING OPERATIONS.

; IN ORDER TO END THE KEYBOARD PHASE EOT
(CONTROL AND 'D') SHOULD BE TYPED AT A NEW
LINE THE PROGRAMME WILL THEN REVERT BACK TO
THE EDIT PHASE

; -KEYBOARD PHASE FORMATS-

; -1ST CHARACTER IN A NEW LINE-

; (CR) OR (LF) BEGINS ANOTHER LINE
; ; WILL SPACE 6,; AND ENTER COMMENT FIELD
; (SP) WILL SPACE UP TO THE CODE FIELD

XXX

; ALPHABETIC CHARACTER, ? OR @ WILL BE PUNCHED
; AND CAUSE THE LABEL FIELD TO BE ENTERED

LABEL:

; ALL OTHER CHARACTERS APART FROM EOT ARE
; IGNORED.

; -LABEL FIELD-

LABEL:

AAAAA:

?????:

EEEE:

A1111:

A:

A:

; UP TO 5 CHARACTERS CAN BE PRINTED
; ALPHANUMERIC, ? AND @ ARE PUNCHED
; (CR) AND (LF) AND ' ; ' ARE IGNORED
; (SP) AND ' : ' ADD A COLON AND SPACE
; TO CODE FIELD

; -CODE FIELD - 1ST TWO CHARACTERS-

; (CR) AND (LF) START A NEW LINE
; OTHERWISE ONLY ALPHABETICAL CHARACTERS
; ARE PUNCHED

; -CODE FIELD - LAST CHARACTER

; ONLY ALPHANUMERIC OR (SP) PUNCHED
; AN EXTRA SPACE IS INSERTED AND THE
; OPERAND FIELD ENTERED

AAA

AA1

AA

; -OPERAND FIELD-

; 1ST CHARACTER - (SP) CAUSES COMMENT FIELD
; TO BE ENTERED, OTHERWISE CHARACTER IS
; PUNCHED

; ALL OTHER OPERAND CHARACTERS -
; IF 2 SPACES OCCUR SUCCESSIVELY THEN
; COMMENT FIELD IS ENTERED OTHERWISE

PROGRAMME:- TS 123 FEV:- 456 PAGE NUMBER:- 03

```

;LABEL: CODE OPERAND: ;----- COMMENT

```

[illegible]

; ALL CHARACTERS PRINTED

[illegible][illegible]

```
XXXXX: XXX XX'XXXXXXXX ; YYYYYY'XXXXXXXXXXXXXXXXXXXX
```

XXXXX: XXX XXXXXXXXXX ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXX: XXX XXXXXXX ;XX

```
XXXXX: XXX XXXXX ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

XXXXX: XYZ XXXXX ;XX

XXXXX: XXX XX;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXX;XXXXXXXXXXXXXXXXXXXX

YXXYX: XYY XY ; XXXXXXXXXXXXXXXXXXXXXXXXXX

[illegible]

XXXXXXXXXX; XXX

XXXXXX: XXXX X Y Y Y X X X X Z Y Y X X X X X X X X X X X X

; NOTE THAT EXTRA SPACES MAY BE ADDED BEFORE

; THE '3'

; -COMMENT FIELD-

ALL CHARACTERS ARE PINCHED UNTIL (CP) OF

; (LF)

; NOTE THAT IN OPERAND AND COMMENT FIELDS

THE LINE LENGTH IS LIMITED TO 54 CHARACTERS

SO THAT NO INFORMATION IS LOST IN PASS 2

PRINT-OUTS.

MC

✻



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 40-8☒ 4004 ☐ 4040 ☐ 8008 ☐ 8080 ☐ 3000

(use additional sheets if necessary)

Program
Title

MCS-4/40 DISSEMBLER

Function

To convert MCS-4 or MCS-40 machine code program back into mnemonic or assembly code to assist modification of program, etc.

Required
Hardware

MCS-4 with two pages of program store, plus I/O program store and I/O device.

Required
Software

Input subroutine to supply next instruction in sequence as addressed by IR.8 and 9. Output subroutine to output undefined-parity ASCII characters to T.P. or TTY etc.

Input
Parameters

Instruction returned in IR.2 and 3; IR6, 8 and 9 not to be affected. The input subroutine called from page 5 locations 2 and 3 by JUN.

Output
Results

Output characters, undefined-parity ASCII, output in IR2, 3 for output to TTY etc. Registers 0-9 to be unaffected. The output will be a program listing in normal mnemonic code format as shown in the listings of the program enclosed.

Registers Modified: 0-9	Assembler/Compiler Used: OWN
RAM Required: NONE	Programmer: A. F. Drummond
ROM Required: 512 words + I/O program	Company: Plessey Company Ltd.
Maximum Subroutine Nesting Level: 1	Address: Poole, Dorset United Kingdom

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
 2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
 3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.
 4. A source paper tape of the contributed program is required. This insures that a clear, original copy of the program is available to photo-copy for publication in a User's Library update publication.
-

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

MCS-4/40 DISSEMBLER PROGRAM.

Introduction.

Whatever kind of MCS-4 or MCS-40 system is used, it will eventually happen that a working program has to be listed in assembly, or mnemonic, code. Under normal circumstances there is no way to do this.

This program, which may be run on either MCS-4 or MCS-40 systems, will dissemble - convert machine code back into mnemonic code - a program which might be in ROM, PROM, or RAM, or input via paper tape, etc. It will dissemble using the MCS-40 instruction set, which includes the MCS-4 set. The program occupies two pages of program store, and is written for pages 4 and 5.

Output.

The output is in ASCII, but with the eighth bit not giving even parity. If parity is needed for the output device, it may be generated using the authors parity generator/checker annex program, replacing the eighth bit with it's complement if the parity happens to be odd.

The output subroutine is called via a JUN in page 5, location 0. This JUN will point to the location of the output subroutine. The subroutine may use two levels of subroutine itself, being the first call from the master program.

This subroutine must return index registers 0 to '11, decimal 0 - 9, unchanged, but may affect anything else. Index registers 2 and 3, or P.1, contain the character to be output.

Input.

The input subroutine must return the next byte in sequence of the program to be dissembled, in index registers 2 and 3, or P.1. The sequence number of this byte is available in registers '10 and '11, decimal 8 and 9, or P.4. This is essential if a FIN is to be used to access the input data.

It must not affect the contents of registers 6, '10 or '11, decimal 6, 8, or 9.

This subroutine also may call two more levels of subroutine.

Access to the input subroutine is via page 5, location 02, where a JUN points to the start of the input subroutine.

Results.

The disassembler outputs all numerical data in OCTAL, and each line is preceded by it's address.

Two-byte instructions are on one line. In the case of FIM instructions, the second byte output shows the contents of the two registers separately.

Instructions referring to register pairs use the pair terminology, that is, P.0 to P.7.

Decimal Version.

A version of the program is available in which the output data is in decimal rather than octal, and this is given in decimal since those who want it will presumably want the program in decimal!

No hexadecimal version has been written, though it would present no problems at all, since it is the authors opinion that the hexadecimal notation is an infernal machination of the devil!

Modifications.

Certain features may be omitted by simple changes in the program. These are:

- i) The 'P' notation may be dropped by changing page 4, location 166 from JCN 12 to JCN 10, and location 076 to NOP.
- ii) The two-register form of outputting the second byte of FIMs may be dropped by changing page 4, location 250 from JCN 2 to JCN 0.

In this latter case, instruction locations 274 to 345 inclusive are unused and so may be replaced with input/output subprograms, accessed via the JUNs at the top of page 5.

Typical Output.

A typical output would look like this:

```
116 FIM P3 16; 7
120 SRC P3
121 LDM 17
122 WRR
123 JCN 10 102
```

and so on.

Method.

The system works by comparing the instruction with each of the possible machine codes in turn, starting from the largest, DCL. These are stored in page 5. When the test instruction is equal to or larger than the machine code, then that is, normally, the code in question.

The three immediately succeeding locations in the lookup table are the three characters of the mnemonic code. But the most significant bit of each, that which normally is the even-parity bit, is used as a control bit. The significance of the three of them, in turn, is;

- i) There exists an alternative mnemonic code which is chosen or not according to the least significant bit of the input instruction. It also therefore specifies that a register pair is referred to by the OPA part of the instruction. e.g. FIM & SRC, FIN & JIN.
- ii) The next byte of input is data, or the instruction is a two-byte instruction. e.g. JCN, JUN etc.
- iii) The OPA part of the instruction is to be output as data, e.g. ADD 3.

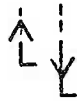
If there is a choice, then a FLAG is set before the least significant bit of the instruction is tested, so that when the second part is output it will be output in P format. This flag is examined again when the second byte is output, if it is output, so that it will be output in two four-bit halves.

Non-existent instructions are output as N/U, these being '377, '376 and '017.

A.F.Drummond,
The Plessey Co. Ltd.,
Poole, Dorset, U.K.

000 FIM P4 0; 0	Address starts at zero
002 FIM P3 0; 0	FLAG set to zero
004 LD 10	
005 XCH 2	Load Address into input
006 LD 11	of "Binary to Octal"
007 XCH 3	conversion subroutine.
010 JMS 4 346	Perform conversion.
012 LDM 3	
013 XCH 2	First character in ASCII
014 LD 7	
015 XCH 3	
016 JMS 5 000	Output character via page 5.
020 LD 4	
021 XCH 3	
022 JMS 5 000	Output second character.
024 LD 5	
025 XCH 3	
026 JMS 5 000	Output third character.
030 FIM P1 12; 0	
032 JMS 5 000	Output 'space' character.
034 JMS 5 002	Input instruction to be disassembled.
036 FIM P0 0; 4	Start location of lookup table = 04.
040 JMS 5 376	look up instruction code.
042 CLB	
043 LD 3	
044 SUB 5	Compare instruction to be disassembled with
045 XCH 7	code returned from lookup table...
046 CMC	store the least significant 4 bits in i.r.7
047 LD 2	
050 SUB 4	
051 JCN 2062	Goto 062 if difference is positive or zero.
053 INC 1	
054 INC 1	Increment P0 on to next code in look up
055 INC 1	table, and try that.
056 ISZ 1 040	Goto 040 to try next one.
060 ISZ 0 040	
062 INC 1	Increment i.r. 1 to give location of first
063 JMS 5 374	mnemonic character, and read it.
065 LD 2	
066 RAL	CY = most significant bit of character.
067 JCN 12 105	If CY = 0, character is OK - output it.
071 LDM 10	
072 XCH 6	FLAG (i.r.6) = 1000
073 CLC	
074 LD 7	Divide difference (i.r.7) by 2, because
075 RAR	this instruction may refer to a
076 XCH 7	register pair, e.g. a FIM
077 JCN 12 105	If least significant bit was 0, output char.
101 INC 1	
102 INC 1	Else, read next mnemonic and output that.

103 ISZ 1 062
 105 LD 0
 106 XCH 4
 107 LD 1
 110 XCH 5
 111 JMS 5 000
 113 LD 4
 114 XCH 0
 115 INC 5
 116 LD 5
 117 XCH 1
 120 JMS 5 374
 122 LD 2
 123 RAL
 124 TCC
 125 ADD 6
 126 XCH 6
 127 JMS 5 000
 131 LD 4
 132 XCH 0
 133 INC 5
 134 LD 5
 135 XCH 1
 136 JMS 5 374
 140 LD 2
 141 RAL
 142 TCC
 143 XCH 4
 144 JMS 5 000
 146 LD 4
 147 JCN 4 207
 151 FIM P1 12; 0
 153 JMS 5 000
 155 LD 7
 156 RAL
 157 XCH 7
 160 JCN 12 164
 162 INC 3
 163 INC 2
 164 LD 6
 165 RAL
 166 JCN 12 172
 170 FIM P1 5; 0
 172 JMS 5 000
 174 LD 7
 175 RAR
 176 XCH 3
 177 LDM 3
 200 XCH 2
 201 JMS 5 000



Goto read next mnemonic.

Store lookup table address in pair 2.

Output first mnemonic character.

Increment lookup table address to P 0 and....

Read next mnemonic code.

If most significant bit of this character = 1, then increment FLAG to 1001 or 0001.

#L.S.B. of FLAG says "This is two-byte instruction."
Output this mnemonic.

Increment lookup table address to P 0

and read third mnemonic code.

Most significant bit of this character stored in i.r. 4
#i.r.4 says "Output OPA as numeric data"
Output this third character.

If i.r.4 = 0, goto 207 - Output CRLF.

Output "space" character.

Most significant bit of difference goes to..
carry, i.r.7 = difference less M.S.B.
If M.S.B. = 0, leave last output character = space.
Else convert it to ASCII '1'

If M.S.B. of FLAG = 0, output this character, either 'space' or '1'
Else put character = ASCII 'P'
Output 'space', '1', or 'P'. as appropriate.

Difference less M.S.B. rotated back to position.

Convert to ASCII.

Output ^a least significant part of OPA.

203 LD 6	Load FLAG...
204 RAR	If least significant bit is 1..
205 JCN 2 231	then go to 231.
207 FIM P1 10;15	Else do CRLF. Load ASCII C.R. to p1
211 JMS 5 000	Output C.R.
213 FIM P1 0;12	Load ASCII L.F. to P1
215 JMS 5 000	Output L.F.
217 ISZ 11 002	Increment input instruction address..
221 ISZ 10 002	return to start if not zero.
223 JMS 5 000	go to output L.F.....
225 ISZ 11 223	sixteen of them, as paper runout.
227 JCN 10 227	Goto 227 - i.e. STOP. Now needs system reset.
231 FIM P1 12; 0	Load ASCII 'space'
233 JMS 5 000	Output 'space'.
235 ISZ 11 240	Increment next instruction address to read
237 INC 10	second byte of two-byte instruction.
240 JMS 5 002	Read that second byte.
242 LD 2	
243 XCH 4	Transfer this byte to p2
244 LD 3	
245 XCH 5	
246 LD 6	Load FLAG.
247 RAL	
250 JCN 2 274	If most significant bit = 1, output data in two parts.
252 JMS 4 346	Else convert data to octal.
254 LDM 3	
255 XCH 2	Make first character into ASCII
256 LD 7	
257 XCH 3	
260 JMS 5 000	Output first character.
262 LD 4	
263 XCH 3	Load second character.
264 JMS 5 000	Output second character.
266 LD 5	
267 XCH 3	Load third character.
270 JMS 5 000	Output third character.
272 JCN 10 207	Jump to 207, output CRLF etc.
274 LD 4	
275 RAL	Most significant bit of I.R.4 is first digit.
276 XCH 4	
277 FIM P1 12; 0	Load character to be output as ASCII 'space'.
301 JCN 12 305	If character to be output is a 0, leave as space.
303 INC 3	
304 INC 2	Else convert ASCII 'space' to '1'
305 JMS 5 000	Output first character, space or '1'.
307 LD 4	
310 RAR	Load second character..
311 XCH 3	
312 LDM 3	Make it into ASCII..
313 XCH 2	

```

314 JMS 5 000
316 FIM P1 13;13
320 JMS 5 000
322 FIM P1 12; 0
324 LD 5
325 RAL
326 XCH 5
327 JCN 12 333
331 INC 3
332 INC 2
333 JMS 5 000
335 LD 5
336 RAR
337 XCH 3
340 LDM 3
341 XCH 2
342 JMS 5 000
344 JCN 10 207
346 LDM 15
347 XCH 6
350 LD 7
351 XCH 4
352 XCH 5
353 LD 2
354 RAL
355 TCC
356 XCH 2
357 RAL
360 XCH 3
361 RAL
362 XCH 7
363 TCC
364 ADD 3
365 XCH 3
366 LD 7
367 RAR
370 XCH 7
371 ISZ 6 350
373 BBL 0
374 NOP
375 NOP
376 FIN P2
377 BBL 0

```

Output second character.

Load character as ASCII ';'.

Output semicolon.

Load character as ASCII 'space'.

Most significant bit of i.r.5 is first digit.

If it is 0, leave character as 'space'.

Else convert ASCII 'space' to '1'.

Output first character, space or '1'.

Load second character.

Make into ASCII

Output second character.

Goto 207; output CRLF.

**Start of "Binary to Octal Conversion" subroutine.

Juggle contents of i.r. 4&5, P2, around to
get most significant two bits in i.r.7,
next less significant three bits in i.r.4,
and least significant three bits in i.r.5.

Return from subroutine.

Used in process of disassembling this program taking
the input from RAM program store containing this
program.

000	106	JUN P	
001	200	LOCATION OF OUTPUT	} SET BY USER.
002	106	JUN P	
003	300	LOCATION OF INPUT	
004	376		
005	116		
006	257		
007	125		
010	375		
011	104		
012	103		
013	114		
014	374		
015	113		
016	102		
017	120		
020	373		
021	104		
022	101		
023	101		
024	372		
025	123		
026	124		
027	103		
030	371		
031	124		
032	103		
033	123		
034	370		
035	104		
036	101		
037	103		
040	367		
041	124		
042	103		
043	103		
044	366		
045	122		
046	101		
047	122		
050	365		
051	122		
052	101		
053	114		
054	364		
055	103		
056	115		
057	101		
060	363		
061	103		
062	115		

063 103
064 362
065 111
066 101
067 103
070 361
071 103
072 114
073 103
074 360
075 103
076 114
077 102
100 357
101 122
102 104
103 063
104 356
105 122
106 104
107 062
110 355
111 122
112 104
113 061
114 354
115 122
116 104
117 060
120 353
121 101
122 104
123 115
124 352
125 122
126 104
127 122
130 351
131 122
132 104
133 115
134 350
135 123
136 102
137 115
140 347
141 127
142 122
143 063
144 346
145 127
146 122

147 062
150 345
151 127
152 122
153 061
154 344
155 127
156 122
157 060
160 343
161 127
162 120
163 115
164 342
165 127
166 122
167 122
170 341
171 127
172 115
173 120
174 340
175 127
176 122
177 115
200 320
201 114
202 104
203 315
204 300
205 102
206 102
207 314
210 260
211 130
212 103
213 310
214 240
215 114
216 104
217 240
220 220
221 123
222 125
223 302
224 200
225 101
226 104
227 304
230 160
231 111
232 323

233 332
234 140
235 111
236 116
237 303
240 120
241 112
242 315
243 323
244 100
245 112
246 325
247 316
250 060
251 306
252 111
253 316
254 060
255 112
256 111
257 316
260 040
261 306
262 311
263 315
264 040
265 123
266 122
267 303
270 020
271 112
272 303
273 316
274 017
275 116
276 257
277 125
300 016
301 122
302 120
303 115
304 015
305 104
306 111
307 116
310 014
311 105
312 111
313 116
314 013
315 123
316 102

317	061	
320	012	
321	123	
322	102	
323	060	
324	011	
325	104	
326	102	
327	061	
330	010	
331	104	
332	102	
333	060	
334	007	
335	101	
336	116	
337	067	
340	006	
341	101	
342	116	
343	066	
344	005	
345	117	
346	122	
347	065	
350	004	
351	117	
352	122	
353	064	
354	003	
355	114	
356	103	
357	122	
360	002	
361	102	
362	102	
363	123	
364	001	
365	110	
366	114	
367	124	
370	000	
371	116	
372	117	
373	120	
374	062	FIN 2
375	300	BBL
376	064	FIN 4
377	300	BBL


```

000 FIM P4 0; 0
002 FIM P3 0; 0
004 LD 8
005 XCH 2
006 LD 9
007 XCH 3
008 JMS 4 229
010 LDM 3
011 XCH 2
012 LD 7
013 XCH 3
014 JMS 5 000
016 LD 4
017 XCH 3
018 JMS 5 000
020 LD 5
021 XCH 3
022 JMS 5 000
024 FIM P1 10; 0
026 JMS 5 000
028 JMS 5 002
030 FIM P0 0; 4
032 JMS 5 254
034 CLB
035 LD 3
036 SUB 5
037 XCH 7
038 CMC
039 LD 2
040 SUB 4
041 JCN 2 050
043 INC 1
044 INC 1
045 INC 1
046 ISZ 1 032
048 ISZ 0 032
050 INC 1
051 JMS 5 252
053 LD 2
054 RAL
055 JCN10 069
057 LDM 8
058 XCH 6
059 CLC
060 LD 7
061 RAR
062 XCH 7 - - Change to NOR to drop 'P' notation
063 JCN 10 069
065 INC 1

```

066	INC	1	
067	ISZ	1	050
069	LD	0	
070	XCH	4	
071	LD	1	
072	XCH	5	
073	JMS	5	000
075	LD	4	
076	XCH	0	
077	INC	5	
078	LD	5	
079	XCH	1	
080	JMS	5	252
082	LD	2	
083	RAL		
084	TCC		
085	ADD	6	
086	XCH	6	
087	JMS	5	000
089	LD	4	
090	XCH	0	
091	INC	5	
092	LD	5	
093	XCH	1	
094	JMS	5	252
096	LD	2	
097	RAL		
098	TCC		
099	XCH	4	
100	JMS	5	000
102	LD	4	
103	JCN	4	135
105	FIM	P1	10; 0
107	JMS	5	000
109	LD	7	
110	CLC		
111	DAA		
112	XCH	7	
113	JCN	10	117
115	INC	3	
116	INC	2	
117	LD	6	
118	RAL		
119	JCN	10	123 -- Change to JCN 8 123 to drop 'P' notation.
121	FIM	P1	5; 0
123	JMS	5	000
125	LD	7	
126	XCH	3	
127	LDM	3	
128	XCH	2	

129	JMS	5	000	
131	LD	6		
132	RAR			
133	JCN	2	153	
135	FIM P1	8;13		
137	JMS	5	000	
139	FIM P1	0;10		
141	JMS	5	000	
143	ISZ	9	002	
145	ISZ	8	002	
147	JMS	5	000	
149	ISZ	9	147	
151	JCN	8	151	
153	FIM P1	10; 0		
155	JMS	5	000	
157	ISZ	9	160	
159	INC	8		
160	JMS	5	002	
162	LD	2		
163	XCH	4		
164	LD	3		
165	XCH	5		
166	LD	6		
167	RAL			
168	JCN	2	188	- - Change to JCNO 188 to o/p second byte of FIM
170	JMS	4	229	as single number.
172	LDM	3		
173	XCH	2		
174	LD	7		
175	XCH	3		
176	JMS	5	000	
178	LD	4		
179	XCH	3		
180	JMS	5	000	
182	LD	5		
183	XCH	3		
184	JMS	5	000	
186	JCN	8	135	
188	LD	4		
189	CLC			
190	DAA			
191	XCH	4		
192	FIM P1	10; 0		
194	JCN	10	198	
196	INC	2		
197	INC	3		
198	JMS	5	000	
200	LD	4		
201	XCH	3		
202	LDM	3		

203 XCH 2
204 JMS 5 000
206 FIMP1 11;11
208 JMS 5 000
210 FIM P1 10; 0
212 LD 5
213 DAA
214 XCH 5
215 JCN 10 219
217 INC 2
218 INC 3
219 JMS 5 000
221 LD 5
222 XCH 3
223 LDM 3
224 XCH 2
225 JMS 5 000
227 JCN 8 135
229 FIM P0 13;12
231 CLC
232 LD 2
233 DAA
234 XCH 2
235 LD 3
236 RAL
237 XCH 3
238 LD 2
239 RAL
240 XCH 2
241 ISZ 1 232
243 LDM 0
244 XCH 2
245 DAA
246 XCH 7
247 XCH 4
248 XCH 5
249 LDM 11
250 XCH 1
251 ISZ 0 235
253 BBL 0
254 FIN P2
255 BBL 0



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 40-9☒ 4004 ☒ 4040 ☐ 8008 ☐ 8080 ☐ 3000

(use additional sheets if necessary)

Program
Title

RIGHT JUSTIFIED HEXADECIMAL DATA SHIFTER

Function

Shifts Hexadecimal Digit (Four binary bits) into RAM Right Justified.

Required
Hardware

Input Parameters: P5 points to Main Memory Character of most significant hexadecimal location +1.

ACC = Hexadecimal digit to be shifted in.

Required
Software

Output Parameters: Most significant hexadecimal digit is lost; each hexadecimal digit has been shifted to next most significant location; (ACC) has been shifted into the least significant location.

Input
ParametersOutput
Results

Registers Modified: R9, R8, R3	Assembler/Compiler Used: ASF4 (Tymshare Inc.)
RAM Required: 1 chip	Programmer: K. M. Steiner
ROM Required: 24 Bytes	Company: CAMBION
Maximum Subroutine Nesting Level: 0	Address: 5 Bay State Road Cambridge, MA 02138

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
 2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
 3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.
 4. A source paper tape of the contributed program is required. This insures that a clear, original copy of the program is available to photo-copy for publication in a User's Library update publication.
-

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

Explanation:

This routine shifts hexadecimal digits (1 4-bit byte) into RAM one by one, leaving the resulting hexadecimal number right justified.

Example:

Figure 1.

RR1: 0000 0000 0090 0A4B
(Char. 15 is left most digit).

With P5 = 00010101B and (ACC) = 0011B.
After calling shift, RR1 will have:

Figure 2.

0000 0000 0090 A43B

(Note that 9 and the B remain; they are out of the range of the shift, but that the 0 in Char. 4 of Figure 1 has been lost in Figure 2.)

The number of hex digits shifted depends on the content of index register 9. In the following program, R9 = 1100B indicating a number of length 16 bits or 4 Hex digits. If a shift of 6 hex digits is required, R9 = 1010. In other words, R9 is the two's complement of the number of hex digits to be shifted. The change can be made at location 1 of the program, changing LDM 1100B to the correct value.

```

/ SHIFT  LET X=(R11) AT SLOOP
/        LET Y=(R11)-1 AT SLOOP
/        LET Z=(R11)-2 AT SLOOP
0000 00179      SHIFT, XCH R3      /STORE(ACC)->R3
0001 00220      LDM 1100B /THIS INSTRUCTION CAN BE CHANGED
                                /ACCORDING TO THE EXPLANATION
0002 00185      XCH R9          /R9 IS INDEX REGISTER
0003 00171      LD R11
0004 00164      XCH R6          /R6 HAS INITIAL CONTENT OF R11
0005 00241      SLOOP, CLC      /MAKE R11=Z BY ADDING 1110B
0006 00222      LDM 1110B
0007 00139      ADD R11
0008 00187      XCH R11      /R11=Z,ACC=X
0009 00043      SRC P5      /POINT TO M.M.CH.ADDRESSED BY Z
0010 00235      RDM          /ACC=(M.M.CH. ADDRESSED BY Z)
0011 00167      XCH R11      /R11=(M.M.CH. ADDRESSED BY Z),ACC=Z
0012 00242      IAC          /ACC=Y
0013 00187      XCH R11      /ACC=(M.M.CH. ADDRESSED BY Z),R11=Y
0014 00043      SRC P5      /POINT TO M.M.CH. ADDRESSED BY Y
0015 00224      WRM          /M.M.CH. ADDRESSED BY Z) IS WRITTEN
0016 00121      ISZ R9 SLOOP /PROCESS IS REPEATED WITH R11=Y
00005
0018 00163      LD R3        /LOAD THE NEW HEX DIGIT INTO ACC
0019 00043      SRC P5      /POINTS TO LEAST SIGNIFICANT MEMORY
                                /LOCATION
0020 00224      WRM          /WRITES NEW HEX DIGIT TO L.S.MEM.LOC.
0021 00168      LD R8        /RETURN INITIAL VALUE TO R11
0022 00187      XCH R11
0023 00192      BBL 0000B

```

INTEL MCS-4 SIMULATOR, VERS. 2.01 (11/21/72)

FOR INSTRUCTION LIST, TYPE Q(AND CAR. RET.):

DATA FILE NAME=

CY CTR RESET

*M1

RAMWD(20 CH,0 TO F,-)=0000000000900A4B0000

← INITIAL CONTENTS OF RAM

*J24

← EXECUTE FROM LOCATION
24 (BREAK SET AT LOCATION 29)

CY CTR RESET

BREAK EXIT:

29 NDP

0 0 0003000050150000 0 1 1 69

← RESULTING CPU
REGISTERS

*E1

000000000090A43E 0000

← RESULTING CONTENTS OF RAM

SECTION 5

MATH AND NUMERICAL MANIPULATION PROGRAMS

REFERENCE NUMBER		PROGRAM
1.	4-1 Chebyshev Approximation Functions
2.	4-2 ASCII to EBCDIC Code Conversion
3.	4-6 Bit Manipulation Routine
4.	4-10 Universal Logic Subroutine
5.	4-13 Data Compare Routine
6.	4-14 4 Digit Multiply
7.	4-15 4 Digit BCD to Binary Converter
8.	4-16 Mobile Mean Program
9.	40-10 Floating Point Arithmetic Subroutine Package
10.	40-11 HEXBCD
11.	40-12 Fast Binary Multiply: Selectable Bit Precision and Constant Execute Time
12.	4-22 Fast Decimal Multiply Routine
13.	4-23 Automatic Digital Integration
14.	4-24 Multiply/Divide 8 Decimal Numbers
15.	4-25 Complement Decimal
16.	4-26 8-bit Binary to BCD Conversion



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-1☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

A Chebyshev Approximation Package

Function

The package contains approximation routines for sine, cosine, arctangent, natural logarithm (\log_e), and exponential functions (e^x). It also contains routines for performing addition, complement, multiplication, and division on 64 bit binary numbers.

Required
Hardware

None

Required
Software

None

Input
Parameters

See attached description of each subroutine.

Output
Results

See attached description of each subroutine.

Registers Modified: See attached	Maximum Subroutine Nesting Level:
RAM Required: 36 bytes	Assembler/Compiler Used: 4004 Cross Assembler
ROM Required: 768 bytes	Programmer:
	Company:

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer and company
2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps should not be included.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

A CHEBYSHEV APPROXIMATION ROUTINE
FOR THE MCS-4 COMPUTER

This routine allows evaluation of transcendental functions such as the trigonometric functions, logarithms, and exponentials. The trigonometric functions Sin, Cos, and Arctan are included in the package. Other functions can be easily added by including the proper Chebyshev coefficients in the ROM. A fixed-point arithmetic package is also provided with the Chebyshev routines, including addition, subtraction, multiplication, and division.

INTEL CORPORATION
January, 1973

1. Introduction.

This document describes the use of a Chebyshev approximation routine for the INTEL MCS-4 computer. This routine allows evaluation of transcendental functions such as the trigonometric functions, logarithms, and exponentials. The trigonometric functions Sin, Cos, and Arctan are included in the package.¹ Other functions can be easily added by including the proper Chebyshev coefficients in the ROM.

A fixed-point arithmetic package is also provided with the Chebyshev routines, including addition, subtraction, multiplication, and division.

2. Overall Organization.

The Chebyshev approximation routine and arithmetic package occupy pages 0 and 1 of the ROM, as shown in Figure 1. Chebyshev coefficients for Sin, Cos, and Arctan are located in page 2 of the ROM. Additional coefficients for other functions would normally be placed into the pages which follow, although they can be placed into page 2 if the trigonometric functions are not needed.

The Chebyshev routine is logically separated from the Chebyshev coefficients through the following mechanism. Address 0 of page 0 contains a branch to page 1, and, as currently implemented, the first instruction on page 1 is a branch to page 2, where the user's program would normally begin. The Chebyshev routine is "width" 2; that is, it uses two of the address stack elements. Thus, the routine (including Sin, Cos, and Arctan) can be called from a subroutine of the user's program.

¹Ln x and e^x have been recently added, see Appendix A.

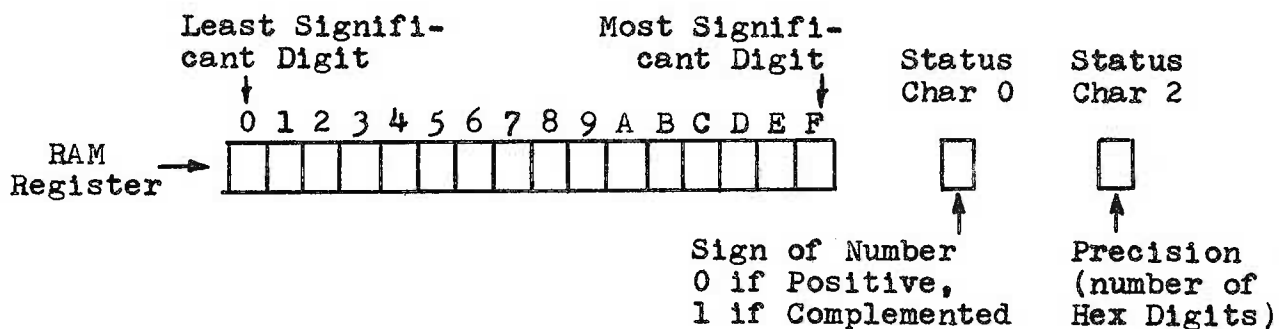


Figure 2. Organization of Data in the RAM.

All arithmetic is performed on 64-bit RAM registers. The arithmetic is binary, with a fixed number of Hexadecimal places of precision. Negative numbers are represented in 16's complement form, with the sign in status character 0 of the register. Note that the least significant digit of a number is at the left of the register, as shown in Figure 2. Thus, the number must be read "backwards." The number of Hexadecimal digits of precision is placed into status character 2.

3. The Chebyshev Routine.

The Chebyshev approximation routine evaluates the first terms of the Chebyshev series

$$f(x) = \sum_{r=0}^{\infty} a_r T_r(x)$$

according to the recurrence relation

$$b_{n+1}=b_{n+2}=0$$

$$b_j = 2xb_{j+1} - b_{j+2} + a_j \quad j=n, n-1, \dots, 0$$

$$f(x) = \frac{1}{2}(b_0 - b_2).$$

where a_0, a_1, \dots, a_n are the Chebyshev coefficients for the function being approximated. The characteristics of the Chebyshev routine are given in Table I.

In order to be of any value, the Chebyshev routine must be combined with a set of coefficients for computing various functions. Examples are given in the following paragraphs.

4. Sin, Cos, and Arctan Routines.

The Sin, Cos, and Arctan functions are implemented through the Chebyshev routine. In each case, the routines load the

TABLE I

ROUTINE NAME	CHEB
ENTRY ADDRESS	003 ₁₆ 003 ₁₀
LENGTH	307 ₁₀
REGISTERS USED	R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,RA,RB,RC, RD,RE,RF Status Characters 2 and 3 of the RAM register containing 2x are also used.
INPUT PARAMETERS	The register pair (R0,R1) addresses the ROM location of the Chebyshev coefficients. (R4,R5) addresses the RAM register containing the value 2x. (R2,R3), (R6,R7), (R8,R9) address temporary RAM registers used for b_j , b_{j+1} and b_{j+2} , respectively.
OUTPUT PARAMETERS	(R2,R3) addresses the RAM register containing the value $f(x)$. ¹
TIMING	650 - 850 msec, depending upon the number of terms evaluated in the Chebyshev series

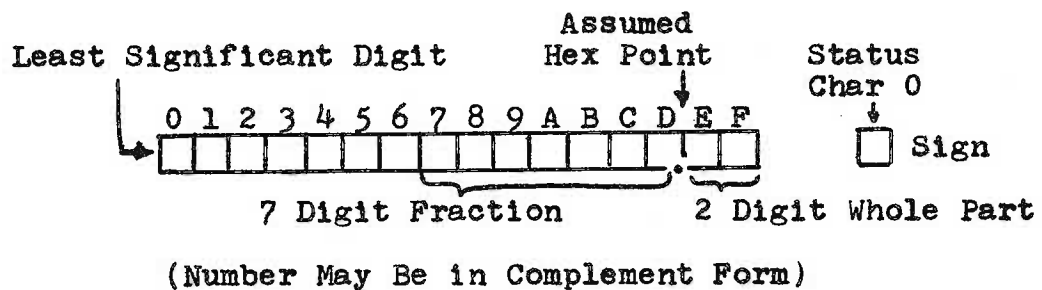


Figure 2. Form of input parameters and output results for Sin, Cos, and Arctan functions.

¹The RAM register addressed by R2, R3 upon return may differ from the RAM register addressed by R2, R3 at the time the call was executed.

base address of the appropriate table in the ROM and branch to the Chebyshev routine. The characteristics of the Sin, Cos, and Arctan routines are given in Tables II, III, and IV.

Note that the value of $2x$ must be provided to these routines in the form given in Figure 3. The result is returned in this form as well. Errors appear in the 27th bit position of the fraction in both the Sin and Cos. Error is found in the 23rd bit position in the case of Arctan. Note, however, that the result of the Arctan call must be multiplied by x to get the final result. Hence, accuracy of the result is increased in most cases.

As mentioned previously, additional functions can be implemented by filling a ROM with the appropriate Chebyshev coefficients. Note, however, that the Chebyshev routine addresses these constants through the LOADT subroutine (see address 202₁₆ of the Appendix). The LOADT must be on the page containing the constants since the ROM is loaded through a FIN instructions. If additional constants are placed into following pages (to implement $\ln x$, for example), the LOADT routine must be expanded so that it passes control to a similar routine on the page containing these new constants. This is most easily accomplished by setting a status character upon entry to the new function; this character then acts as a flag upon entry to LOADT.

TABLE II

ROUTINE NAME	SINE: computes $\sin \frac{\pi x}{2}$, $-1 \leq x \leq 1$
ENTRY POINT ADDRESS	$20B_{16}$ 523_{10}
REGISTERS USED	(same as CHEB)
INPUT PARAMETERS	Same as CHEB, except (R0,R1) is set automatically before the call to CHEB. (R4,R5) must address the RAM register containing $2x$ in the form shown in Figure 3.
OUTPUT PARAMETERS	(R2,R3) addresses the value of $\sin \frac{\pi x}{2}$ in the form shown in Figure 3.
TIMING	Approximately 650 msec per call

TABLE III

ROUTINE NAME	COSINE: computes $\cos \frac{\pi x}{2}$, $-1 \leq x \leq 1$
ENTRY POINT ADDRESS	234_{16} 564_{10}
REGISTERS USED	(Same as CHEB)
INPUT PARAMETERS	(Same as SINE)
OUTPUT PARAMETERS	(R2,R3) addresses the value of $\cos \frac{\pi x}{2}$ in the form shown in Figure 3.
TIMING	Approximately 750 msec per call.

TABLE IV

ROUTINE NAME	ARCTAN: computes $\frac{\arctan x}{x}$ for $x \neq 0$, and 1 if $x=0$.
ENTRY POINT ADDRESS	$25D_{16}$ 605_{10}
REGISTERS USED	(Same as CHEB)
INPUT PARAMETERS	(Same as SINE, COSINE)
OUTPUT PARAMETERS	(R2,R3) addresses the result of the call in the form shown in Figure 3. The result must then be multiplied by x to get the value of $\arctan x$, $-1 \leq x \leq 1$ (result is in Radians).
TIMING	Approximately 850 msec per call.

Constants can be referenced by loading the base address into (R0,R1) and successively calling the LOADT routine. Each call to LOADT brings the next pair of Hexadecimal digits into the register pair (RA,RB) (registers 10 and 11).

Finally, it should be noted that the content of the accumulator upon exit from the above routines indicates any error conditions which occur during the computations. If the routines complete normally then the accumulator is zero upon exit; the accumulator is non-zero when an error has occurred (e.g., overflow during multiplication).

5. The Arithmetic Package.

The arithmetic functions provide facilities for fixed point addition, subtraction, multiplication, and division. The characteristics of these routines are given in Tables V through VIII. The addition subroutine can be used for subtraction by first invoking the COMPLEMENT subroutine on one of the arguments. After the addition, the COMPLEMENT subroutine is invoked again on the same register to restore the operand to its original form.

Note that MULTIPLY and DIVIDE expect their arguments in signed magnitude form. Thus, negative numbers can first be complemented to get their magnitudes. Multiplication or division is then performed upon the magnitudes. The operands can then

TABLE V

ROUTINE NAME	ADDITION: computes the sum of two aligned fixed-point numbers in true or complement form.
ENTRY POINT ADDRESS	094 ₁₆ 148 ₁₀
REGISTERS USED	R2, R3, R8, R9, R13, R14, R15
INPUT PARAMETERS	(R2,R3) and (R8,R9) address two RAM registers which are either true or complement form. The numbers must be aligned so that their Hexadecimal points correspond. The sign characters must also be set, as shown in Figure 2 (it is not necessary to include the precision character).
OUTPUT PARAMETERS	(R2,R3) addresses the RAM register containing the result of the addition. ACC=0 if no overflow occurred.

TABLE VI

ROUTINE NAME	COMPLEMENT: changes a number into its 16's complement form with sign.
ENTRY POINT ADDRESS	116 ₁₆ 278 ₁₀
REGISTERS USED	R5, RA (R10), RB (R11), RD (R13)
INPUT PARAMETERS	(RA,RB) addresses a RAM register to be changed to 16's complement form. Note that two successive calls to COMPLEMENT with the same register results in the original value.
OUTPUT PARAMETERS	(RA,RB) addresses the RAM register containing the complemented number.

TABLE VII

ROUTINE NAME	MULTIPLY; computes the product of two aligned fixed-point numbers in true form.
ENTRY POINT ADDRESS	039 ₁₆ 057 ₁₀
REGISTERS USED	R2, R3, R4, R5, R6, R7, RA (R10), RD (R13), RE (R14), RF (R15).
INPUT PARAMETERS	<p>The RAM registers addressed by (R4,R5) and (R6,R7) are multiplied together. The result is placed in the register addressed by (R2,R3).</p> <p>Status character 2 of the RAM register addressed by (R4,R5) must contain the precision p to be maintained throughout the multiplication. All operands must be aligned so that there are 2p places after the Hexadecimal point. Thus, if p=7, the Hexadecimal point is assumed between characters 13 and 14 of the RAM registers.</p>
OUTPUT PARAMETERS	<p>ACC=0 upon exit if no overflow occurred.</p> <p>(R2,R3) addresses the result of the multiplication, with 2p Hexadecimal places of precision</p>
TIMING	Approximately 40 msec per call (dependent upon the operands involved).

TABLE VIII

ROUTINE NAME	DIVIDE; divides (possibly unaligned) fixed-point numbers
ENTRY POINT ADDRESS	142 ₁₆ 322 ₁₀
REGISTERS USED	R2, R3, R4, R5, R6, R7, R8, R9, RA(R10), RB(R11), RC(R12), RD(R13), RE(R14), RF.
INPUT PARAMETERS	<p>The RAM register addressed by (R2,R3) is divided by the register addressed by (R6,R7). The numbers must be in true form, with precision given in status character 2 of each operand, as shown in Figure 2. The numbers need not be aligned.</p> <p>The precision desired in the quotient must be given in status character 2 of the RAM quotient register which is addressed by (R8,R9).</p> <p>(R4,R5) must address a temporary RAM working register.</p>
OUTPUT PARAMETERS	<p>(R8,R9) addresses the RAM register containing the quotient, in the form shown in Figure 2.</p> <p>The remainder can be recovered from the register addressed by (R4,R5). This number, however, is shifted toward the high order position by an amount given by RE (R14). Thus, it must be shifted back before the remainder is correct.</p> <p>ACC=0 if no overflow or underflow occurred.</p>
TIMING	Approximately 100 msec per call, depending upon the operands involved.

be restored by re-complementing them after the multiplication or division.

6. Disclaimer.

The author has tested the above routines for a number of cases and believes that they are correct. The author, however, is not responsible for the program's correctness, but will endeavor to make corrections to errors which are found.

APPENDIX A

This Appendix documents the use of the logarithm and exponential routines which were added to the subroutines previously described. The characteristics of these routines are given in Tables IX and X.

LN computes the natural logarithm of a number s as follows:

1. determine the number t such that $s = t \cdot 2^m$ where $1 \leq t \leq 2$ (or $\frac{1}{2} \leq t \leq 1$) and m is an appropriate integer. The value of t can be determined by shifting the number s until it reaches the Hexadecimal point. The number of positions shifted corresponds to m . The value which is sent to the LN routine is then determined in step 2.

2. Compute $x = 3(t-1)/(t+1)$, and send the value $2x$ to the LN routine (or equivalently, $2x = 6(t-1)/(t+1)$).

3. The value returned by LN is the natural logarithm of t . Thus, $\text{Ln } s = \text{Ln } t + m(\text{Ln } 2)$. The natural logarithm of 2 is 0.6931 4718 0559 9453.

The argument which is sent to the EXP routine must be first multiplied by 2 (shift one position toward the most significant position).

TABLE IX

ROUTINE NAME	LN: computes the natural logarithm as described in the text.
ENTRY POINT ADDRESS	292 ₁₆ 658 ₁₀
REGISTERS USED	(same as CHEB)
INPUT PARAMETERS	Same as CHEB, except (R0,R1) is set automatically before the call to CHEB (R4,R5) must address a RAM register containing 2x in the form shown in Figure 3 (2x is computed as shown in the text)
OUTPUT PARAMETERS	(R2,R3) address the value of the natural logarithm of t (see text)
TIMING	approximately 500 msec

TABLE X

ROUTINE NAME	EXP: computes e^x $-1 \leq x \leq 1$
ENTRY POINT ADDRESS	2B5 ₁₆ 693 ₁₀
REGISTERS USED	(Same as CHEB)
INPUT PARAMETERS	Same as CHEB, except (R0,R1) is set automatically before the call to CHEB. (R4,R5) must address a RAM register containing the value 2x in the form shown in Figure 3.
OUTPUT PARAMETERS	(R2,R3) addresses the RAM register containing the value e^x .
TIMING	Approximately 500 msec

THE FIRST CONSTANT IN EACH OF THE TABLES IN PAGE 2 PROVIDE
 PARAMETERS TO THE CHEBYSHEV APPROXIMATION ROUTINE. THE CONSTANT
 IS OF THE FORM "NM". GIVEN THAT N' TERMS ARE TO BE COMPUTED,
 N IS 16 - N' (N' CANNOT EXCEED 16). THE CONSTANT M DETERMINES
 THE PRECISION OF THE RESULT: 2M DECIMAL PLACES ARE RETAINED IN
 THE MULTIPLICATION PROCESSES. EACH CONSTANT REPRESENTING THE
 CHEBYCHEV COEFFICIENTS ARE ENTERED IN 2'S COMPLEMENT FORM (ACTUA
 IN 16'S COMPLEMENT FORM) WHERE EACH CONSTANT IS GIVEN BY
 "Q CCCC CC ... CC CCCC S"
 WHERE Q = 0 IF THE COEFFICIENT IS ZERO. OTHERWISE Q IS THE NUMB
 OF ZEROES WHICH APPEAR IN THE LEAST SIGNIFICANT DIGITS OF THE HE
 DECIMAL CONSTANT (THIS TECHNIQUE IS USED TO ELIMINATE EXTRA TABL
 ENTRIES IN THE ROM). THE C'S IN THE ABOVE FORM REPRESENT THE
 COEFFICIENT (PLACED INTO THE ROM IN REVERSE ORDER). THE S IS A
 CHARACTER S0-S1-S2-S3 WHERE S0 = S1 = 0. S2 IS THE SIGN OF THE
 FICIENT (S2 = 0 IF IN TRUE FORM AND 1 IF COMPLEMENTED). S3 = 0
 FOLLOWING CHEBYCHEV COEFFIECIENT IS ZERO AND 1 OTHERWISE.

```

0000 00000 NOP
0001 00102 JUN CONPG
0000

/
/
/
0003 00122 CHEB, JMS LOADT
00002
0005 00252 LD 10 / T
0006 00274 XCH 12 / N
0007 00253 LD 11 / TP
0010 00045 SRC 04 / X2
0011 00346 WR2
0012 00321 LDM 1
0013 00277 XCH 15 / SUBIND
/
/
/
COMPLEMENT X2 IF IT IS NEGATIVE
/
0014 00045 SRC 04 / X2
0015 00354 RD0
0016 00024 JCN AZ NOCOP
00027
0020 00347 WR3
0021 00244 LD 04 / X2
0022 00272 XCH 10 / C
0023 00320 LDM 0
0024 00273 XCH 11 / CP
0025 00121 JMS COMPE
00026
  
```

0027	00320	NOCOP,	LDM	0	
0030	00043	ZEROB,	SRC	02	/ BJ
0031	00340		WRM		
0032	00047		SRC	06	/ BJ1
0033	00340		WRM		
0034	00147		INC	07	/ BJ1P
0035	00163		ISZ	03 ZEROB	/ BJP
	00030				
0037	00344		WR0		
0040	00043		SRC	02	/ BJ
0041	00344		WR0		
/					
/	ROTATE B (J), B (J+1), AND B (J+2)				
/					
0042	00242	RECUR,	LD	02	/ BJ
0043	00266		XCH	06	/ BJ1
0044	00270		XCH	08	/ BJ2
0045	00262		XCH	02	/ BJ
/					
/	B (J) # 2X*B(J+1)				
/					
0046	00320		LDM	0	
0047	00043	ZEROJ,	SRC	02	/ BJ
0050	00340		WRM		
0051	00163		ISZ	03 ZEROJ	/ BJP
	00047				
0053	00344		WR0		
0054	00047		SRC	06	/ BJ1
0055	00354		RD0		
0056	00347		WR3		
0057	00024		JCN	AZ MULT	
	00073				
/					
/	COMPLEMENT B (J+1)				
/					
0061	00246		LD	06	/ BJ1
0062	00272		XCH	10	/ C
0063	00360		CLB		
0064	00273		XCH	11	/ CP
0065	00121		JMS	COMPE	
	00026				
0067	00100		JUN	MULT	
	00073				
/					
/	MULT CAN BE CALLED AS A SUBROUTINE				
/					
0071	00322	MULTP,	LDM	2	
0072	00277		XCH	15	/ SUBIND
/					
/	FIXED POINT MULTIPLICATION ROUTINE				
/					
0073	00045	MULT,	SRC	04	/ X2
0074	00356		RD2		
0075	00276		XCH	14	/ S
0076	00256		LD	14	/ S
0077	00267		XCH	07	/ BJ1P
0100	00360		CLB		
0101	00272		XCH	10	/ T
0102	00047	MUL1,	SRC	06	/ BJ1
0103	00351		RDM		
0104	00024		JCN	AZ SKIPK	

00147				
0106 00364	CMA			
0107 00362	IAC			
0110 00275	XCH	13	/	K
0111 00252	COUNK, LD	10	/	T
0112 00263	XCH	03	/	BJP
0113 00256	LD	14	/	S
0114 00265	XCH	05	/	X2P
0115 00245	COUNT, LD	05	/	X2P
0116 00024	JCN	AZ ADDXB		
00123				
0120 00045	SRC	04	/	X2
0121 00351	RDM			
0122 00145	INC	05	/	X2P
0123 00043	ADDXB, SRC	02	/	BJ
0124 00353	ADM			
0125 00340	WRM			
0126 00163	ISZ	03 COUNT	/	BJP
00115				
0130 00032	JCN	CZ NOVER		
00133				
0132 00301	BBL	1		
0133 00245	NOVER, LD	05	/	X2P
0134 00024	JCN	AZ ENDCK		
00145				
0136 00045	CHECKX, SRC	04	/	X2
0137 00351	RDM			
0140 00024	JCN	AZ NOVED		
00143				
0142 00301	BBL	1		
0143 00165	NOVED, ISZ	05 CHECKX	/	X2P
00136				
0145 00175	ENDCK, ISZ	13 COUNK	/	K
00111				
0147 00152	SKIPK, INC	10	/	T
0150 00167	ISZ	07 MUL1 / BJ1P		
00102				

CY MUST BE ZERO AT THIS POINT

0152 00257	LD	15	/	SUBIND
0153 00366	RAR			
0154 00024	JCN	AZ CONTN		
00157				
0156 00300	BBL	0		

MAY HAVE TO COMPLEMENT BJ1 AGAIN

0157 00360	CONTN, CLB			
0160 00276	XCH	14	/	S
0161 00047	SRC	06	/	BJ1
0162 00357	RD3			
0163 00024	JCN	AZ NOCOM		
00177				
0165 00360	CLB			
0166 00347	WR3			
0167 00321	LDM	1		
0170 00276	XCH	14	/	S
0171 00246	LD	06	/	BJ1
0172 00272	XCH	10	/	C
0173 00247	LD	07	/	BJ1P

0174	00273	XCH	11	/ CP
0175	00121	JMS	COMPE	
	00026			

/

/ SET SIGN OF RESULT

/

0177	00045	NOCOM,	SRC	04	/ X2
0200	00357		RD3		
0201	00236		SUB	14	/ S
0202	00366		RAR		
0203	00367		TCC		
0204	00024		JCN	AZ POSML	
	00214				
0206	00242		LD	02	/ BJ
0207	00272		XCH	10	/ C
0210	00360		CLB		
0211	00273		XCH	11	/CP
0212	00121		JMS	COMPE	
	00026				

/

/ COMPLEMENT B (J+2) AND ADD INTO B (J)

/

0214	00250	POSML,	LD	08	/ BJ2
0215	00272		XCH	10	/ C
0216	00251		LD	09	/ BJ2P
0217	00273		XCH	11	/ CP
0220	00121		JMS	COMPE	
	00026				
0222	00100		JUN	PASTD	
	00226				

/

/ ADD CAN ALSO BE CALLED AS A SUBROUTINE

/

0224	00322	ADD,	LDM	2	
0225	00277		XCH	15	/ SUBIND
0226	00361	PASTD,	CLC		
0227	00051	ADD1,	SRC	08	/ BJ2
0230	00351		RDM		
0231	00043		SRC	02	/ BJ
0232	00353		ADM		
0233	00340		WRM		
0234	00151		INC	09	/ BJ2P
0235	00163		ISZ	03 ADD1 / BJP	
	00227				
0237	00367		TCC		
0240	00275		XCH	13	/ K
0241	00051		SRC	08	/ BJ2
0242	00354		RD0		
0243	00276		XCH	14	/ S
0244	00121		JMS	CHKSN	
	00056				
0246	00024		JCN	AZ NOVE1	
	00251				
0250	00301		BBL	1	
0251	00361	NOVE1,	CLC		
0252	00257		LD	15	/ SUBIND
0253	00366		RAR		
0254	00024		JCN	AZ CONT1	
	00257				
0256	00300		BBL	0	

/

```

/      NOW ADD CONSTANT FROM ROM
/
0257 00032      CONT1, JCN      CZ ZCOEF
      00322
0261 00122      JMS      LOADT
      00002
0263 00252      LD      10      / T
0264 00263      XCH      03      / BJP
/!!!! CLC
/

```

```

/      WE ASSUME LOADT RETURNS WITH CY#0
/
0265 00253      ADD2,   LD      11      / TP
0266 00043      SRC      02      / BJ
0267 00353      ADM
0270 00340      WRM
0271 00143      INC      03      / BJP
0272 00367      TCC
0273 00275      XCH      13      / K
0274 00122      JMS      LOADT
      00002
0276 00255      LD      13      / K
0277 00366      RAR
0300 00252      LD      10      / T
0301 00043      SRC      02      / BJ
0302 00353      ADM
0303 00340      WRM
0304 00163      ISZ      03 ADD2      / BJP
      00265
/

```

```

/      CHECK SIGNS
/
0306 00367      TCC
0307 00275      XCH      13      / K
0310 00253      LD      11      / TP
0311 00366      RAR
0312 00276      XCH      14      / S
0313 00367      TCC
0314 00277      XCH      15      / SUBIND
0315 00121      JMS      CHKSN
      00056
0317 00024      JCN      AZ NXTEM
      00324
0321 00301      BBL      1
0322 00321      ZCOEF,   LDM      1
0323 00277      XCH      15      / SUBIND
0324 00174      NXTEM,   ISZ      12 RECUR      / N
      00042
/

```

```

/      B(J):#(B(J)-B(J+2))/2
/
0326 00361      CLC
0327 00051      ADD3,   SRC      08      / BJ2
0330 00351      RDM
0331 00043      SRC      02      / BJ
0332 00353      ADM
0333 00340      WRM
0334 00151      INC      09      / BJ2P
0335 00163      ISZ      03 ADD3 / BJP
      00327
0337 00367      TCC

```

0340	00275	XCH	13	/ K
0341	00051	SRC	08	/ BJ2
0342	00354	RD0		
0343	00276	XCH	14	/ S
0344	00121	JMS	CHKSN	
	00056			
0346	00024	JCN	AZ NOVRL	
	00351			
0350	00301	SOVRL, BBL	1	

/

/ NOW SHIFT LEFT ONE POSITION TO DIVIDE BY 2

/

0351	00043	NOVRL, SRC	02	/ BJ
0352	00354	RD0		
0353	00276	XCH	14	/ S
0354	00256	LD	14	/ S
0355	00024	JCN	AZ NOCM1	
	00365			
0357	00242	LD	02	/ BJ
0360	00272	XCH	10	/ C
0361	00320	LDM	0	
0362	00273	XCH	11	/ CP
0363	00121	JMS	COMPE	
	00026			
0365	00361	NOCM1, CLC		
0366	00254	SHLET, LD	12	/ N
0367	00364	CMA		
0370	00263	XCH	03	/ BJP
0371	00043	SRC	02	/ BJ
0372	00351	RDM		
0373	00366	RAR		
0374	00340	WRM		
0375	00174	ISZ	12 SHLET	/ N
	00366			

*256

0400	00256	LD	14	/ S
0401	00024	JCN	AZ NOCM2	
	00011			
0403	00242	LD	02	/ BJ
0404	00272	XCH	10	/ C
0405	00320	LDM	0	
0406	00273	XCH	11	/ CP
0407	00121	JMS	COMPE	
	00026			

/

/ MAY HAVE TO RESTORE NEGATIVE X2

/

0411	00045	NOCM2, SRC	04	/ X2
0412	00357	RD3		
0413	00024	JCN	AZ NOCM3	
	00025			
0415	00360	CLB		
0416	00347	WR3		
0417	00244	LD	04	/ X2
0420	00272	XCH	10	/ C
0421	00320	LDM	0	
0422	00273	XCH	11	/ CP
0423	00121	JMS	COMPE	
	00026			
0425	00300	NOCM3, BBL	0	

/

/ COMPLEMENT THE NUMBER IN C
/ Z SHARES REGISTER WITH X2P
/

0426	00320	COMPE,	LDM	0	
0427	00275		XCH	13	/ K
0430	00320		LDM	0	
0431	00265		XCH	05	/ Z
0432	00372		STC		
0433	00053	CLOOP,	SRC	10	/ C
0434	00351		RDM		
0435	00364		CMA		
0436	00205		ADD	05	/ Z
0437	00340		WRM		
0440	00153		INC	11	/ CP
0441	00175		ISZ	13 CLOOP	/ K
	00033				
0443	00022		JCN	CN POSCM	
	00053				
0445	00354		RD0		
0446	00364		CMA		
0447	00366		RAR		
0450	00367		TCC		
0451	00344		WR0		
0452	00300		BBL	0	
0453	00360	POSCM,	CLB		
0454	00344		WR0		
0455	00300		BBL	0	
0456	00043	CHKSN,	SRC	02	/ BJ
0457	00354		RD0		
0460	00361		CLC		
0461	00216		ADD	14	/ S
0462	00370		DAC		
0463	00024		JCN	AZ DIFSN	/ OTHERWISE SAME SIGN
	00074				
0465	00361		CLC		
0466	00256		LD	14	/ S
0467	00235		SUB	13	/ K
0470	00024		JCN	AZ SIGNK	
	00073				
0472	00301		BBL	1	
0473	00300	SIGNK,	BBL	0	
0474	00255	DIFSN,	LD	13	/ K
0475	00366		RAR		
0476	00363		CMC		
0477	00367		TCC		
0500	00344		WR0		
0501	00300		BBL	0	

/ FIXED POINT DIVISION ROUTINE. THE REGISTER ADDRESSED
/ BY (A,AP) IS DIVIDED BY THE REGISTER ADDRESSED BY (C,CP). THE
/ RESULTING QUOTIENT IS PLACED IN REGISTER (Q,QP) WITH REMAINDER
/ IN (R,RP). THE NUMBER OF DIGITS FOLLOWING THE HEX POINT IN
/ THE OPERANDS AND QUOTIENT MUST BE STORED IN STATUS CHARACTER 2
/ OF EACH REGISTER.
/

/ FIRST ZERO OUT REGISTERS
/

0502	00360	DIVIE,	CLB		
0503	00051	ZEROR,	SRC	08	/ Q
0504	00340		WRM		

0505	00045	SRC	04	/ R
0506	00340	WRM		
0507	00145	INC	05	/ RP
0510	00171	ISZ	09 ZEROR	/ QP
	00103			

/

/ DETERMINE THE NUMBER OF POSITIONS TO SHIFT A WHILE

/ PLACING A INTO R

/

0512	00360	CLB		
0513	00273	XCH	11	/ NP
0514	00253	LOCZR, LD	11	/ NP
0515	00364	CMA		
0516	00263	XCH	03	/ AP
0517	00043	SRC	02	/ A
0520	00351	RDM		
0521	00034	JCN	AN FNDZR	
	00127			
0523	00173	ISZ	11 LOCZR	/ NP
	00114			
0525	00101	JUN	ENDIV	
	00346			

/

/ OTHERWISE DIVISION INTO ZERO

/

0527	00253	FNDZR, LD	11	/ NP
0530	00265	XCH	05	/ RP
0531	00360	CLB		
0532	00263	XCH	03	/ AP
0533	00043	COPYA, SRC	02	/ A
0534	00351	RDM		
0535	00045	SRC	04	/ R
0536	00340	WRM		
0537	00143	INC	03	/ AP
0540	00165	ISZ	05 COPYA	/ RP
	00133			
0542	00253	LD	11	/ NP
0543	00276	XCH	14	/ SHFTA
0544	00043	SRC	02	/ A
0545	00356	RD2		
0546	00213	ADD	11	/ NP
0547	00273	XCH	11	/ NP
0550	00367	TCC		
0551	00272	XCH	10	/ N

/ N NOW CONTS THE VALUE OF (PA+NA) WHICH IS USED BELOW

/ A IS SHIFTED APPROPRIATELY INTO R, NOW LOCATE HIGH ORDER

/ POSITION OF C

/

0552	00360	CLB		
0553	00275	XCH	13	/ KP
0554	00255	LOCZ1, LD	13	/ KP
0555	00364	CMA		
0556	00267	XCH	07	/ CP
0557	00047	SRC	06	/ C
0560	00351	RDM		
0561	00034	JCN	AN FNDZ1	
	00166			
0563	00175	ISZ	13 LOCZ1	/ KP
	00154			

/

OTHERWISE DIVISION BY ZERO

0565	00301	BBL	1	
0566	00255	FNDZ1, LD	13	/ KP
0567	00277	XCH	15	/ SHFTC

COMPUTE THE POSITION TO START THE QUOTIENT BY:

$10 : \# (PC+NC) + PQ - (PA + NA) / N$

WHERE PC, PQ, PA ARE THE PRECISION OF C, Q, AND A, AND

NA AND NC ARE THE NUMBER OF POSITIONS A AND C WHERE SHIFTED.

0570	00356	RD2		
0571	00215	ADD	13	/ KP
0572	00275	XCH	13	/ KP
0573	00367	TCC		
0574	00274	XCH	12	/ K
0575	00051	SRC	08	/ Q
0576	00356	RD2		
0577	00215	ADD	13	/ KP
0600	00275	XCH	13	/ KP
0601	00320	LDM	0	
0602	00214	ADD	12	/ K
0603	00274	XCH	12	/ K
0604	00361	CLC		
0605	00255	LD	13	/ KP
0606	00233	SUB	11	/ NP
0607	00271	XCH	09	/ QP
0610	00363	CMC		
0611	00254	LD	12	/ K
0612	00232	SUB	10	/ N
0613	00022	JCN	CN NOUNE	
	00216			
0615	00300	BBL	0	
0616	00034	NOUNE, JCN	AN DOVRL	
	00372			

OTHERWISE STARTING POSITION IN Q IS STORED IN QP
NOW SHIFT DIVISOR TO EXTREME RIGHT SIDE

0620	00337	LDM	15	
0621	00273	XCH	11	/ NP
0622	00246	LD	06	/ C
0623	00272	XCH	10	/ N
0624	00047	COPY1, SRC	06	/ C
0625	00351	RDM		
0626	00053	SRC	10	/ N
0627	00340	WRM		
0630	00247	LD	07	/ CP
0631	00024	JCN	AZ	PCOP1
	00242			
0633	00370	DAC		
0634	00267	XCH	07	/ CP
0635	00253	LD	11	/ NP
0636	00370	DAC		
0637	00273	XCH	11	/ NP
0640	00101	JUN	COPY1	
	00224			
0642	00253	PCOP1, LD	11	/ NP
0643	00024	JCN	AZ SUBO	
	00254			
0645	00370	DAC		

0646	00273		XCH	11		/ NP
0647	00053		SRC	10		/ N
0650	00320		LDM	0		
0651	00340		WRM			
0652	00101		JUN	PCOP1		
	00242					
0654	00360	SUB0,	CLB			
0655	00263		XCH	03		/ CNT
0656	00360	SUB1,	CLB			
0657	00265		XCH	05		/ RP
0660	00253		LD	11		/ NP
0661	00267		XCH	07		/ CP
0662	00045		SRC	04		/ R
0663	00351	SUB2,	RDM			
0664	00047		SRC	06		/ C
0665	00350		SBM			
0666	00045		SRC	04		/ R
0667	00340		WRM			
0670	00363		CMC			
0671	00145		INC	05		/ RP
0672	00045		SRC	04		/ R
0673	00167		ISZ	07	SUB2 /	CP
	00263					
0675	00245		LD	05		/ RP
0676	00024		JCN	AZ	CHKCY	
	00304					
0700	00351		RDM			
0701	00227		SUB	07		/ CP
0702	00340		WRM			
0703	00363		CMC			
0704	00022	CHKCY,	JCN	CN	CRYOT	
	00311					
/						
/		OTHERWISE INCREMENT COUNTER				
/						
0706	00143		INC	03		/ CNT
0707	00101		JUN	SUB1		
	00256					
/						
/		ADD C TO R ONCE TO RESTORE				
/						
0711	00253	CRYOT,	LD	11		/ NP
0712	00267		XCH	07		/ CP
0713	00360		CLB			
0714	00265		XCH	05		/ RP
0715	00047	ADD4,	SRC	06		/ C
0716	00351		RDM			
0717	00045		SRC	04		/ R
0720	00353		ADM			
0721	00340		WRM			
0722	00145		INC	05		/ RP
0723	00167		ISZ	07	ADD4 /	CP
	00315					
0725	00245		LD	05		/ RP
0726	00024		JCN	AZ	SKPAD	
	00334					
0730	00367		TCC			
0731	00045		SRC	04		/ R
0732	00353		ADM			
0733	00340		WRM			
/						

/ NOW WRITE COUNTER INTO Q

0734	00051	SKPAD,	SRC	08	/ Q
0735	00243		LD	03	/ CNT
0736	00340		WRM		
0737	00251		LD	09	/ QP
0740	00024		JCN	AZ ENDIV	
	00346				
0742	00370		DAC		
0743	00271		XCH	09	/ QP
0744	00173		ISZ	11 SUBO / NP	
	00254				

/ NOW SHIFT R BACK

0746	00360	ENDIV,	CLB		
0747	00273		XCH	11	/ NP
0750	00257		LD	15	/ SHFTC
0751	00267		XCH	07	/ CP
0752	00047	COPY2,	SRC	06	/ C
0753	00351		RDM		
0754	00053		SRC	10	/ N
0755	00340		WRM		
0756	00153		INC	11	/ NP
0757	00167		ISZ	07 COPY2	/ CP
	00352				

/ NOW FILL WITH ZEROES ON RIGHT

0761	00253		LD	11	/ NP
0762	00024		JCN	AZ PFILL	
	00371				
0764	00053	FILLZ,	SRC	10	/ N
0765	00360		CLB		
0766	00340		WRM		
0767	00173		ISZ	11 FILLZ	/ NP
	00364				
0771	00300	PFILL,	BBL	0	
0772	00301	DOVRL,	BBL	1	
	*512				
1000	00103	CONPG,	JUN	++256	
	00000				
1002	00072	LOADT,	FIN	10	/ T
1003	00372		STC		
1004	00320		LDM	0	
1005	00201		ADD	R1	
1006	00261		XCH	R1	
1007	00320		LDM	0	
1010	00200		ADD	R0	
1011	00260		XCH	R0	
1012	00300		BBL	0	

/ SINE ($\pi \cdot X/2$), $-1 \leq X \leq 1$, WHERE $X \cdot 2$ IS PRECOMPUTED AND PLACED
 / INTO THE RAM REGISTER ADDRESSED BY X2, AND BJ, BJ1, AND BJ2 ALL
 / ADDRESS FREE RAM REGISTERS WHICH ARE USED IN THE CHEBYSHEV APPROX
 / MATION THE RESULT IS RETURNED IN THE REGISTER ADDRESSED BY BJ
 / (THESE CONVENTIONS HOLD FOR THE OTHER FUNCTIONS WHICH FOLLOW ALS

/ SET UP THE ADDRESSES OF THE ROM CONSTANTS

1013	00040	SINE,	FIM	00 SINCN	/ SC
	00017				
1015	00100		JUN	CHEB	
	00003				

/

/

	.B				
1017	00107	SINCN,	01000111		

/

1020	00142		01100010		
1021	00377		11111111		
1022	00377		11111111		
1023	00377		11111111		
1024	00377		11111111		
1025	00362		11110010		

/

1026	00142		01100010		
1027	00351		11101001		
1030	00000		00000000		
1031	00000		00000000		
1032	00000		00000000		
1033	00000		00000000		

/

1034	00151		01101001		
1035	00040		00100000		
1036	00233		10011011		
1037	00377		11111111		
1040	00377		11111111		
1041	00362		11110010		

/

1042	00157		01101111		
1043	00255		10101101		
1044	00106		01000110		
1045	00041		00100001		
1046	00000		00000000		
1047	00000		00000000		

/

1050	00153		01101011		
1051	00363		11110011		
1052	00127		01010111		
1053	00254		10101100		
1054	00337		11011111		
1055	00362		11110010		

/

1056	00151		01101001		
1057	00124		01010100		
1060	00306		11000110		
1061	00062		00110010		
1062	00041		00100001		
1063	00000		00000000		

/

/ COSINE (PI*X/2), -1<#X<#1

/

1064	00040	COSIE,	FIM	00 COSCN	/ SC
	00070				
1066	00100		JUN	CHEB	
	00003				

/

1070	00127	COSCN,	01010111		
------	-------	--------	----------	--	--

/

1071	00150		01101000		
------	-------	--	----------	--	--

1072	00077	00111111
1073	00377	11111111
1074	00377	11111111
1075	00377	11111111
1076	00362	11110010

/		
1077	00153	01101011
1100	00160	01110000
1101	00160	01110000
1102	00000	00000000
1103	00000	00000000
1104	00000	00000000

/		
1105	00156	01101110
1106	00025	00010101
1107	00350	11101000
1110	00337	11011111
1111	00377	11111111
1112	00362	11110010

/		
1113	00153	01101011
1114	00055	00101101
1115	00172	01111010
1116	00047	00100111
1117	00000	00000000
1120	00000	00000000

/		
1121	00152	01101010
1122	00253	10101011
1123	00027	00010111
1124	00040	00100000
1125	00217	10001111
1126	00362	11110010

/		
1127	00143	01100011
1130	00264	10110100
1131	00052	00101010
1132	00241	10100001
1133	00360	11110000
1134	00000	00000000

/

ARCTAN (X), -1<#X<#1. RESULT MUST BE MULTIPLIED BY X

1135	00040	ARCTN,	FIM	00	ATNCN	/ SC
	00141					
1137	00100		JUN		CHEB	
	00003					

/		
1141	00027	ATNCN, 00010111

/		
1142	00140	01100000
1143	00166	01110110
1144	00377	11111111
1145	00377	11111111
1146	00377	11111111
1147	00362	11110010

/		
1150	00153	01101011
1151	00020	00010000
1152	00100	01000000
1153	00000	00000000

1154	00000	00000000
1155	00000	00000000
/		
1156	00157	01101111
1157	00050	00101000
1160	00116	01001110
1161	00377	11111111
1162	00377	11111111
1163	00362	11110010
/		
1164	00140	01100000
1165	00114	01001100
1166	00054	00101100
1167	00000	00000000
1170	00000	00000000
1171	00000	00000000
/		
1172	00145	01100101
1173	00153	01101011
1174	00165	01110101
1175	00257	10101111
1176	00377	11111111
1177	00362	11110010
/		
1200	00155	01101101
1201	00154	01101100
1202	00311	11001001
1203	00322	11010010
1204	00000	00000000
1205	00000	00000000
/		
1206	00150	01101000
1207	00203	10000011
1210	00064	00110100
1211	00344	11100100
1212	00357	11101111
1213	00362	11110010
/		
1214	00150	01101000
1215	00026	00010110
1216	00143	01100011
1217	00103	01000011
1220	00301	11000001
1221	00000	00000000

/ COMPUTES LN S BY FIRST EXPRESSING S AS $S \# T * 2 ** M$ WHERE
 / $1/2 < \#T < \#1$. LN S IS THEN $LN T + M * LN 2$. NOTE THAT T IS EASILY FO
 / BY "NORMALIZING" THE ARGUMENT S. M IS THE NUMBER OF POSITIONS S
 / IS SHIFTED. LN (2) # .6931 4718 0559 9450
 /

1222	00040	LN,	FIM	00 LNCON	/ SC
	00226				
1224	00100		JUN	CHEB	
	00003				

1226	00147	LNCON,	01100111
------	-------	--------	----------

1227	00146	01100110
1230	00360	11110000
1231	00000	00000000
1232	00000	00000000

1233	00000	00000000
1234	00000	00000000
/		
1235	00145	01100101
1236	00371	11111001
1237	00040	00100000
1240	00000	00000000
1241	00000	00000000
1242	00000	00000000
/		
1243	00142	01100010
1244	00213	10001011
1245	00307	11000111
1246	00000	00000000
1247	00000	00000000
1250	00000	00000000
/		
1251	00144	01100100
1252	00304	11000100
1253	00131	01011001
1254	00261	10110001
1255	00000	00000000
1256	00000	00000000
/		
1257	00140	01100000
1260	00314	11001100
1261	00300	11000000
1262	00277	10111111
1263	00240	10100000
1264	00000	00000000

EXP (X)

1265	00040	EXP,	FIM	00 EXPCN	/ SC
	00271				
1267	00100		JUN	CHEB	
	00003				
/					
1271	00147	EXPCN,	01100111		
/					
1272	00157		01101111		
1273	00040		00100000		
1274	00000		00000000		
1275	00000		00000000		
1276	00000		00000000		
1277	00001		00000001		
/					
1300	00150		01101000		
1301	00123		01010011		
1302	00000		00000000		
1303	00000		00000000		
1304	00000		00000000		
1305	00001		00000001		
/					
1306	00151		01101001		
1307	00245		10100101		
1310	00060		00110000		
1311	00000		00000000		
1312	00000		00000000		
1313	00001		00000001		

1314	00150	01101000
1315	00222	10010010
1316	00362	11110010
1317	00000	00000000
1320	00000	00000000
1321	00001	00000001
/		
1322	00153	01101011
1323	00304	11000100
1324	00223	10010011
1325	00040	00100000
1326	00000	00000000
1327	00001	00000001
/		
1330	00144	01100100
1331	00202	10000010
1332	00306	11000110
1333	00141	01100001
1334	00000	00000000
1335	00001	00000001
/		
1336	00150	01101000
1337	00350	11101000
1340	00251	10101001
1341	00133	01011011
1342	00000	00000000
1343	00001	00000001
/		
1344	00144	01100100
1345	00367	11110111
1346	00260	10110000
1347	00205	10000101
1350	00100	01000000
1351	00001	00000001
/		
1352	00151	01101001
1353	00270	10111000
1354	00214	10001100
1355	00121	01010001
1356	00041	00100001
1357	00001	00000001
/		
1360	00147	01100111
1361	00151	01101001
1362	00311	11001001
1363	00070	00111000
1364	00202	10000010
1365	00001	00000001
/		
/		
/		

.D

SYMBOL TABLE :

ADD 000224
ADD1 000227
ADD2 000265
ADD3 000327
ADD4 000715

DIFSN 000474
DIVIE 000502
DOVRL 000772
ENDCK 000145
ENCIV 000746

NOVED 000143
NOVER 000133
NOVRL 000351
NXTEM 000324
PASTO 000226

ADDXB 000123
ARCTN 001135
ATNCN 001141
CHEB 000003
CHECX 000136
CHKCY 000704
CHKSN 000456
CLOOP 000433
COMPE 000426
CONPG 001000
CONT1 000257
CONTN 000157
COPY1 000624
COPY2 000752
COPYA 000533
COSCN 001070
COSIE 001064
COUNK 000111
COUNT 000115
CRYOT 000711

EXP 001265
EXPCN 001271
FILLZ 000764
FNDZ1 000566
FNDZR 000527
LN 001222
LNCON 001226
LOADT 001002
LOCZ1 000554
LOCZR 000514
MUL1 000102
MULT 000073
MULTP 000071
NOCM1 000365
NOCM2 000411
NOCM3 000425
NOCOM 000177
NOCOP 000027
NOUNE 000616
NOVE1 000251

PCOP1 000642
PFILL 000771
POSCM 000453
POSM1 000214
RECUR 000042
SHLET 000366
SIGNK 000473
SINCN 001017
SINE 001013
SKIPK 000147
SKPAD 000734
SOVRL 000350
SUB1 000656
SUB2 000663
SUBO 000654
ZCOEF 000322
ZEROB 000030
ZEROJ 000047
ZEROR 000503



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-2☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

Code Conversion: ASCII to EBCDIC

Function

Table and routine to perform 7-bit ASCII to 8-bit EBCDIC code conversion. Full table with 128 entries provided.

Required
Hardware

Any MCS-4 system with at least 2 pages of program memory (table cannot reside in page 0 without modification).

Required
Software

None: Routine for use in larger program

Input
Parameters

Any 7-bit ASCII character in P0 MSB of R0 must be zero.

Output
Results

Corresponding 8-bit EBCDIC character in P0 (or other register pair as desired).

Registers Modified: R0, R1	Maximum Subroutine Nesting Level: None
RAM Required: None	Assembler/Compiler Used: Intel ASM4 Assembler on G.E. Timesharing
ROM Required: 128 for table; 1 for conversion instruction.	Programmer: J. Parchesky
	Company: Datatype Corporation Miami, Florida

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer and company
2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps should not be included.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

FILE30 05/14/74

ASSEMBLY OF FILE3 ON 05/14/74 AT 10:35EDT

/ASCII TO EBCDIC CODE CONVERSION TABLE
/THE TABLE OCCUPIES THE FIRST 128 LOCATIONS OF
/ANY PROGRAM MEMORY PAGE EXCEPT PAGE 0:
/THE ADDRESS OF A LOCATION IS THE ASCII CHARACTER
/THE CONTENT OF THAT LOCATION IS THE EBCDIC CHARACTER.
/THE FIN INSTRUCTION IS USED TO ACCOMPLISH THE
/ACTUAL CODE CONVERSION (SEE MCS-4 USERS MANUAL).
/TWO VARIATIONS OF THE CODE CONVERSION ROUTINE
/ARE POSSIBLE:
/ 1) THE TWO INSTRUCTIONS IMMEDIATELY FOLLOWING
/ THE TABLE ARE: CNVRT, FIN P0
/ BBL 0
/ IN WHICH CASE THE CODE CONVERSION MAY BE PERFORMED
/ ANY WHERE IN THE PROGRAM WITH A SUBROUTINE CALL
/ SUCH AS: JMS CNVRT /CONVERT ASCII TO EBCDIC
/ 2) THE FIN INSTRUCTION IS PART OF A LARGER
/ ROUTINE THAT OCCUPIES THE LAST 128 LOCATIONS
/ LOCATED ON THE SAME PAGE WITH THE TABLE.
/ FOR EXAMPLE: FIN WAS ONE BYTE OF A 67 BYTE
/ OUTPUT-TO-MAG-TAPE SUBROUTINE LOCATED ON THE SAME
/ PAGE WITH THE THE TABLE IN THE AUTHOR'S ORIGINAL
/ PROGRAM.
/THE TABLE IS GIVEN IN DECIMAL SO THAT PSEUDO OPS
/ARE NOT REQUIRED
/
*256 / (OR *512 OR *768 ETC.)
0256 00000 TABLE, 0 /NUL
0257 00001 1 /SOH
0258 00002 2 /STX
0259 00003 3 /ETX
0260 00055 55 /EOT
0261 00045 45 /ENQ
0262 00046 46 /ACK
0263 00047 47 /BEL
0264 00022 22 /BS
0265 00005 5 /HT
0266 00037 37 /LF
0267 00011 11 /VT
0268 00012 12 /FF
0269 00013 13 /CR
0270 00014 14 /SO
0271 00015 15 /SI
0272 00016 16 /DLE
0273 00017 17 /DC1
0274 00018 18 /DC2
0275 00019 19 /DC3

FILE30

05/14/74

0276 00060	60	/DC4
0277 00061	61	/NAK
0278 00050	50	/SYN
0279 00038	38	/ETB
0280 00024	24	/CAN
0281 00025	25	/EM
0282 00063	63	/SUB
0283 00039	39	/ESC
0284 00028	28	/FS
0285 00029	29	/GS
0286 00030	30	/RS
0287 00031	31	/US
0288 00064	64	/SP
0289 00079	79	/!
0290 00127	127	/"
0291 00123	123	/#
0292 00091	91	/\$
0293 00108	108	/%
0294 00080	80	/&
0295 00125	125	/'
0296 00077	77	/(
0297 00093	93	/)
0298 00092	92	/*
0299 00078	78	/+
0300 00107	107	/,
0301 00096	96	/-
0302 00075	75	/.
0303 00097	97	//
0304 00240	240	/0
0305 00241	241	/1
0306 00242	242	/2
0307 00243	243	/3
0308 00244	244	/4
0309 00245	245	/5
0310 00246	246	/6
0311 00247	247	/7
0312 00248	248	/8
0313 00249	249	/9
0314 00123	123	/:
0315 00094	94	/;
0316 00076	76	/<
0317 00126	126	/=
0318 00110	110	/>
0319 00111	111	/?
0320 00124	124	/@
0321 00193	193	/A
0322 00194	194	/B
0323 00195	195	/C
0324 00196	196	/D
0325 00197	197	/E

FILE30

05/14/74

0326	00198	198	/F
0327	00199	199	/G
0328	00200	200	/H
0329	00201	201	/I
0330	00209	209	/J
0331	00210	210	/K
0332	00211	211	/L
0333	00212	212	/M
0334	00213	213	/N
0335	00214	214	/O
0336	00215	215	/P
0337	00216	216	/Q
0338	00217	217	/R
0339	00226	226	/S
0340	00227	227	/T
0341	00228	228	/U
0342	00229	229	/V
0343	00230	230	/W
0344	00231	231	/X
0345	00232	232	/Y
0346	00233	233	/Z
0347	00074	74	/[
0348	00224	224	/\
0349	00090	90	/]
0350	00095	95	/^
0351	00109	109	/BACK ARROW
0352	00121	121	/GRAVE ACCENT
0353	00129	129	/A\LOWER CASE
0354	00130	130	/B\
0355	00131	131	/C\
0356	00132	132	/D\
0357	00133	133	/E\
0358	00134	134	/F\
0359	00135	135	/G\
0360	00136	136	/H\
0361	00137	137	/I\
0362	00145	145	/J\
0363	00146	146	/K\
0364	00147	147	/L\
0365	00148	148	/M\
0366	00149	149	/N\
0367	00150	150	/O\
0368	00151	151	/P\
0369	00152	152	/Q\
0370	00153	153	/R\
0371	00162	162	/S\
0372	00163	163	/T\
0373	00164	164	/U\
0374	00165	165	/V\
0375	00166	166	/W\

FILE30 05/14/74

0376	00167	167	/X\
0377	00168	168	/Y\
0378	00169	169	/Z\
0379	00192	192	/LEFT BRACKET
0380	00106	106	/VERTICAL LINE
0381	00208	208	/RIGHT BRACKET
0382	00161	161	/TILDE
0383	00007	7	/DEL
0384	00048	CNVRT,	FIN P0
0385	00192	BBL 0	



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-6☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

Bit Manipulation Routine

Function

AND, OR, XOR, COMPARE set unselected bits,
clear selected bits, test ones (unselected bits are set
to zero and selected ones bits are set to zero. If all
selected bits are one, result will be zero and may be
tested with JCN AZ, ACCU ZERO)

Required
Hardware

4004 and 39 instructions

Required
Software

None

Input
Parameters

As shown, the word and select bits are loaded immediate.
In most cases they will come from data memory or input
ports. The control bits should be in IR3, the instruction
in IR3P, and the word in the accumulator.

Output
Results

The result will be found in IR2.

Registers Modified: IR 0-7	Maximum Subroutine Nesting Level: One
RAM Required: None	Assembler/Compiler Used: Intellec 4 Version 1.0
ROM Required: 39 Instructions	Programmer: Paul F. Fitts
	Company: INNOVATEK ENT.

- 1 Complete Submittal Form as follows (Please print or type)
 - a Processor (check appropriate box)
 - b Program title: Name or brief description of program function
 - c Function: Detailed description of operations performed by the program
 - d Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f Input parameters: Description of register values, memory areas or values accepted from input ports
 - g Output results: Values to be expected in registers, memory areas or on output ports
 - h Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j Programmer and company
2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps should not be included.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

INNOVATEK ENT.
SMITHFIELD ROAD
MILLERTON, N. Y. 12546
914 - 373-9122

```

0:/                               /BIT MANIPULATION  MAY 29,1974
0:STRT,LDM 6
1:   XCH 3                       /CNTRL BITS TO OPR ON
2:   FIM 3P 32                   /OPN TO PERFORM
4:   LDM 5                       /WORD TO OPR ON
5:   JMS BIT
7:   SRC 1P                      /PERMIT OBSERVATION ON PANEL
8:   JUN STRT
10:  = 16
16:BIT, FIM 2P 12               /LP CNTR
18:BLP, XCH 3                   /CNTRL BITS TO ACCU
19:  RAL                        /CNTRL BIT TO LINK
20:  XCH 4                      /SAVE CNTRL BITS
21:  TCC
22:  ADD 7                      /INC ADRS
23:  XCH 1
24:  TCC
25:  ADD 6                      /PROP CARRY
26:  XCH 0
27:  LD 4                      /RESTORE CNTRL BITS
28:  XCH 3                      /WORD TO ACCU
29:  RAL                        /WORD BIT TO LINK
30:  XCH 2                      /MODIFIED WORD TO ACCU
31:  JIN 0P
32:  STC                       /CLC ON 0
33:  CMC                       /CMC ON 1  TEST ONES
34:  JCN 0 250                 /STC ON 1/0 OR SET SEL BITS/OR CMPL CNTRL BITS
36:  JCN 0 241                 /CLC ON 1/0 CLR SEL BITS/AND  TEST ZEROS
38:  JCN 0 243                 /CMC ON 1/0 XOR/ COMPARE
40:  RAL
41:  XCH 2                      /STORE MODIFIED BITS
42:  ISZ 5 BLP                 /LOOP
44:  BBL 0                     /RETURN
45:$

```

BIT MANIPULATION ROUTINE

INSTRUCTION WORD	SELECT BITS	WORD OPR ON	RESULT	FUNCTION
0010 0000	0110	0101	0010	TEST ONES
0010 0001	0110	0101	1100	COMPARE
0010 0010	0110	0101	0111	OR
0010 0011	0110	0101	1101	OR CMPL CNTRL BS
0010 0100	0110	0101	0001	SET SEL BITS
0010 0101	0110	0101	0100	AND
0010 0110	0110	0101	0011	XOR
0010 0111	0110	0101	1100	COMPARE (SAME 1)



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-10☒ 4004 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

Universal Logic Subroutines

August 2, 1974

Function

Forms logical AND, OR, XOR, $\overline{\text{XOR}}$ functions between the contents of Index Registers 0 and 1. Two entry labels are used, "AND" and "OR". Calling the subroutine with "AND" results in the logical AND in IR0 and the XOR in IR1. A subroutine call with "OR" yields the logical OR in IR0 and the $\overline{\text{XOR}}$ in IR1. Execution time: 50 machine cycles.

Required
HardwareRequired
SoftwareInput
Parameters

Operand A in Index Register 0
Operand B in Index Register 1
Carry state (0 or 1)

Output
Results

"AND"

"OR"

 $(\text{IR0}) = A \cdot B$ $(\text{IR0}) = A + B$ $(\text{IR1}) = A \oplus B$ $(\text{IR1}) = \overline{A \oplus B}$ $(\text{IR2}) = 0$ $(\text{IR2}) = 15$

Carry state unchanged

Accumulator = 0

 $(\text{IR3}) = 0$

Registers Modified: IR 0,1,2,3	Maximum Subroutine Nesting Level: 3
RAM Required: None	Assembler/Compiler Used: MAC4
ROM Required: 19 bytes	Programmer: Richard C. Lee
	Company: Boonton Electronics Corp.

4004 MACRO ASSEMBLER, VER 2.1 ERRORS = 0 PAGE 1

0000	22FB	ORR:	FIM 2,251	:IR2 = 15, IR3 = 11
0002	400D		JUN STR	
0004	320B	ANDD:	FIM 2,11	:IR2 = 0, IR3=11
0006	400D		JUN STR	
0008	A2	LOG:	LD 2	:FETCH AND/OR CONTROL
0009	F6		RAR	:ROTATE IN DATA BIT TO MSB
000A	80		ADD 0	:PERFORM LOGIC ON MSB
000B	F5		RAL	:ROTATE AND/OR TO ACC, XOR TO CY
000C	B0		XCH 0	:SAVE PARTIAL 'AND' IN IR0
000D	A1	STR:	LD 1	:FETCH OPERAND
000E	F5		RAL	:ROTATE MSB TO CY, XOR TO LSB
000F	B1		XCH 1	:SAVE PARTIAL RESULT IN IR1
0010	7303		ISZ 3,LOG	:LOOP FOR ALL BITS
0012	C0		BBL 0	:RETURNS CY UNCHANGED

END

NO PROGRAM ERRORS



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-13☒ 4004 ☐ 8008 ☐ 8080 ☐ 4040

(use additional sheets if necessary)

Program Title	DACMP (data compare)
Function	Program compares two 8-digit numbers and returns a pointer to the greater in the carry.
Required Hardware	MCS-4 4004, 1 RAM, 1 ROM.
Required Software	None
Input Parameters	IRP0 contains pointer to least significant digit of #1 number. IRP1 contains pointer to least significant digit of #2 number.
Output Results	Carry = 0 if P0 is greater or equal. <i>P0 & P1 0,1</i> Carry = 1 if P1 is greater. <i>P0 & P1 2,3</i> P0 and P1 are restored to their original values.

Registers Modified: 6, 14, 15	Maximum Subroutine Nesting Level: 3
RAM Required: 2 registers	Assembler/Compiler Used: ASF4
ROM Required: 35 locations	Programmer: Lyndon Calerdine
	Company: Communications Components

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer and company
2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps should **not** be included.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

↑
ΦF

↑
Φ7

Φ

Decimal

Fin Φ, ΦF } Assume this is what we
 Fin 2, Φ7 } need as code for locations
 1/2 no. to be compared

/DACMP THIS ROUTINE COMPARES THE ABSOLUTE MAGNITUDE OF
 /THE CONTENTS OF TWO RAM REGISTERS THE CARRY IS SET
 /TO ONE IF P1 IS GREATER SET TO 0 IF P0 IS GREATER
 /IF EQUAL THE CARRY IS SET TO 0

630	216	DACMP	LDM 6	/COMPARE UP TO EIGHT DIGITS
631	182		XCH 6	/IR 6 = LOOP COUNTER
632	161		LD 1	/SET UP RESTORE
633	190		XCH 14	/IR14=ORIGINAL IR 1
634	163		LD 3	/LOAD ORIG IR3
635	191		XCH 15	/STORE IN IR15
636	215		LDM 7	/POINT TO MOST SIG DIGIT
637	241		CLC	/CLEAR CARRY FOR ADD
638	129		ADD 1	/POINT TO MOST SIG DIGIT OF P0
639	177		XCH 1	/STORE IN P0
640	215		LDM 7	/ADD 7 TO IR 3
641	241		CLC	/CLEAR CARRY FOR ADD
642	131		ADD 3	/POINT TO MOST SIG DIGIT OF P1
643	179		XCH 3	/STORE IN IR 3
644	241	DACM,	CLC	/CLEAR CARRY FOR SUBTRACT
645	35		SRC P1 2	/SELECT P1
646	233		RDM	/READ MOST SIG DIGIT
647	33		SRC P0	/SELECT P0
648	232		SBM	/SUBTRACT MOST SIG DIGIT
649	28		JCN AN DAC	/IF NOT EQUAL, RETURN
	148			
651	163		LD 3	/LOAD POINTER
652	248		DAC	/POINT TO NEXT MOST SIG DIGIT
653	179		XCH 3	/STORE
654	161		LD 1	/POINT TO NEXT DIGIT P0
655	248		DAC	/NEXT MOST SIG DIGIT
656	177		XCH 1	/STORE
657	118		ISZ 6, DACM	/DO UP TO 8 LOOPS
	132			
659	241		CLC	/CLEAR CARRY IF EQUAL
660	174	DAC.	LD 14	/RESTORE P0
661	177		XCH 1	
662	175		LD 15	/RESTORE P1
663	179		XCH 3	/
664	192		BBL 0	/RETURN WITH 0 IN ACC

When return from sub

if carry=1, Then no in RAM Φ7 is larger
 if carry=0, Then no in RAM ΦF is larger



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-14☒ 4004 ☒ 4040 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

Four Digit by Four Digit Multiplier Subroutine

Function

Takes the four digits in the multiplicand area of RWM and multiplies them by the four digits in the multiplier area and leaves the eight digit result in a result area. The result area is not automatically cleared so that partial sums of successive products are easily accumulated. Otherwise, a ten byte subroutine to clear the result area is recommended. This result clearing subroutine is also included. Both the multiplier and multiplicand are left unchanged and can be used again repetitively. By loading four digit numbers moved in to the multiplicand. Worst case time is in the range of 25 milliseconds.

Required
Hardware

None

Required
Software

None

Input
ParametersProgram
Details

1. Uses accumulator and registers 15 and 0 thru 11.
2. RWM required is Bank 0, RAM 0 and:
Locations 0 thru 7 = LSD to MSD Result area
Locations 8 thru 11 = LSD to MSD of Multiplier
Locations 12 thru 15 = LSD to MSD of Multiplicand
3. ROM required is 62 bytes. The ten byte subroutine to clear the result area is also included for reference purposes.
4. In use it requires one subroutine call. Therefore it goes one level deeper in the stack than the subroutine itself.

Registers Modified: Accumulator and 15 and 0 thru 11	Assembler/Compiler Used: Hardware Assembler A0740 - A0743
RAM Required: Main memory work 0 - 15	Programmer: Roger Camp
ROM Required: 62 bytes	Company: Camp Instruments
Maximum Subroutine Nesting Level: One	Address: 1046 Gaskill Ames, Iowa 50010

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

Your program will be photo-copied for publication in the User's Library. Please send an original, clear, un-marked copy.

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

/**/ CLEAR RESULT AREA SUBROUTINE /**/

CLRRES, FIM 0P,0

FIM 5P,8

CLR

SRC 0P

WRM

INC 1

ISZ 11,*-3

BBL 0

/**/ MULTIPLY SUBROUTINE /**/

MLPLY, FIM 0P,0

FIM 1P,0

FIM 2P,0

FIM 3P,12

FIM 4P,8

JMS MIDIGIT

INC 1

INC 9

JMS MIDIGIT

INC 1

INC 9

JMS MIDIGIT

INC 1

INC 9

JMS MIDIGIT

BBL 0

MIDIGIT SRC 4P

RDM read 4 to acc.

JAZ END → to BBL if acc=0

CMA acc

IAC acc+1

XCH 15 acc → INDEX REG-15

ENTR3 FIM 3P,12 reg pair b=1,2

LD 1 index reg 1

XCH 3 index reg 3=1

LD 1 acc index 1

XCH 5 index reg 5=1

CLC

LDM 8 acc 8

SUB 3 acc 8 - reg 3

CMA acc

IAC acc+1

XCH 10 reg 10 → acc

ENTR2 LD 10 acc → reg 10

XCH 11 reg 11 → acc

CLB

SRC 3P

RDM

ENTR1 SRC 2P

ADM

DAA

WRM

JCZ NOCRY

TCC

INC 5

ISZ 11, ENTR1

NOCRY INC 10

INC 3

LD 3

XCH 5

ISZ 7, ENTR2

ISZ 15, ENTR3

END BBL 0

123

456

reg

8

9

A

B

#

3

2

1

0

C 1100
D 1101
E 1100
F 1111

```

0: /*** CLEAR RESULT AREA SUBROUTINE ***
0: CLPRES, FIM 0P, 0
2: FIM 50, 8
4: CLR
5: SRC 0P
6: WRM
7: INC 1
8: ISZ 11, *-3
10: BRL 0
11:
11: /*** MULTIPLY SUBROUTINE ***
11: MLPLY, FIM 0P, 0
13: FIM 1P, 0
15: FIM 2P, 0
17: FIM 3P, 12
19: FIM 4P, 8
21: JMS MIDIGIT
23: INC 1
24: INC 9
25: JMS MIDIGIT
27: INC 1
28: INC 9
29: JMS MIDIGIT
31: INC 1
32: INC 9
33: JMS MIDIGIT
35: BRL 0
36: MIDIGIT SRC 4P
37: RDM
38: JAZ FND
40: CMA
41: IAC
42: XCH 15
43: ENTR3 FIM 3P, 12
45: LD 1
46: XCH 3
47: LD 1
48: XCH 5
49: CLC
50: LDM 8
51: SUB 3
52: CMA
53: IAC
54: XCH 10
55: ENTR2 LD 10
56: XCH 11
57: CLB
58: SRC 3P
59: RDM
60: ENTR1 SRC 2P
61: ADM
62: DAA
63: WRM
64: JCZ NOCOPY
66: TCC
67: INC 5
68: ISZ 11, ENTR1
70: NOCOPY INC 10
71: INC 3
72: LD 3
73: XCH 5
74: ISZ 7, ENTR2
76: ISZ 15, ENTR3
78: END BRL 0
79:

```

reads 4 digits
using
sub
✓

reads digit



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-19☒ 4004 ☐ 8008 ☐ 8080 ☐ 4040

(use additional sheets if necessary)

Program Title	TRANSLATE HEX
Function	Translates a paper tape, represented in the hexadecimal format, back into mnemonic code.
Required Hardware	Teletype at Port 0 and 1
Required Software	None
Input Parameters	None
Output Results	TTY-output

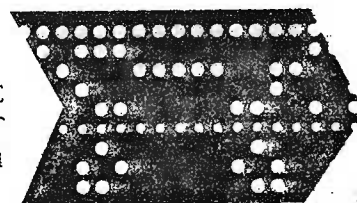
Registers Modified: 0 through 15 (all)	Maximum Subroutine Nesting Level: 3 (all)
RAM Required: None	Assembler/Compiler Used: Intellec 4 Version 1.0
ROM Required: 512 bytes	Programmer: Urs Mueller
	Company: Autophon AG

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer and company
2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps should **not** be included.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

TESTPROGRAMM FUER TRANSLATE HEX

```
:1004000002011011202121233031409050F460610E
:1004100070718090A0B0C0D0E0E1E2E3E4E5E6E7EF
:10042000E8E9EAECEDEEEFF0F1F2F3F4F5F6F7D4
:08043000F8F9FAFBFCFDFEFFF8
.E:00
```



INPUT

00	400	NOP		
01	401	--?		
1011	402	JCN	0	11
2021	404	FIM	0P	21
21	406	SRC	0P	
23	407	SRC	1P	
30	408	FIN	0P	
31	409	JIN	0P	
4090	40A	JUN	090	
50F4	40C	JMS	0F4	
60	40E	INC	0	
61	40F	INC	1	
7071	410	ISZ	0	71
80	412	ADD	0	
90	413	SUB	0	
A0	414	LD	0	
B0	415	XCH	0	
C0	416	BBL	0	
D0	417	LDM	0	
E0	418	WRM		
E1	419	WMP		
E2	41A	WRR		
E3	41B	WPM		
E4	41C	WR0		
E5	41D	WR1		
E6	41E	WR2		
E7	41F	WR3		
E8	420	SBM		
E9	421	RDM		
EA	422	RDR		
EB	423	ADM		
EC	424	RD0		
ED	425	RD1		
EE	426	RD2		
EF	427	RD3		
F0	428	CLB		
F1	429	CLC		
F2	42A	IAC		
F3	42B	CMC		
F4	42C	CMA		
F5	42D	RAL		
F6	42E	RAR		
F7	42F	TCC		
F8	430	DAC		
F9	431	TCS		
FA	432	STC		
FB	433	DAA		
FC	434	KBP		
FD	435	DCL		
FE	436	--?		
FF	437	--?		

OUTPUT

Comment to TRANSLATE HEX

Object: Sometimes the program shall be changed directly in the program memory. Then the initial source program disagrees with the content of memory. This program can limited recover the source program from a paper tape in W-file.

Limitation: Jump address, labels, data etc. are reproduced in HEX code. Available data standing in the place of instructions in the program are not recognizable. Therefore, they will be interpreted like instructions.

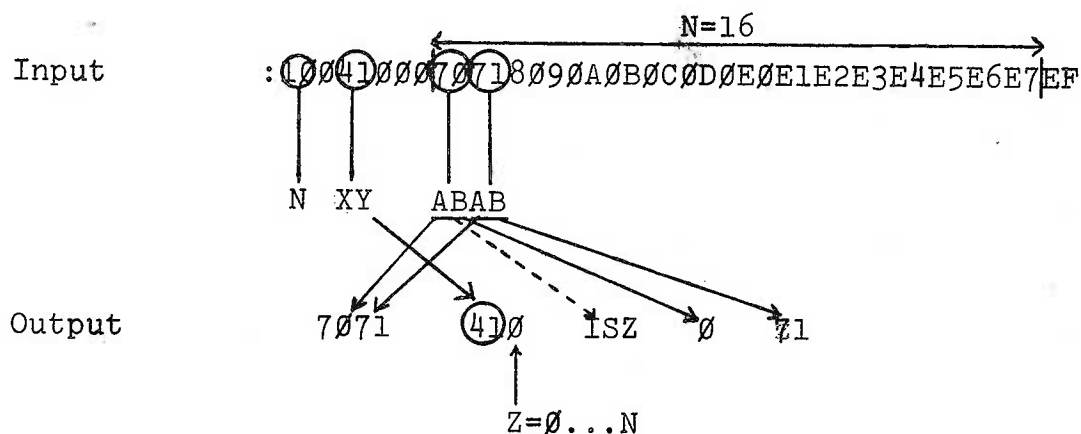
Condition: The first instruction of TRANSLATE HEX has to be assembled at the number 0 respectively $n \times 256$.

The symbols used in the comment have the following meaning:

N: number of instructions that are contained in the following file

XYZ: address of an instruction

AB: instruction or half-instruction in HEX code




```

0:/      T R A N S L A T E   H E X
0:/
0:/
0:START,FIM 5P 0      /N=0,Z=0
2:OK,   FIM 1P 13
4:      JMS PRL      /RETURN
6:      FIM 1P 10
8:      JMS PRL      /LINE FEED
10:     JMS RNW      /LOAD A,B (IF NEW FILE LOAD X,Y )
12:     JMS WCO      /WRITE A,B
14:     JMS WSP      /SPACE
16:     JMS WSP
18:     LD 14
19:     XCH 3
20:     JMS PRD      /WRITE X
22:     LD 15
23:     XCH 3
24:     JMS PRD      /WRITE Y
26:     LD 10
27:     XCH 3
28:     JMS PRD      /WRITE Z
30:     INC 10      /Z=Z+1
31:     JMS WSP      /SPACE
33:     LD 12
34:     JCN 12 VAL   /JUMP IF A#0
36:     LD 13
37:     JCN 4 VAL    /JUMP IF B=0
39:     LDM 15
40:     XCH 13
41:     JUN ACI      /A=0,B#0(INVALID INSTRUCTION),DO LIKE "FF"
43:VAL,  IAC        /VALID INSTRUCTION
44:     JCN 4 ACI    /A=15 ACCUMULATOR INSTRUCTION
46:     IAC
47:     JCN 4 IOI    /A=14 INPUT-OUTPUT-INSTRUCTION
49:WUM,  LDM 2      /WORK UP MACHINE INSTRUCTIONS
50:     CLC        /(BECAUSE FIM/SRC AND FIN/JIN HAS SAME "A")
51:     SUB 12
52:     JCN 12 WUM1
54:     LD 13
55:     RAR
56:     JCN 2 WUM1
58:     LDM 15
59:     XCH 12
60:WUM1, LDM 3
61:     CLC
62:     SUB 12
63:     JCN 12 WUM2
65:     LD 13
66:     RAR
67:     JCN 2 WUM2
69:     LDM 14
70:     XCH 12
71:WUM2, LD 12      /INTERCHANGE A AND B
72:     XCH 13
73:     XCH 12
74:     LDM 14
75:     XCH 9
76:     JUN WIN

```

```

78:ACI, LDM 10
79: XCH 9
80: JUN WIN
82:IOI, LDM 12
83: XCH 9
84:WIN, LDM 4 /WRITE INSTRUCTION
85: XCH 6 /IR6=4
86: LD 13
87: XCH 1
88: LD 9
89: XCH 0
90: FIN 1P /FETCH CODED DATA
91: LD 2
92: CLC
93: RAR
94: XCH 8 /STORE REST OF CODE
95: LDM 0
96: ADD 6
97: XCH 2
98: JMS PRL /PRINT 1ST LETTER
100: INC 9
101: LD 9
102: XCH 0
103: LD 13
104: XCH 1
105: FIN 1P /FETCH 2ND DATA
106: LD 2
107: XCH 7
108: CLB
109: XCH 8
110: RAR
111: XCH 8
112: LDM 0
113: ADD 6
114: XCH 2
115: JMS PRL /PRINT 2ND LETTER
117: CLC
118: LDM 2
119: ADD 8
120: XCH 2
121: LD 7
122: XCH 3
123: JMS PRL /PRINT 3RD LETTER
125: INC 9
126: LD 9
127: JCN 12 OK /JUMP IF IT WAS NO ACCUMULATOR INSTRUCTION
129: JMS WSP /SPACE
131: LD 13
132: JCN 4 OK /JUMP IF IT WAS A NOP
134: JUN P7
136:CON, LD 12 /SBR"CONVERSE"
137: CLC
138: RAR
139: XCH 3
140: JMS PRD
142: FIM 1P 80
144: JMS PRL
146: BBL 0
147:WCO, LD 12 /SBR "WRITE CODE"
148: XCH 3
149: JMS PRD
151: LD 13
152: XCH 3
153: JMS PRD
155: BBL 0
156:SR0, FIM 0 0
158: SRC 0
159: BBL 0

```

/CODED DATA

160: 131
161: 131
162: 137
163: 131
164: 131
165: 146
166: 210
167: 148
168: 132
169: 212
170: 179
171: 132
172: 203
173: 132
174: 127
175: 127
176: 44
177: 60
178: 49
179: 61
180: 29
181: 193
182: 33
183: 51
184: 49
185: 51
186: 52
187: 17
188: 2
189: 195
190: 255
191: 255
192: 183
193: 215
194: 247
195: 183
196: 119
197: 119
198: 119
199: 119
200: 147
201: 146
202: 210
203: 129
204: 82
205: 82
206: 82
207: 82
208: 210
209: 13
210: 34
211: 208
212: 2
213: 18
214: 34
215: 50
216: 210
217: 212
218: 36
219: 212
220: 4
221: 20
222: 36
223: 52
224: 206

17
248
256
502

225: 138
 226: 179
 227: 138
 228: 170
 229: 202
 230: 137
 231: 233
 232: 129
 233: 179
 234: 12
 235: 152
 236: 130
 237: 140
 238: 134
 239: 134
 240: 15
 241: 227
 242: 50
 243: 233
 244: 229
 245: 61
 246: 62
 247: 163
 248: 68
 249: 37
 250: 4
 251: 131
 252: 194
 253: 212
 254: 233
 255: 217
 256: P7, IAC
 257: JCN 4 MI5
 259: IAC
 260: JCN 4 MI2
 262: LD 13
 263: DAC
 264: JCN 4 MI3
 266: DAC
 267: JCN 4 MI2
 269: DAC
 270: JCN 4 MI2
 272: DAC
 273: JCN 4 MI4
 275: DAC
 276: JCN 4 MI4
 278: DAC
 279: DAC
 280: JCN 4 MI3
 282: MI1, JMS WCH /INC,ADD,SUB,LD,XCH,BBL,LDM
 284: JUN OK
 286: MI2, JMS CON /SRC,JIN,FIN
 288: JUN OK
 290: MI3, JMS WCH /JCN,ISZ
 292: JMS WSP
 294: JUN TWI
 296: MI4, JMS WCH /JUN,JMS
 298: JUN TWI
 300: MI5, JMS CON /FIM
 302: JMS WSP
 304: TWI, JMS RNW /TWO WORD INSTRUCTION
 306: LD 12
 307: XCH 3
 308: JMS PRD /PRINT NEW A
 310: LD 13
 311: XCH 3
 312: JMS PRD /PRINT NEW B

256
 251
 507
 148

314: FIM 1P 13
 316: JMS PRL
 318: JMS WSP
 320: JMS WCO
 322: INC 10
 323: JUN OK
 325:PRD, CLC
 326: LDM 6
 327: ADD 3
 328: JCN 2 PRI
 330: LDM 3
 331: XCH 2
 332: JUN ~~PRI~~
 334:PRI, IAC
 335: XCH 3
 336: LDM 4
 337: XCH 2
 338:PRL, JMS SR0
 340: WMP
 341: LDM 8
 342: XCH 4
 343:TO1, JMS SBR2
 345: CLC
 346: XCH 2
 347: RAR
 348: XCH 2
 349: XCH 3
 350: RAR
 351: XCH 3
 352: TCC
 353: WMP
 354: ISZ 4 TO1
 356: JMS SBR2
 358: LDM 1
 359: WMP
 360: JMS SBR2
 362: JMS SBR2
 364: BBL 0
 365: NOP
 366: NOP
 367: NOP
 368:RCH, CLC
 369: LDM 1
 370: FIM 6 64
 372: SRC 6
 373: WMP
 374: LDM 8
 375: XCH 4
 376: JMS SR0
 378:TI0, RDR
 379: RAR
 380: JNC TI0
 382: JMS SBR1
 384: SRC 6
 385: WMP
 386: SRC 0
 387: WMP
 388:TI1, JMS SBR2

/RETURN
 /SPACE
 /PRINT NEW A AND B AGAIN
 /Z=Z+1

 /PRINT A DIGIT (TTY-OUTPUT)

/PRINT A LETTER (TTY-OUTPUT)

/READ CHARACTER FROM TAPE READER

256
 + 128
 384

157
 852
 287
 343

//

390:	RDR	
391:	CMA	
392:	RAR	
393:	LD 2	
394:	RAR	
395:	XCH 2	
396:	LD 3	
397:	RAR	
398:	XCH 3	
399:	ISZ 4 TI1	
401:	JMS SBR2	
403:	LDM 1	
404:	WMP	
405:	JMS SBR2	
407:	JMS SBR1	
409:	XCH 2	
410:	RAL	
411:	CLC	
412:	RAR	
413:	XCH 2	
414:	CLC	
415:	LDM 13	
416:	ADD 2	
417:	CLC	
418:	JCN 4 RCH1	
420:	LDM 9	
421:	ADD 3	
422:	XCH 3	
423:	RCH2, BBL 0	/NO:"WAS READ
424:	RCH1, LDM 6	
425:	ADD 3	
426:	JCN 12 RCH2	
428:	BBL 1	/"WAS READ
429:	SBR2, FIM 0 60	/9.09 MS DELAY
431:	L2, ISZ 1 L2	
433:	ISZ 0 L2	
435:	SBR1, FIM 0 60	/4.55 MS DELAY
437:	L1, ISZ 1 L1	
439:	ISZ 0 L1	
441:	BBL 0	
442:	RNW, LD 10	/READ NEXT WORD
443:	CLC	
444:	SUB 11	
445:	JCN 12 RN3	/JUMP IF Z#N
447:	RN1, JMS RCH	/WAIT FOR ":"
449:	JCN 4 RN1	
451:	JMS RCH	
453:	LD 3	

```

454:      JCN 4 RN4
456:      CLB
457:      XCH 11      /STORE N (0=16)
458:      JMS RCH
460:      JUN RN2
462:RN4, JMS RCH
464:      LD 3
465:END, JCN 4 LOOP  /IF ":00" WAS READ
467:      XCH 11
468:RN2, JMS RCH
470:      JMS RCH
472:      LD 3
473:      XCH 14      /STORE X
474:      JMS RCH
476:      LD 3
477:      XCH 15      /STORE Y
478:      JMS RCH
480:      JMS RCH
482:      JMS RCH
484:      XCH 10      /Z=0
485:RN3, JMS RCH
487:      LD 3
488:      XCH 12      /STORE A
489:      JMS RCH
491:      LD 3
492:      XCH 13      /STORE B
493:      BBL 0
494:LOOP,JUN LOOP
496:WSP, LDM 13      /WRITE SPACE (3 BLANCS)
497:      XCH 9
498:WSP1,FIM 1P 32
500:      JMS PRL
502:      ISZ 9 WSP1
504:      BBL 0
505:WCH, LD 12      /WRITE CHARACTER (B)
506:      XCH 3
507:      JMS PRD
509:      BBL 0
510:

```

159
 C8
 C3
 9
 9



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-15☒ 4004 ☒ 4040 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

**Program
Title**

4Digit BCD to Binary Converter

Function

This subroutine converts a 4 digit binary coded decimal number into its straight binary equivalent. The 4 digit BCD numbers are placed in 4 registers by the calling routine, and the binary equivalents are returned in the same 4 registers by this subroutine. In addition, this subroutine needs 2 more registers for work space. The registers are coded symbolically, thus the actual registers used can be easily assigned at assemble time. Accumulator and carry contents are ignored at entry and zeroed at return. This subroutine does not call any other subroutines (internal or external). It needs 69 words of ROM for program storage and no RAM or I/O ports.

**Required
Hardware****Required
Software**

The execution time of this subroutine is always 69 cycles, independent of the number that is being converted. This makes it possible to incorporate it in an I/O timing loop and utilize the CPU while it is waiting for I/O.

**Input
Parameters**

The conversion method used in this subroutine is described in the source code in detail. Basically, the conversion is decomposed into additions and multiplications by 16, 8, or 2. The decomposition is optimized in such a way that a minimum number of steps of additions and multiplications are needed and that at each step a minimum number of bits are needed. Multiply by 16 is done by moving the lower significant words (4 bit each) to higher significant words. Multiply by 8 is done by rotate-right and then moving to higher significant words. Multiply by 2 is done by rotate-left.

**Output
Results**

A test program is included. This test program uses a 4-register BCD counter and a 4-register binary counter. Both counters are initialized to zeroes and incremented by decimal add (using DAA instruction) and binary add. In each step, the BCD counter is copied to another set of 4 registers and the converter subroutine is called. The result of the conversion is then compared with the binary counter. The test is finished when the BCD counter reaches 9999. This takes about 14 seconds.

Registers Modified: Six registers (any six)	Assembler/Compiler Used: PDP-9 Cross Assembler
RAM Required: None	Programmer: Lichen Wang
ROM Required: 69 words	Company: SLAC, Stanford University
Maximum Subroutine Nesting Level None	Address: Box 4349, Stanford, Ca. 94305

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps is not necessary.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

Your program will be photo-copied for publication in the User's Library. Please send an original, clear, un-marked copy.

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

```

/      * * * * *
/      * 4 DIGIT BCD TO BINARY CONVERTER *
/      *      FOR INTEL 4004 OR 4040      *
/      * * * * *
/
/      * REGISTER ASSIGNMENT *
/
/ REG.:      INPUT BCD:      OUTPUT BINARY:
000 RA=0      / THOUSANDS DIGIT A, MOST SIGNIFICANT
001 RB=1      / HUNDREDS DIGIT B, SECOND ...
002 RC=2      / TENS DIGIT C, THIRD ...
003 RD=3      / UNIT DIGIT D, LEAST ...
004 RE=4      / - - - WORK SPACE - - -
005 RF=5      / - - - WORK SPACE - - -
/
/ THIS SUBROUTINE NEEDS FOUR REGISTERS TO PASS
/ INPUT OUTPUT PARAMETERS AND TWO ADDITIONAL
/ REGISTERS FOR WORK SPACE. THE REGISTERS USED
/ ARE CODED SYMBOLICALLY. THUS AT ASSEMBLE TIME
/ THE USER CAN ASSIGN ANY SIX DISTINCT REGISTERS
/ THAT HE WISHES TO USE BY CHANGING THE ABOVE
/ ASSIGNMENT TABLE.
/ THE CARRY (CY) AND THE ACCUMULATOR (AC) ARE BOTH
/ ZEROED AT RETURN.
/ THIS SUBROUTINE DOES NOT CALL OTHER SUBROUTINES,
/ THUS IT USES ONLY ONE LEVEL OF THE STACK.
/
/      * HARDWARE REQUIREMENT *
/
/ THIS SUBROUTINE NEEDS 69 WORDS OF ROM PROGRAM
/ STORAGE. IT DOES NOT NEED ANY RAM OR I/O PORT.
/
/      * EXECUTION TIME *
/
/ THE EXECUTION TIME IS INDEPENDENT OF DATA AND IS
/ ALWAYS 69 CYCLES (745.2 USEC). THIS MAKES IT
/ POSSIBLE TO BE INCORPORATED INSIDE AN I/O TIMING
/ LOOP AND UTILIZE THE CPU WHILE IT IS WAITING I/O.
/
/      * METHOD USED *
/
/ TO CALCULATE:
/  $1000*A + 100*B + 10*C + D = 50*2*(10*A+B) + (10*C+D)$ 
/ WE DEFINE:
/  $U = 2*(10*A+B) = 2*(8*A+B+2*A)$ 
/  $V = (10*C+D) = 8*C+D+2*C$ 
/ THEN:
/  $1000*A + 100*B + 10*C + D = 50*U + V = (16*U+V) + 2*(16*U+U)$ 
/
/ NOTE THAT U AND V NEED TWO 4-BIT WORDS EACH
/ WHEREAS (16*U+V) AND (16*U+U) NEED THREE 4-BIT
/ WORDS EACH AND THE FINAL ANSWER NEEDS FOUR
/ 4-BIT WORDS. IN THE COMMENT FIELD, WE USE THE
/ CHARACTER ":" BETWEEN REGISTER NAMES TO INDICATE
/ MULTI-WORD.

```

```

/
/ FIRST WE DO 8*A+B, WE NEED 2 WORDS, HIGH AND LOW
DT0B, CLC /CY=0
00000 361 LD RA /CY=0, AC=A
00001 240 RAR /AC:CY=8*A
00002 366 XCH RE /CY=8*A LOW, RE=8*A HIGH
00003 264 LDM 0 /CY=8*A LOW, AC=0
00004 320 RAR /CY=0, AC=8*A LOW
00005 366 ADD RB /CY:AC=8*A+B LOW
00006 201 XCH RB /CY=8*A+B LOW OVF, RB=8*A+B LOW
00007 261 TCC /CY=0, AC=8*A+B LOW OVF
00010 367 ADD RE /CY=0, AC=8*A+B HIGH
00011 204 XCH RA /CY=0, AC=A, RA:RB=8*A+B
00012 260

/
/ NOW ADD 2*A TO 8*A+B, STILL 2 WORDS LONG
00013 365 RAL /CY:AC=2*A
00014 265 XCH RF /CY=2*A HIGH, RF=2*A LOW
00015 367 TCC /CY=0, AC=2*A HIGH
00016 265 XCH RF /CY=0, AC=2*A LOW, RF=2*A HIGH
00017 201 ADD RB /CY:AC=10*A+B LOW
00020 265 XCH RF /CY=10*A+B LOW OVF, AC=2*A HIGH
00021 200 ADD RA /CY=0, AC:RF=10*A+B

/
/ NEXT FIND U=2*(10A+B), STILL 2 WORDS LONG
00022 265 XCH RF /AC=10*A+B LOW, RF=10*A+B HIGH
00023 365 RAL /CY:AC=U LOW
00024 265 XCH RF /AC=10*A+B HIGH, RF=U LOW
00025 365 RAL /CY=0, AC=U HIGH
00026 264 XCH RE /CY=0, RE:RF=U

/
/ V=10*C+D=8*C+D+2*C IS CALCULATED THE SAME WAY
00027 242 LD RC /CY=0, AC=C
00030 366 RAR /AC:CY=8*C
00031 261 XCH RB /CY=8*C LOW, RB=8*C HIGH
00032 320 LDM 0 /CY=8*C LOW, AC=0
00033 366 RAR /CY=0, AC=8*C LOW
00034 203 ADD RD /CY:AC=8*C+D LOW
00035 263 XCH RD /CY=8*C+D LOW OVF, RD=8*C+D LOW
00036 367 TCC /CY=0, AC=8*C+D LOW OVF
00037 201 ADD RB /CY=0, AC=8*C+D HIGH
00040 262 XCH RC /CY=0, AC=C, RC:RD=8*C+D
00041 365 RAL /CY:AC=2*C
00042 260 XCH RA /CY=2*C HIGH, RA=2*C LOW
00043 367 TCC /CY=0, AC=2*C HIGH
00044 260 XCH RA /CY=0, AC=2*C LOW, RA=2*C HIGH
00045 203 ADD RD /CY:AC=10*C+D LOW
00046 260 XCH RA /CY=10*C+D LOW OVF, AC=2*C HIGH
00047 202 ADD RC /CY=0, AC:RA=10*C+D=V

/
/ NOW DO 16*U+V, THIS NEEDS 3 WORDS, HIGH MED LOW
00050 205 ADD RF /CY:AC=16*U+V MED
00051 263 XCH RD /RD=16*U+V MED, RA=V=16*U+V LOW
00052 367 TCC /CY=0, AC=16*U+V MED OVF
00053 204 ADD RE /CY=0, AC=16*U+V HIGH
00054 262 XCH RC /CY=0, RC:RD:RA=16*U+V

```

```
/
/ SAME WAY WE CAN DO  $17*U=16*U+U$ 
00055 245 LD RF /CY=0, AC=U LOW
00056 204 ADD RE /CY:AC=17*U MED
00057 261 XCH RB /RB=17*U MED
00060 367 TCC /CY=0, AC=17*U MED OVF
00061 204 ADD RE /CY=0, AC:RB:RF=17*U
/
/ MAKE  $34*U=2*(17*U)$  BY ROTATE LEFT
00062 265 XCH RF /AC=17*U LOW, RF=17*U HIGH
00063 365 RAL /CY:AC=34*U LOW
00064 261 XCH RB /AC=17*U MED, RB=34*U LOW
00065 365 RAL /CY:AC=34*U MED
00066 265 XCH RF /AC=17*U HIGH, RF=34*U MED
00067 365 RAL /CY:AC=34*U HIGH
00070 264 XCH RE /CY=34*U HIGH OVF, RE=34*U HIGH
00071 367 TCC /CY=0, AC=34*U HIGH OVF
00072 260 XCH RA /AC=16*U+V LOW, RA:RE:RF:RB=34*U
/
/ FINALLY, WE ADD  $16*U+V$  AND  $34*U$  TO GET THE ANSWER
00073 201 ADD RB /CY:AC=50*U+B LOW
00074 263 XCH RD
00075 205 ADD RF /CY:AC=50*U+V MED
00076 262 XCH RC /AC=16*U+V HIGH, RC=50*U+V MED
00077 204 ADD RE /CY:AC=50*U+V HIGH
00100 261 XCH RB /CY=HIGH OVF, RB=50*U+V HIGH
00101 367 TCC /CY=0, AC=HIGH OVF
00102 200 ADD RA /AC=50*U+V OVF
00103 260 XCH RA /RA=50*U+V OVF
00104 300 BBL 0 /CY=0, AC=0, RA:RB:RC:RD=ANSWER
000 .END
NO ERROR LINES
```

PIP V7A

>

```

/      * * * * *
/      *                TEST                *
/      * 4 DIGIT BCD TO BINARY CONVERTER    *
/      *      FOR INTEL 4004 OR 4040        *
/      * * * * *
/
/      * REGISTER ASSIGNMENT *
/
/ REG.:      INPUT BCD:      OUTPUT BINARY:
000 RA=0      / THOUSANDS DIGIT A, MOST SIGNIFICANT
001 RB=1      / HUNDREDS DIGIT B, SECOND ...
002 RC=2      / TENS DIGIT C, THIRD ...
003 RD=3      / UNIT DIGIT D, LEAST ...
004 RE=4      / - - - WORK SPACE - - -
005 RF=5      / - - - WORK SPACE - - -
/
006 RG=6      / THOUSANDS DIGIT FOR BCD COUNTER
007 RH=7      / HUNDREDS
010 RI=8      / TENS
011 RJ=9      / UNIT
012 RK=10     / MOST SIGNIFICAND PART OF BINARY COUNTER
013 RL=11     / SECOND
014 RM=12     / THIRD
015 RN=13     / LEAST
/
/ THIS PROGRAM TESTS THE 4 DIGIT BCD TO BINARY
/ CONVERTER. IT TAKES ABOUT 14 SECONDS TO TEST
/ ALL THE CONVERSIONS FROM 0000 TO 9999.
/ FOR 4040 THE PROGRAM WILL HALT AT ERROR. FOR
/ 4004 THE HALT INSTRUCTION SHOULD BE CHANGED TO
/ A JUN; * INSTRUCTION.
/
/ INITIALIZE BCD AND BINARY COUNTERS
00000 046 BGN, FIM RG; 0
00001 000
00002 050 FIM RI; 0
00003 000
00004 052 FIM RK; 0
00005 000
00006 054 FIM RM; 0
00007 000
/ COPY BCD COUNTER INTO INPUT
00010 246 COP, LD RG
00011 260 XCH RA
00012 247 LD RH
00013 261 XCH RB
00014 250 LD RI
00015 262 XCH RC
00016 251 LD RJ
00017 263 XCH RD
/ CALL SUBROUTINE DTOB ON PAGE 1
00020 121 JMS I; DTOB
00021 000
/ COMPARE OUTPUT WITH BINARY COUNTER
00022 361 CLC

```

TEST.DTOB PAGE 2

```

00023 240      LD RA
00024 232      SUB RK
00025 030      JCN AZ; CK1
00026 030
00027 001      HLT          /OR JUN; * FOR 4004
00030 361      CK1, CLC
00031 241      LD RB
00032 233      SUB RL
00033 030      JCN AZ; CK2
00034 036
00035 001      HLT          /OR JUN; * FOR 4004
00036 361      CK2, CLC
00037 242      LD RC
00040 234      SUB RM
00041 030      JCN AZ; CK3
00042 044
00043 001      HLT          /OR JUN; * FOR 4004
00044 361      CK3, CLC
00045 243      LD RD
00046 235      SUB RN
00047 030      JCN AZ; CK4
00050 052
00051 001      HLT          /OR JUN; * FOR 4004
/ IF ALL COMPARES, INCREMENT BINARY COUNTER
00052 372      CK4, STC
00053 367      TCC
00054 215      ADD RN
00055 275      XCH RN
00056 367      TCC
00057 214      ADD RM
00060 274      XCH RM
00061 367      TCC
00062 213      ADD RL
00063 273      XCH RL
00064 367      TCC
00065 212      ADD RK
00066 272      XCH RK
/ INCREMENT BCD COUNTER
00067 372      STC
00070 367      TCC
00071 211      ADD RJ
00072 373      DAA
00073 271      XCH RJ
00074 367      TCC
00075 210      ADD RI
00076 373      DAA
00077 270      XCH RI
00100 367      TCC
00101 207      ADD RH
00102 373      DAA
00103 267      XCH RH
00104 367      TCC
00105 206      ADD RG
00106 373      DAA
00107 266      XCH RG

```

TEST.DTOB PAGE 3

```
      / IF BCD=0000 TO 9999, GO BACK TO COPY AND TEST
      / IF BCD=10000 (OVERFLOW), GO TO BEGINNING
00110 032      JCN CZ; COP
00111 010
00112 100      JUN 0; BGN
00113 000
      .END
      NO ERROR LINES
```

PIP V7A

>



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-16☒ 4004 ☐ 8008 ☐ 8080 ☐ 4040

(use additional sheets if necessary)

Program Title	MOBLE MEAN PROGRAM
Function	CALCULATES THE MEAN BETWEEN THE CURRENT VALUE OF A 3 DECADES BCD NUMBER AND THE PREVIOUS 4 VALUES
Required Hardware	MCS4 OR MCS40 SYSTEM
Required Software	MAC40 ASSEMBLER
Input Parameters	THE NEW VALUE IS ASSUMED TO BE IN THE 3 INDEX REGISTERS VAL1(UNITS), VAL2 (TENTH), VAL3 (HUND.)
Output Results	THE MEAN IS RETURNED IN THE INDEX REGISTERS VAL1 VAL2 VAL3

Registers Modified: n ⁰ 3 + n ⁰ 4 FOR THE MEAN ALLOC.	Maximum Subroutine Nesting Level: 0
RAM Required 1 REGISTER (16 CHARACTERS)	Assembler/Compiler Used MAC40
ROM Required: 68	Programmer: E. MASSETTI
	Company: Lab. Massetti Milano ITALY

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer and company
2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps should **not** be included.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

```

;
; MOBLE MEAN PROGRAM
;
;
; INITIALIZE THE PROGRAM PARAMETERS
;
0000 BNKMN EQU 0 ;MOBLE MEAN REGISTER ALLOCATION
0000 POINT EQU 0 ;POINTER INDEX REGISTER ALLOCATION
0002 VAL1 EQU 2 ;1 DECADE INDEX REGISTER ALLOCATION
0003 VAL2 EQU 3 ;2 DECADE INDEX REGISTER ALLOCATION
0004 VAL3 EQU 4 ;3 DECADE INDEX REGISTER ALLOCATION
0005 VAL4 EQU 5 ;4 DECADE INDEX REGISTER ALLOCATION
0006 COUNT EQU 6 ;LOOP COUNTER INDEX REGISTER ALLOCATION
; THE NEW VALUE IS ASSUMED TO BE IN THE REGISTER VAL1 TO VAL3
; WITH 3 BCD DECADES PRECISION
;
; WRITE A NEW VALUE IN THE MOBLE MEAN REGISTER
;
0000 MMEAN:
0000 D0 LDM BNKMN ;SELECT THE MOBLE MEAN REGISTER
0001 FD DCL
0002 21 SRC POINT
0003 A2 LD VAL1 ;LOAD THE LEAST SIGNIFICANT DECADE OF THE NEW
0004 E0 WRM VALUE
0005 61 INC POINT+1
0006 21 SRC POINT
0007 A3 LD VAL2
0008 E0 WRM
0009 61 INC POINT+1
000A 21 SRC POINT
000B A4 LD VAL3
000C E0 WRM
000D 7110 ISZ POINT+1,NZERO ;TEST IF POINTER=0
000F 61 INC POINT+1
;
; CALCULATE THE SUM OF ALL THE 5 VALUES OF THE MEAN
;
0010 NZERO:
0010 260C FIM COUNT,12 ;INIZIALIZA LOOP COUNTER
0012 F0 CLB
0013 B5 XCH VAL4 ;INIZIALIZA THE MOST SIGNIFICANT DECADE
0014 21 LOOP: SRC POINT
0015 A2 LD VAL1
0016 EB ADM ;SUM OF VALUES
0017 FB DAA
0018 B2 XCH VAL1
0019 61 INC POINT+1
001A 21 SRC POINT
001B A3 LD VAL2
001C EB ADM

```

```

001D FB DAA
001E B3 XCH VAL2
001F 61 INC POINT+1
0020 21 SRC POINT
0021 A4 LD VAL3
0022 EB ADM
0023 FB DAA
0024 B4 XCH VAL3
0025 7128 ISZ POINT+1,NOTZE ;TEST POINTER FOR ZERO
0027 61 INC POINT+1 ;POINTER=15
0028 1A2B NOTZE:JNC TEST ; CARRY THROUGH DECADES
002A 65 INC VAL4
002B 7614 TEST: ISZ COUNT,LOOP ;TEST FOR END OF VALUES
002D 61 INC POINT+1 ;REST POINTER
002E 61 INC POINT+1 ;TO THE NEXT ENTRY
002F 7132 ISZ POINT+1,FOLLO
0031 61 INC POINT+1
;
; DIVIDE THE SUM BY 5
;
FOLLO: CLC
0032 F1 LD VAL1 ;VAL1 TO VAL4 MULTIPLIED BY 2 AND DIVIDED BY
0033 A2 ADD VAL1 ;VAL1*2
0034 82 DAA
0035 FB LD VAL2
0036 A3 ADD VAL2 ;VAL2*2
0037 83 DAA
0038 FB XCH VAL1 ;/10
0039 B2 LD VAL3
003A A4 ADD VAL3 ;VAL3*2
003B 84 DAA
003C FB XCH VAL2 ;/10
003D B3 LD VAL4
003E A5 ADD VAL4 ;VAL4*2
003F 85 DAA
0040 FB XCH VAL3 ;/10
0041 B4 CLB
0042 F0 XCH VAL4 ;VAL4=0
0043 B5
;
; THE MOBLE MEAN IS NOW IN THE INDEX REGISTERS VAL1 TO VAL4
;
END

```

NO PROGRAM ERRORS

4004 MACRO ASSEMBLER, VER 2.2 ERRORS = 0 PAGE 3

SYMBOL TABLE

♦ 01

BNKMN	0000	COUNT	0006	FOLLO	0032	LOOP	0014
MMEAN	0000 ♦	NOTZE	0028	NZERO	0010	POINT	0000
TEST	002B	VAL1	0002	VAL2	0003	VAL3	0004
VAL4	0005						

READY



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 40-11☐ 4004 ☐ 8008 ☐ 8080 ☒ 4040

(use additional sheets if necessary)

Program Title	HEXBCD
Function	Convert 2 digit HEX Value to decimal range - FF
Required Hardware	None
Required Software	Self supporting
Input Parameters	Any value 00 - FF located in register pair 7 (IR 14 = MSD, IR 15 = LSD) IR pair 0 & 1 used for routine
Output Results	Value 0 -255 Decimal MSD = located in IR 0 MIDD = located in IR1 LSD = located in IR2

Registers Modified: 0, 1, 2, 3	Maximum Subroutine Nesting Level: One + initial call
RAM Required: None	Assembler/Compiler Used: Intellec 4/40 Ver 3.0
ROM Required: 23H	Programmer: Charles M. Miller
	Company: Audio Services, Inc.

3140 East Jefferson Ave.
Detroit Michigan 48207

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only):
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer and company
2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps should **not** be included.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

0000	2000	HEXBCD:	FIM 0,0	;CLEAR WORK AREA
0002	2200		FIM 2,0	; "
0004	F1		CLC	
0005	AF		LD 15	;READ LSD HEX DIGIT
0006	FB		DAA	
0007	B2		XCH 2	;STORE LSD DECIMAL VALUE
0008	81		ADD 1	;CARRY TO MID. DIGIT
0009	B1		XCH 1	;STORE MID. DIGIT
000A	AE		LD 14	;READ MSD HEX DIGIT
000B	F4		CMA	
000C	E3		XCH 3	;STORE AND USE AS COUNTER
000D	7310	AAA:	ISZ 3,BBB	;IF ZERO THEN BBL
000F	C0		BBL 0	
0010	D8	BBB:	LDM 8	;PREPARE TO ADD 8
0011	5018		JMS ADER	
0013	D8		LDM 8	;PREPARE TO ADD 8 AGAIN
0014	5018		JMS ADER	
0016	400D		JUN AAA	
0018	F1	ADER:	CLC	;THIS IS A THREE DIGIT ADD ROUTINE
0019	82		ADD 2	
001A	FB		DAA	
001B	B2		XCH 2	
001C	D0		LDM 0	
001D	81		ADD 1	
001E	FB		DAA	
001F	B1		XCH 1	
0020	D0		LDM 0	
0021	80		ADD 0	
0022	B0		XCH 0	
0023	C0		BBL 0	
			END	

;HEX TO BCD TEST PROGRAM.
 ;WITH THE "S"COMMAND FROM THE MONITOR MODE
 ;THE FIM INSTRUCTION (LOCATION 0-1) CAN BE MODIFIED
 ;TO LOAD ANY TWO DIGIT HEX NUMBER INTO IR PAIR 7
 ;THEN WITH THE PROGRAM RUNNING IN RAM,THE
 ;RESULTS CAN BE SEEN DURING "X3" WHEN THE
 ;XCH2,1,0 INSTRUCTIONS AT THE BEGINNING OF
 ;THE PROGRAM ARE EXECUTED.

```

0000 2E64 TEST: FIM 14,64H ;VALUE TO CONVERT
0002 500C JMS HEXBCD
0004 A2 LOOP: LD 2
0005 B2 XCH 2 ;LSD DECIMAL VALUE
0006 A1 LD 1
0007 B1 XCH 1 ;MIDD DECIMAL VALUE
0008 A0 LD 0
0009 B0 XCH 0 ;MSD DECIMAL VALUE
000A 4004 JUN LOOP
000C 2000 HEXBCD: FIM 0,0 ;CLEAR WORK AREA
000E 2200 FIM 2,0 ;
0010 F1 CLC
0011 AF LD 15 ;READ LSD HEX DIGIT
0012 FB DAA
0013 B2 XCH 2 ;STORE LSD DECIMAL VALUE
0014 81 ADD 1 ;CARRY TO MID. DIGIT
0015 B1 XCH 1 ;STORE MID. DIGIT
0016 AE LD 14 ;READ MSD HEX DIGIT
0017 F4 CMA
0018 B3 XCH 3 ;STORE AND USE AS COUNTER
0019 731C AAA: ISZ 3,BBB ;IF ZERO THEN BBL
001B C0 BBL 0
001C D8 BBB: LDM 8 ;PREPARE TO ADD 8
001D 5024 JMS ADER
001F D8 LDM 8 ;PREPARE TO ADD 8 AGAIN
0020 5024 JMS ADER
0022 4019 JUN AAA
0024 F1 ADER: CLC ;THIS IS A THREE DIGIT ADD ROUTINE
0025 82 ADD 2
0026 FB DAA
0027 B2 XCH 2
0028 D0 LDM 0
0029 81 ADD 1
002A FB DAA
002B B1 XCH 1
002C D0 LDM 0
002D 80 ADD 0
002E B0 XCH 0
002F C0 BBL 0
END

```




MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 40-12☒ 4004 ☒ 4040 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title	FAST BINARY MULTIPLY: SELECTABLE BIT-PRECISION & CONSTANT EXECUTE TIME.
Function	See the attached functional description & CPU register map. In general, the user loads the input variables to CPU registers and specifies one of five (5) multiply precisions (12x12, 12x8, 8x8, 4x8, 4x4) via a code character in register E.
Required Hardware	Only for the User Test Routine. See the attached User Test Routine discussion and schematic.
Required Software	The User Test Routine requires the INTELLEC MONITOR "S" command to specify bit-precision and input variables.
Input Parameters	See the attached Functional description.
Output Results	See the attached Functional description. The Multiply subroutine returns with the Product in registers 8 thru D, depending on selected bit-precision. The User Test Routine displays the contents of all CPU product registers on several hexadecimal, LED lamp-banks. A timing pulse is generated, bracketing the MULTIPLY subroutine, thus allowing a measurement of execution time.

Registers Modified: From 7 to 15 CPU regs depending on selected bit-precision	Assembler/Compiler Used: INTELLEC 4 ASSEMBLER:VERSION 3.0
RAM Required: NONE	Programmer: Richard J. Fabbri
ROM Required: MULTIPLY = 136 Bytes 4*4 LOOK-UP TABLE = 256 Bytes USER TEST ROUTINE = 100 Bytes	Company: CBS Laboratories
Maximum Subroutine Nesting Level: Two	Address: 227 High Ridge Road Stamford, Conn. 06905

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
 2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
 3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.
-

Your program will be photo-copied for publication in the User's Library. Please send an original, clear, un-marked copy.

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

FUNCTION:

A "Look-Up" table(storing all possible 4*4Bit products) technique is used to multiply five, user-selectable Bit-precisions,namely:

$$12*12,12*8,8*8,4*8,4*4.$$

These particular precisions are suited to the variety of calculations encountered with 4 to 12Bit A/D data. It is interesting to note that useful signal processing factors such as $2/\pi$ and $1/\sqrt{2}$ are approximated to 12Bit, 1/2 LSB accuracy by 8Bit constants:

$$2/\pi = 0.A3H \quad 1/\sqrt{2} = 0.B5H.$$

Furthermore, this multiply subroutine features a constant execute time for each selected precision i.e. independent of the variables' particular value. The 4-Bit CPU user can thus readily include multiplication in time-dependent, control applications.

This technique also yields a very fast multiply i.e. 12*12Bit in 2.160 msec. Contrasting, 8*8Bit utilizing the "shift and add" (Booth) algorithm requires 800 to 4500 usec. All five multiply execute times are given in the Source program listing.

The idea behind this subroutine is to treat binary numbers as a sum of weighted hexadecimal digits. So, if "A" and "B" are 12-Bit numbers, then:

$$A*B = (A3A2A1)*(B3B2B1).$$

A 12*12Bit multiplication is then expressed:

$$\begin{array}{r} (A3*2^8) + (A2*2^4) + A1 \\ (B3*2^8) + (B2*2^4) + B1 \\ \hline \begin{array}{r} A3*B1*2^8 \\ A2*B1*2^4 \\ A1*B1 \\ A3*B2*2^8 \\ A2*B2*2^4 \\ A1*B2*2^0 \\ A3*B3*2^{12} \\ A2*B3*2^8 \\ A1*B3*2^4 \end{array} \end{array}$$

The individual 4*4Bit products are "looked-up" via a FIN P3 instruction. By placing the FIN at the last byte of a page, the bytes of the next page can be(and are) accessed (via P0) as the 8-Bit products of the 4-Bit numbers in R0 and R1. Since BBL must occupy the first byte of this look-up table(page), then several instructions must(and do) determine if P0 = 00H previous to addressing the table.

These 8-Bit products are then added to each other with a proper attention paid to their relative binary weighting, completing the total multiply function. A register map of the initialized sub-routine is as follows:

E	CODE	7	—	F
C	C6	6	C5	D
A	C4	5	C3	B
8	C2	4	C1	9
6	Int.	3	Prod	7
4	A3	2	B3	5
2	A2	1	B2	3
0	A1	0	B1	1

CODE:

$F = 4*4 = (A1)*(B1)$
 $E = 4*8 = (A1)*(B2B1)$
 $D = 8*8 = (A2A1)*(B2B1)$
 $C = 12*8 = (A3A2A1)*(B2B1)$
 $B = 12*12 = (A3A2A1)*(B3B2B1)$

There are three things the user must do to initialize this sub-routine. First, the variables(A3A2A1B3B2B1) must be entered, to the desired precision, as per the above register map. Second, enter the Bit-precision code. Third, all expected product(C6C5C4C3C2C1) registers must be cleared to 0H and the contents of P3 saved if otherwise important. The "A*B" subroutine is now called. A return, with the product found in RC,RD,RA,RB,R8,R9, occurs when the selected precision is reached since the routine proceeds logically through increasing precision(4*4 to 12*12) via ISZ interrogation of the Bit precision code register RE.

A table of input variable and product registers, as a function of desired Bit-precision and execute time, is given in the Source program listing.

:1

:2

```

*$C
;MULTIPLY: FAST/MULTIPLE-PRECISIONS/CONST-EXECUTE TIME
;-----
;
;          INPUT VARIABLES'   PRODUCT FOUND
;          REGISTERS:        IN REGISTERS:
;    BIT
;PRECISION E (4 2 0)*(5 3 1)  C D A B 8 9    EXECUTE
;-----
; (4)*(4)   F      (X)*      (X)              X X    194.4US
; (4)*(8)   E      (X)*      (X X)            X X X    442.8US
; (8)*(8)   D      (X X)*     (X X)            X X X X   928.8US
; (12)*(8)  C (X X X)*     (X X)            X X X X X  1.415MS
; (12)*(12) B (X X X)*     (X X X)          X X X X X X 2.160MS
;-----
;NOTES:
;    1. PRECISION SELECTED BY REG"E" INITIALIZATION.
;    2. INPUT VARIABLES INITIALIZED IN REGISTERS 0/5.
;    3. EXPECTED PRODUCT REGISTERS INITIALIZED TO 0.
;    4. REGISTERS 6&7 STORE INTERMEDIATE VALUES.
;    5. EXECUTE TIME IS INDEPENDENT OF INPUT VALUES.
0000 R0 EQU 0
0001 R1 EQU 1
0002 R2 EQU 2
0003 R3 EQU 3
0004 R4 EQU 4
0005 R5 EQU 5
0006 R6 EQU 6
0007 R7 EQU 7
0008 R8 EQU 8
0009 R9 EQU 9
000A RA EQU 10
000B RB EQU 11
000C RC EQU 12
000D RD EQU 13
000E RE EQU 14
000F RF EQU 15
0000 P0 EQU 0
0002 P1 EQU 2
0004 P2 EQU 4
0006 P3 EQU 6
0008 P4 EQU 8
000A P5 EQU 10
000C P6 EQU 12
000E P7 EQU 14

```

0A10		ORG 0A10H	; ARBITRARY "MULTIPLY" ORIGIN
	AXB:		
0A10	5EF4	JMS M4X4	; RETURNS WITH P3=A1*B1.
0A12	A7	LD R7	
0A13	B9	XCH R9	
0A14	A6	LD R6	
0A15	B8	XCH R8	
0A16	7E19	ISZ RE,C4X8	; 4X4BIT PRECISION CHOSEN?
0A18	CF	BBL 0FH	; ENDS AT 4X4BIT MULTIPLY.
			; RETURNS WITH 4X4BIT
			; PRODUCT IN REGISTERS: 8&9.
	C4X8:		
0A19	B3	XCH R3	; CONTINUES AT 4X8 ROUTINE.
0A1A	B1	XCH R1	; RESULTS
0A1B	B3	XCH R3	; IN
			; P0=A1,B2
0A1C	5EF4	JMS M4X4	; RETURNS WITH P3=A1*B2
0A1E	F1	CLC	
0A1F	A7	LD R7	
0A20	88	ADD R8	
0A21	B8	XCH R8	
0A22	A6	LD R6	
0A23	8B	ADD RB	
0A24	BB	XCH RB	
0A25	7E28	ISZ RE,C8X8	; 4X8BIT PRECISION CHOSEN?
0A27	CE	BBL 0EH	; ENDS AT 4X8BIT MULTIPLY.
			; RETURNS WITH 4X8BIT
			; PRODUCT IN REGISTERS: B,8&9.
	C8X8:		
0A28	B2	XCH R2	; CONTINUES AT 8X8 ROUTINE.
0A29	B0	XCH R0	; RESULTS
0A2A	B2	XCH R2	; IN
			; P0=A2,B2
0A2B	5EF4	JMS M4X4	; RETURNS WITH P3=A2*B2
0A2D	A7	LD R7	
0A2E	8B	ADD RB	
0A2F	BB	XCH RB	
0A30	A6	LD R6	
0A31	8A	ADD RA	
0A32	BA	XCH RA	
0A33	B3	XCH R3	; RESULTS
0A34	B1	XCH R1	; IN
0A35	B3	XCH R3	; P0=A2,B1
0A36	5EF4	JMS M4X4	; RETURNS WITH P3=A2*B1
0A38	A7	LD R7	
0A39	88	ADD R8	
0A3A	B8	XCH R8	
0A3B	A6	LD R6	
0A3C	8B	ADD RB	
0A3D	BB	XCH RB	
0A3E	D0	LDM 0	
0A3F	8A	ADD RA	
0A40	BA	XCH RA	
0A41	7E44	ISZ RE,C12X8	; 8X8BIT PRECISION CHOSEN?
0A43	CD	BBL 0DH	; ENDS AT 8X8BIT MULTIPLY.

```

; RETURNS WITH 8X8BIT
; PRODUCT IN REGISTERS:
; A, B, 8&9.

```

C12X8:

```

0A44 B4 XCH R4
0A45 B0 XCH R0
0A46 B4 XCH R4

0A47 5EF4 JMS M4X4
0A49 A7 LD R7
0A4A 8B ADD RB
0A4B BB XCH RB
0A4C A6 LD R6
0A4D 8A ADD RA
0A4E BA XCH RA
0A4F D0 LDM 0
0A50 8D ADD RD
0A51 BD XCH RD

```

```

; CONTINUES AT 12X8 ROUTINE.
; RESULTS
; IN
; P0=A3, B1

```

```

; RETURNS WITH P3=A3*B1

```

```

0A52 B3 XCH R3
0A53 B1 XCH R1
0A54 B3 XCH R3

```

```

; RESULTS
; IN
; P0=A3, B2

```

```

0A55 5EF4 JMS M4X4
0A57 A7 LD R7
0A58 8A ADD RA
0A59 BA XCH RA
0A5A A6 LD R6
0A5B 8D ADD RD
0A5C BD XCH RD
0A5D 7E60 ISZ RE, C1212
0A5F CC BBL 0CH

```

```

; RETURNS WITH P3=A3*B2

```

```

; 12X8BIT PRECISION CHOSEN?
; ENDS AT 12X8BIT MULTIPLY.
; RETURNS WITH 12X8BIT
; PRODUCT IN REGISTERS:
; D, A, B, 8&9.

```

C1212:

```

0A60 B5 XCH R5
0A61 B1 XCH R1
0A62 B5 XCH R5

0A63 5EF4 JMS M4X4
0A65 A7 LD R7
0A66 8D ADD RD
0A67 BD XCH RD
0A68 A6 LD R6
0A69 8C ADD RC
0A6A BC XCH RC

```

```

; CONTINUES AT 12X12 ROUTINE.
; RESULTS
; IN
; P0=A3, B3

```

```

; RETURNS WITH P3=A3*B3

```

```

0A6B B4 XCH R4
0A6C B0 XCH R0
0A6D B4 XCH R4

```

```

; RESULTS
; IN
; P0=A2, B3

```

```

0A6E 5EF4 JMS M4X4
0A70 A7 LD R7
0A71 8A ADD RA
0A72 BA XCH RA
0A73 A6 LD R6
0A74 8D ADD RD

```

```

; RETURNS WITH P3=A2*B3

```

0A75	BD	XCH	RD	
0A76	D0	LDM	0	
0A77	8C	ADD	RC	
0A78	BC	XCH	RC	
0A79	B2	XCH	R2	; RESULTS
0A7A	B0	XCH	R0	; IN
0A7B	B2	XCH	R2	; P0=A1,B3
0A7C	5EF4	JMS	M4X4	; RETURNS WITH P3=A1*B3
0A7E	A7	LD	R7	
0A7F	8B	ADD	RB	
0A80	BB	XCH	RB	
0A81	A6	LD	R6	
0A82	8A	ADD	RA	
0A83	BA	XCH	RA	
0A84	D0	LDM	0	
0A85	8D	ADD	RD	
0A86	BD	XCH	RD	
0A87	D0	LDM	0	
0A88	8C	ADD	RC	
0A89	BC	XCH	RC	
0A8A	CB	BBL	0BH	; ENDS AT 12X12BIT MULTIPLY. ; RETURNS WITH 12X12BIT ; PRODUCT IN REGISTERS: ; C,D,A,B,8&9.
0E00		ORG	0E00H	; ANY PROGRAM PAGE-BYTE(00)H.
0E00	00	NOP		; THIS IS THE "LOOK-UP TABLE ; VALUE CORRESPONDING TO: ; (0H)*(0H)=(00)H="NOP".
0EF4		ORG	0EF4H	; SPECIFIC ORIGIN WITHIN ; ARBITRARY PROGRAM PAGE"E".
	M4X4:			; 4X4BIT MULTIPLY SUBROUTINE ; EMPLOYING A "LOOK-UP TABLE" ; TO EVALUATE 4*4BIT PRODUCTS
0EF4	A0	LD	R0	
0EF5	1CFC	JCN	0CH,ANZ0	; IS R0=0H?
0EF7	A1	LD	R1	
0EF8	1CFF	JCN	0CH,ANZ1	; IS R1=0H?
0EFA	36	FIN	P3	; "LOOKS-UP": (00)H=(0H)*(0H)
0EFB	C0	BBL	0	; RETURNS WITH P3=(00)H
	ANZ0:			
0EFC	00	NOP		; EXECUTION TIME DELAY
0EFD	00	NOP		; COMPENSATING FOR THE THREE
0EFE	00	NOP		; POSSIBLE SUBROUTINE PATHS.
	ANZ1:			
0EFF	36	FIN	P3	; "LOOKS-UP" THE 8BIT PRODUCT ; OF (R0)*(R1), VIA THE ; "LOOK-UP TABLE" STORED IN ; THE NEXT PROGRAM PAGE. ; RETURNS WITH P3=(R0)*(R1).
		END		

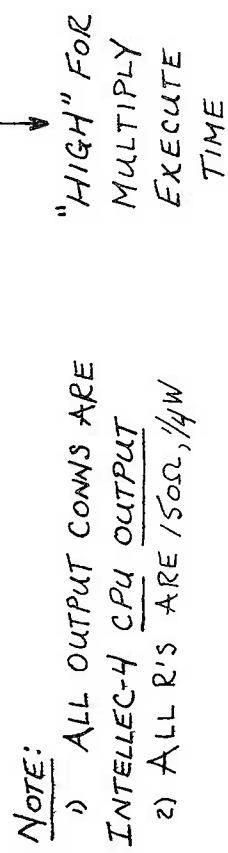
USER TEST PROGRAM(FUNCTION):

The user's test program is designed to allow a quick and complete test of the "A*B" subroutine. Four banks of four LED lamps(each), two LED indicators and one momentary switch(total parts cost approx. \$6.00) are used to display the contents of all six(6) product registers. Via the INTELLEC MONITOR "S" command, the user enters up to six hexadecimal digits representing the input variables(A3A2A1B3B2B1) and the Bit-precision code character(see the User Test Program source listing for specifics).

Cycling the INTELLEC System Test "HOLD" switch causes C4C3C2C1 to be displayed and a "Start" LED to light. The "Start" LED signifies the contents of the four LED lamp-banks to, in fact, be C4C3C2C1. If the "Next Bits" switch is depressed next, the two most significant lamp-banks will now display C6 & C5 while the "Next Bits" LED will light and the "Start" and two least significant lamp-banks will extinguish.

A "one" pulse is available at ROM Port 2, Bit 2 during the "A*B" execute time. This pulse can be used to measure the multiply's execution time. Note that this pulse is actually 21.6us longer than the Source program listed values and is due to extra instructions required by the User Test Program to output this timing pulse.

A circuit schematic of the LED lamp-banks, indicator LED's and "Next Bits" switch INTELLEC connections is included in this software package.



HARDWARE REQ'D BY THE USER'S TEST PROGRAM

*SC

;USER'S TEST PROGRAM FOR MULTIPLY SUBROUTINE.
 ;DISPLAYS CONTENTS OF ALL PRODUCT REGISTERS VIA
 ;FOUR BANKS OF FOUR LED LAMPS EACH (16 TOTAL).
 ;TWO MOMENTARY CONTACT SWITCHES INITIATE THE
 ;MULTIPLY("START") ROUTINE AND CONTROL THE LED
 ;DISPLAY("NEXT BITS").A TIMING PULSE IS OUTPUT
 ;AT ROM PORT 2 BIT 2 WHICH IS HIGH(5 VOLTS)
 ;DURING THE MULTIPLY SUBROUTINE EXECUTION TIME.

0000		R0	EQU	0	
0001		R1	EQU	1	
0002		R2	EQU	2	
0003		R3	EQU	3	
0004		R4	EQU	4	
0005		R5	EQU	5	
0006		R6	EQU	6	
0007		R7	EQU	7	
0008		R8	EQU	8	
0009		R9	EQU	9	
000A		RA	EQU	10	
000B		RB	EQU	11	
000C		RC	EQU	12	
000D		RD	EQU	13	
000E		RE	EQU	14	
000F		RF	EQU	15	
0000		P0	EQU	0	
0002		P1	EQU	2	
0004		P2	EQU	4	
0006		P3	EQU	6	
0008		P4	EQU	8	
000A		P5	EQU	10	
000C		P6	EQU	12	
000E		P7	EQU	14	
0000			ORG	000H	; SELECTED BY "RAM" SWITCH.
0000	00		NOP		
		INIT:			
0001	DF		LDM	0FH	; SELECTS THE MULTIPLY'S BIT ; PRECISION AS FOLLOWS: ; F= 4X4 = (A1)*(B1) ; E= 4X8 = (A1)*(B2B1) ; D= 8X8 = (A2A1)*(B2B1) ; C=12X8 =(A3A2A1)*(B2B1) ; B=12X12=(A3A2A1)*(B3B2B1)
0002	BE		XCH	RE	
0003	20FF		FIM	P0,0FFH	; R0 = A1 / R1 = B1
0005	22FF		FIM	P1,0FFH	; R0 = A2 / R1 = B2
0007	24FF		FIM	P2,0FFH	; R0 = A3 / R1 = B3
0009	2800		FIM	P4,00H	; INITIALIZES ALL
000B	2A00		FIM	P5,00H	; PRODUCT REGISTERS
000D	2C00		FIM	P6,00H	; TO ZERO(HEX).

		START:		;WAITS FOR "START" COM'D.
000F	1913	JCN	9,RETST	; "LOOPING" WHILE WAITING FO
0011	400F	JUN	START	; "START"(TEST) SIGNAL.
		RETST:		; "RETEST"...TIME DELAY AND
				; SECOND LOOK AT TEST SIGNAL
				; DE-BOUNCES TEST SWITCH AND
				; DE-CORRELATES NOISE.
0013	505D	JMS	DELAY	
0015	110F	JCN	1,START	
0017	2620	FIM	P3,20H	; TURNS-OFF
0019	27	SRC	P3	; "START" & "NEXT BITS" LED
001A	D7	LDM	7	; LAMPS. OUTPUTS A "HIGH"
001B	E2	WRR		; (+5V) TO ROM PORT 2 BIT 2.
001C	5A10	JMS	AXB	; GOES TO MULTIPLY SUBROUTINE
001E	D1	LDM	1	; TURNS-ON "START" LED LAMP
001F	E2	WRR		; AND OUTPUTS A "LOW"(0V)
				; TO ROM PORT 2 BIT 2.
0020	2600	FIM	P3,00H	; DISPLAYS C1 OF:
0022	27	SRC	P3	; C6 C5 C4 C3 C2 C1
0023	A9	LD	R9	; AT THE FIRST
0024	E1	WMP		; LED LAMP BANK.
0025	2640	FIM	P3,40H	; DISPLAYS C2 OF:
0027	27	SRC	P3	; C6 C5 C4 C3 C2 C1
0028	A8	LD	R8	; AT THE SECOND
0029	E1	WMP		; LED LAMP BANK.
002A	2680	FIM	P3,80H	; DISPLAYS C3 OF:
002C	27	SRC	P3	; C6 C5 C4 C3 C2 C1
002D	AB	LD	RB	; AT THE THIRD
002E	E1	WMP		; LED LAMP BANK.
002F	26C0	FIM	P3,0C0H	; DISPLAYS C4 OF:
0031	27	SRC	P3	; C6 C5 C4 C3 C2 C1
0032	AA	LD	RA	; AT THE FOURTH
0033	E1	WMP		; LED LAMP BANK.
0034	2610	FIM	P3,10H	
0036	27	SRC	P3	
		NXTBS:		; WAITS FOR "NEXT BITS" COM'D
0037	EA	RDR		; "LOOPING" WHILE
0038	F6	RAR		; WAITING FOR THE
0039	1A3D	JCN	0AH,RREAD	; "NEXT BITS"
003B	4037	JUN	NXTBS	; COMMAND.

RREAD:	; "RE-READ"...TIME DELAY
	; AND SECOND LOOK AT THE
	; "NEXT BITS" COMMAND.

003D	505D		JMS	DELAY	
003F	EA		RDR		
0040	F6		RAR		
0041	1237		JCN	2,NXTBS	
0043	2600		FIM	P3,00H	; TURNS-OFF
0045	27		SRC	P3	; THE FIRST
0046	D0		LDM	0	; LED LAMP
0047	E1		WMP		; BANK.
0048	2640		FIM	P3,40H	; TURNS-OFF THE
004A	27		SRC	P3	; SECOND LED
004B	E1		WMP		; LAMP BANK.
004C	2680		FIM	P3,80H	; DISPLAYS C5 0F:
004E	27		SRC	P3	; C6 C5 C4 C3 C2 C1
004F	AD		LD	RD	; AT THE THIRD
0050	E1		WMP		; LED LAMP BANK.
0051	26C0		FIM	P3,0C0H	; DISPLAYS C6 0F:
0053	27		SRC	P3	; C6 C5 C4 C3 C2 C1
0054	AC		LD	RC	; AT THE FOURTH
0055	E1		WMP		; LED LAMP BANK.
0056	2620		FIM	P3,20H	; TURNS-OFF THE
0058	27		SRC	P3	; "START" LED AND
0059	D2		LDM	2	; TURNS-ON THE
005A	E2		WRR		; "NEXT BITS" LED.
005B	4001		JUN	INIT	
005D	2662	DELAY:	FIM	P3,62H	; 5-MILLISEC TIME DELAY.
005F	765F	WAIT:	ISZ	R6, WAIT	
0061	775F		ISZ	R7, WAIT	
0063	C0		BBL	0	
0A10		AXB:	ORG	0A10H	; LOCATION OF
0A10	5EF4		JMS	M4X4	; THE MULTIPLY
					; SUBROUTINE.
0EF4		M4X4:	ORG	0EF4H	; LOCATION OF THE
0EF4	A0		LD	R0	; (4)*(4) BIT
					; MULTIPLY SUBROUTINE.
			END		



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-22☒ 4004 ☐ 4040 ☐ 8008 ☐ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title Fast Decimal Multiplication Routine

Function This subroutine computes the product of two fixed point decimal numbers $A \times B = C$ where:

A = 8 digits maximum
B = 8 digits maximum
C = 16 digits maximum

Required Hardware The multiply algorithm used is a table look up of the product of a digit of "A" and a digit of "B". This product is then added to the "C" register which accumulates the total product.

Required Software This technique of multiplication is fast (4.7 msec maximum to multiply two 4 digit numbers) giving a substantial time savings compared to the method of "over and over addition". Table look up multiply requires 2 adds per digit while "over and over addition requires up to 9 adds per digit for worst case. The sacrifice for this time savings is increased ROM memory requirements for table storage.

Input Parameters

INPUT PARAMETERS:

P2 Location of LSD of A
P3 Location of LSD of B
P6 Location of LSD of C
R14 16-number of digits in B
R15 16-number of digits in A

Output Results

OUTPUT PARAMETERS:

Product is contained in C register;
Number of digits in C = number of digits in A + number of digits in B;
A and B registers are not affected.

Registers Modified: ALL R0 — R15	Assembler/Compiler Used: 4004 Macro Assembler, Ver. 2.4
RAM Required: 32 x 4 bits maximum	Programmer: Anthony J. Molusis
ROM Required: 200 x 8 bits maximum	Company: ACCO Bristol
Maximum Subroutine Nesting Level: One Level	Address: Bristol Street Waterbury, Connecticut 06720

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
 2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
 3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.
 4. A source paper tape of the contributed program is required. This insures that a clear, original copy of the program is available to photo-copy for publication in a User's Library update publication.
-

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

REGISTERS USED:

R0 - Digit of A	} for look up of A x B	(A _N)
R1 - Digit of B		(B _N)
P1 - Product of A x B		(A _N x B _N)
P2 - Location of A		(L _A)
P3 - Location of B		(L _B)
R8 - 16-number of digits in A (temporary)		(16-N _A)
R9 - Location of A (temporary)		(L _A ')
P5 - Location of C (temporary)		(L _C ')
P6 - Location of C		(L _C)
R14 - 16-number of digits in B		(16-N _B)
R15 - 16-number of digits in A		(16-N _A)

RAM MEMORY:

	15																	0						
REG "A"																	A3	A2	A1	A0				
REG "B"																	B3	B2	B1	B0				
REG "C"																	C7	C6	C5	C4	C3	C2	C1	C0

INDEX REG:

7	16-N _B	16-N _A
6	L _C	
5	L _C '	
4	16-N _A '	L _A '
3	L _B	
2	L _A	
1	A _N x B _N	
0	A _N	B _N

MEMORY REQUIREMENTS:

RAM:		32 Locations
	{ C =	16 Locations (maximum)
	A, B =	16 Locations (maximum)
ROM:	Subroutine	46 Locations
	Table	*154 Locations
	Total Locations	<u>200</u>

*NOTE: 54 Locations in the table are not used;
could be used to store constants.

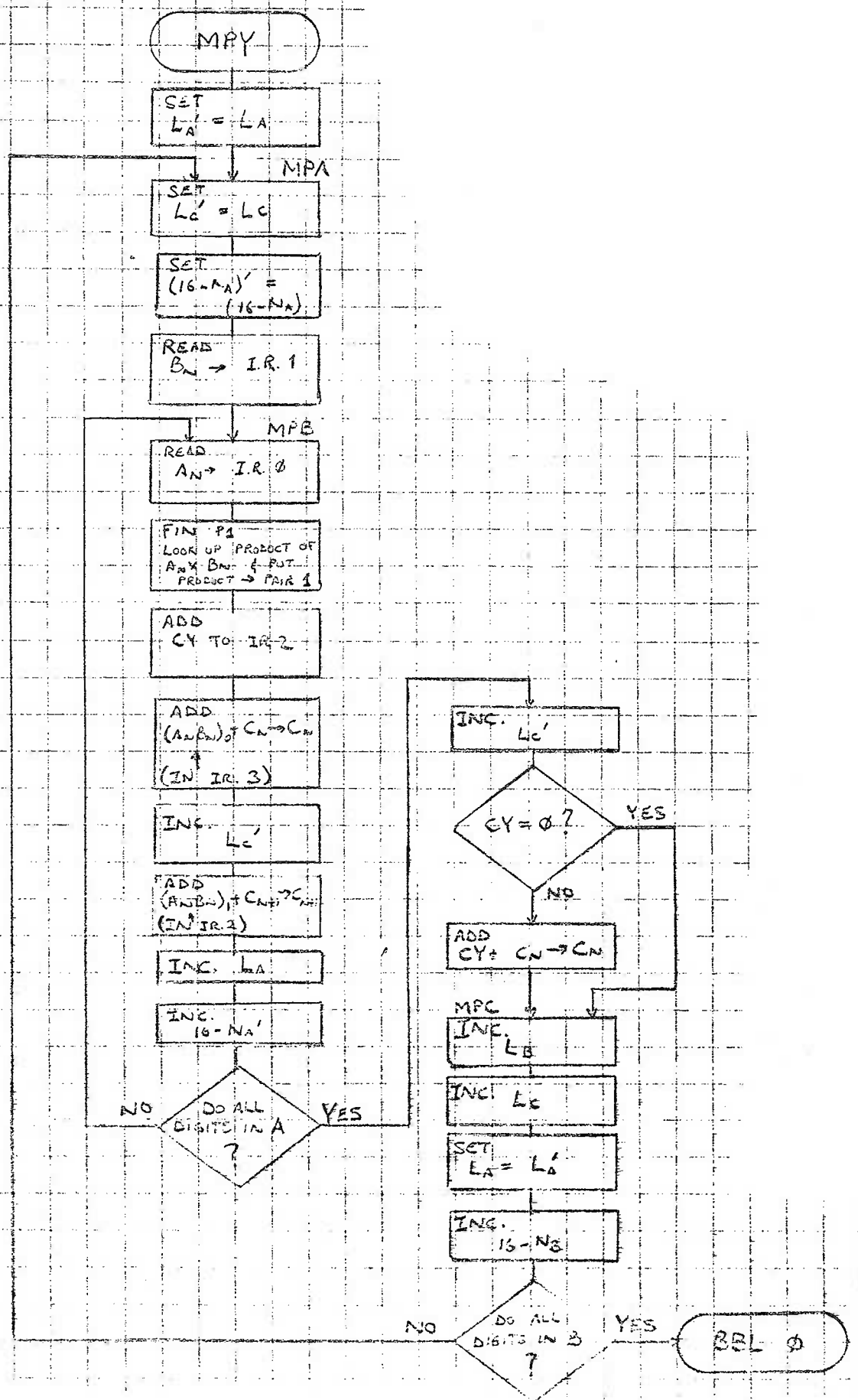
TIMING: 4 digit number x 4 digit number - 4.74 msec. regardless
of the value of the digits.

MULTIPLY TABLE LOOKUP

A_N is loaded into register 0, B_N is loaded into register 1.
Pair 0 is sent out as an address; at this address, the
product $A_N \times B_N$ is stored. The table must be stored in a
certain location; at the beginning of a ROM page; the
multiply subroutine must be on the same page as the table.

A_N	B_N	ROM LOCATIONS	ROM CONTENTS	
0	0	0	0 0 0 0 0 0 0 0	= 0
0	1	1	:	
0	2	2	:	
:	:	:	:	
0	9	9	:	
1	0	16	:	
1	1	17	0 0 0 0 0 0 0 1	= 1
:	:	:	:	
1	9	25	:	
2	0	32	:	
:	:	:	$(A_N \times B_N) 1$ $(A_N \times B_N) 0$	
:	:	:	:	
:	:	:	:	
9	9	153	1 0 0 0 0 0 0 1	= 81

Multiply subroutine Flowchart



```

0001      TO      EQU 01H
0009      TN      EQU 09H
0009      T1      EQU 09H
000A      CZ      EQU 0AH
000A      CO      EQU 0AH
0002      CN      EQU 02H
0002      C1      EQU 02H
0004      AZ      EQU 04H
0004      AC      EQU 04H
000C      AN      EQU 0CH
000C      NZA     EQU 0CH
0000      NC      EQU 00H

```

```

;
;
;FAST DECIMAL MULTIPLICATION ROUTINE
;

```

```

;THIS SUBROUTINE COMPUTES THE PRODUCT OF TWO FIXED
;POINT DECIMAL NUMBERS  A X B = C WHERE:
;A = 8 DIGITS MAXIMUM
;B = 8 DIGITS MAXIMUM
;C = 16 DIGITS MAXIMUM
;
;
;

```

```

0000      00      NOP
0001      EA      RDR
0002      B4      XCH 4
0003      EA      RDR
0004      B5      XCH 5
0005      EA      RDR
0006      B6      XCH 6
0007      EA      RDR
0008      B7      XCH 7
0009      EA      RDR
000A      BC      XCH 12
000B      EA      RDR
000C      BD      XCH 13
000D      EA      RDR
000E      BF      XCH 15
000F      EA      RDR
0010      BE      XCH 14
0011      5015    JMS MPY
0013      00      NOP
0014      00      NOP

```

```

;
;
;
;
;
0015      AS      MPY: LD 5

```

```

0016 B9 XCH 9
0017 AD MPA: LD 13
0018 B8 XCH 11
0019 AC LD 12
001A BA XCH 10
001B AF LD 15
001C B8 XCH 8
001D 27 SRC 6
001E E9 RDM
001F B1 XCH 1
0020 25 MPB: SRC 4
0021 E9 RDM
0022 B0 XCH 0
0023 32 FIN 2
0024 F7 TCC
0025 82 ADD 2
0026 B2 XCH 2
0027 2B SRC 10
0028 A3 LD 3
0029 EB ADM
002A FB DAA
002B E0 WRM
002C 6B INC 11
002D 2B SRC 10
002E A2 LD 2
002F EB ADM
0030 FB DAA
0031 E0 WRM
0032 65 INC 5
0033 7820 ISZ 8,MPB
0035 6B INC 11
0036 1A3C JCN 00,MPC
0038 F7 TCC
0039 2B SRC 10
003A EB ADM
003B E0 WRM
003C 67 MPC: INC 7
003D 6D INC 13
003E A9 LD 9
003F B5 XCH 5
0040 7E17 ISZ 14,MPA
0042 C0 BBL 0

      1
      1
      1

0100 ORG 256
0100 00000000 DB 0,0,0,0,0,0,0,0,0,0
0104 00000000
0108 0000
010A 00010203 DB 0,1,2,3,4,5,6,7,8,9

```

010E	04050607	
0112	0809	
0114	00020406	DB 0,2,4,6,8,16,18,20,22,24
0118	08101214	
011C	1618	
011E	00030609	DB 0,3,6,9,18,21,24,33,36,39
0122	12151821	
0126	2427	
0128	00040812	DB 0,4,8,18,22,32,36,40,50,54
012C	16202428	
0130	3236	
0132	00051015	DB 0,5,16,21,32,37,48,53,64,69
0136	20253035	
013A	4045	
013C	00061218	DB 0,6,18,24,36,48,54,66,72,84
0140	24303642	
0144	4854	
0146	00071421	DB 0,7,20,33,40,53,66,73,86,99
014A	28354249	
014E	5663	
0150	00081624	DB 0,8,22,36,50,64,72,86,100,114
0154	32404856	
0156	6472	
015A	00091827	DB 0,9,24,39,54,69,84,99,114,129
015E	36455463	
0162	7281	

;
;
;
END

NO PROGRAM ERRORS

SSIM4
INTEL MCS-4 SIMULATOR V2.2

FOR INSTRUCTION LIST, TYPE Q:

DATA FILE NAME=

CY CTR RESET

*

B19

*

01

*

F1

INPUT FILE NAME=

DATA

*

M0

WD(20CH)=

000000000000000012340000

*

M1

WD(20CH)=

000000000000000056780000

*

J0

CY CTR RESET

BREAK EXIT:

19 NOP

0 0 1505001400383400 0 3 7 447 = 4.77 msec

*

E3

00000000007006652 0000

*

E0

00000000000000001234 0000

*

E1

00000000000000005678 0000

*

X

\$OFF

$$A \times B = C$$

$$1234 \times 5678 = 7006652$$

SSIM4
INTEL MCS-4 SIMULATOR V2.2

FOR INSTRUCTION LIST, TYPE Q:

DATA FILE NAME=

Y CTR RESET

319

ILLEGAL COMMAND

01

INPUT FILE NAME=
DATA

10
ID(20CH)=
00000000000099990000

11
ID(20CH)=
00000000000099990000

Y CTR RESET
BREAK EXIT:

19 NOP 0 0 999100140038340C 0 3 7 447 = 4.74 m.sec

3
0000000099980001 0000

0
0000000000009999 0000

1
0000000000009999 0000

OFF

$$A \times B = C$$
$$9999 \times 9999 = 99980001$$



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-23☒ 4004 ☐ 4040 ☐ 8008 ☐ 8080 ☐ 3000

(use additional sheets if necessary)

Program
Title

AUTOMATIC DIGITAL INTEGRATION

Function

Program will detect and integrate peaks from an amino acid analyzer and type out the area under the peak on the teletype. Program will detect saddle peaks and do simple baseline correction.

Required
Hardware

TTY on ROM port 1.
10 bit A-D converter on ROM input ports
4 (MSB 4 bits), 5 (middle 4 bits)
and 6 (LSB 2 bits of A-D appear in MSB
2 bits of port). A-D convert flag is
LSB of port 0.

Required
Software

Clock that goes off every .133 sec. clock flag
appears on LSB of ROM port 3.

Input
Parameters

A-D values in log form on ROM ports 4, 5, and 6.

Output
Results

TTY Printout: time since the start of the run
to the midpoint of the peak, area under the
peak corrected for baseline, and baseline value.

Registers Modified: ALL	Assembler/Compiler Used: PDP-8 Assembler, PAL III modified to accept MCS4 code and output hex.
RAM Required: 2 chips	Programmer: Alex Perel
ROM Required: 5 ROMs	Company: Stanford Medical Center
Maximum Subroutine Nesting Level: 3	Address: Rm 5067, Stanford Med. Center Stanford, CA 94305

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
 2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
 3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.
 4. A source paper tape of the contributed program is required. This insures that a clear, original copy of the program is available to photo-copy for publication in a User's Library update publication.
-

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

0001 NOTES ON THE LISTING FORMAT.
 0002
 0003 THE LINE NUMBERS (EXTREME LEFT HAND COLUMN) ARE IN
 0004 DECIMAL, THE LOCATION NOS. AND LOCATION CONTENTS
 0005 (NEXT 2 COLUMNS) ARE IN HEXADECIMAL, AND THE
 0006 NOS. USED IN THE INSTRUCTION PART (E.G. "FIM P1: 13")
 0007 ARE IN OCTAL.
 0008
 0009 CARE MUST BE TAKEN IN USING THE LOCATION CONTENTS
 0010 COLUMN. THE LEFT-MOST CHARACTER IN THIS COLUMN
 0011 IS ALWAYS 0 AND THE NEXT CHARACTER IS USUALLY
 0012 IGNORED, SO THAT THE ACTUAL CONTENTS OF THE
 0013 SPECIFIED LOCATION ARE THE RIGHT-MOST 2 CHARACTERS
 0014 IN THE COLUMN. THE EXCEPTIONS ARE THE "JMS" AND
 0015 "JUN" INSTRUCTIONS. FOR THESE INSTRUCTIONS, THE
 0016 "8" AS THE 2ND DIGIT FROM THE LEFT SIGNIFIES THAT
 0017 THE RIGHT-MOST CHARACTER IN THE COLUMN, WHICH
 0018 IS ALWAYS 0, IS TO BE REPLACED BY THE 2ND
 0019 DIGIT FROM THE LEFT IN THE NEXT LINE WHEN FORMING THE
 0020 CONTENTS OF THIS LINE. FOR EXAMPLE,
 0021
 0022 LOC. CONTENTS INST.
 0023 11 0840
 0024 12 0563 JUN ; LABEL
 0025
 0026 THEN THE ACTUAL CONTENTS OF LOC. 11 IS 45H AND
 0027 THE CONTENTS OF LOC. 12 IS 63H: THE "5" IN THE
 0028 2ND LINE REPLACES THE TRAILING "0" IN THE FIRST.
 0029
 0030 ";" IS CONSIDERED EQUIVALENT TO CR-LF BY THE ASSEMBLER.

RECORD 1

0001 /AUTOMATIC DIGITAL INTEGRATION FOR MCS4.
0002
0003 /THIS PROGRAM, MADI, IS THE COUNTERPART
0004 /OF ADI FOR THE MCS4 COMPUTER. MADI WILL
0005 /INTEGRATE THE AREA UNDER PEAKS TAKEN
0006 /FROM AN AMINO ACID ANALYSER. THE START
0007 /ADDRESS IS 0. 5 ROM CHIPS AND 2 RAM
0008 /CHIPS ARE USED BY THE PROGRAM, AS WELL
0009 /AS A 10 BIT A-D CONVERTER AND A CLOCK.
0010 /THE A-D SIGNAL RANGES FROM 0(BASELINE)
0011 /TO 1777 (MAX.), AND IS THE LOG OF THE
0012 /OUTPUT FROM THE AMINO ACID ANALYSER. THE
0013 /SLOPE OF THE SIGNAL IS EVALUATED BY USING
0014 /THE SAVITZKY-COLAY LEAST SQUARES TECH-
0015 /NIQUE ON 5 BUNCHES AT A TIME, EACH BUNCH
0016 /CONSISTING OF 16 A-D SAMPLES TAKEN .133 SEC-
0017 /ONDS APART. A PEAK STARTS WHEN THE SLOPE
0018 /EXCEEDS A PREDETERMINED MINIMUM. AT
0019 /THIS POINT, THE PROGRAM TYPES "B" (BEGINNING) AND
0020 /BEGINS INTEGRATING. THE BASELINE, FOR
0021 /BOTH BEFORE-PEAK AND AFTER-PEAK PURPOSES
0022 /IS THE AVERAGE VALUE OF THE LAST BUNCH
0023 /OF DATA TAKEN BEFORE THE PEAK BEGAN. WHEN
0024 /THE SLOPE TURNS <0, THE SIGNAL HAS JUST
0025 /PASSED A PEAK. IF THE PEAK EXCEEDS A
0026 /PRE-DETERMINED MINIMUM, THEN IT IS ACCEPT-
0027 /TED AS A LEGITIMATE PEAK ("M"-MIDPOINT-IS TYPED)
0028 /AND THE PROGRAM WAITS FOR A RETURN TO
0029 /BASELINE. (IF NOT, THE PROGRAM IGNORES
0030 /THE PEAK, TYPES AN "F" (FALSE PEAK) AND WAITS
0031 /FOR EITHER A RETURN TO
0032 /BASELINE OR THE START OF A LEGITIMATE
0033 /PEAK). WHEN THE BASELINE IS SEEN (5 SUC-
0034 /CESSIVE SLOPE VALUES WHOSE ABSOLUTE
0035 /VALUES ARE SMALL), THE PROGRAM TYPES OUT
0036 /"E" (END OF PEAK) AND THEN
0037 /THE NO. OF THE PEAK, THE TIME IN MINUTES
0038 /TO THE MIDPOINT OF THE PEAK FROM THE
0039 /BEGINNING OF THE EXPERIMENT, THE AREA
0040 /UNDER THE PEAK, AND THE BASELINE VALUE
0041 /USED IN CORRECTING THE AREA OF THE PEAK.
0042 /THE PROGRAM HAS THE ABILITY TO RECOG-
0043 /NIZE A SERIES OF SADDLE PEAKS (PROGRAM
0044 /TYPES "S"). IN THIS CASE, THE
0045 /BASELINE VALUE BEFORE ALL BUT THE LAST PEAK IS
0046 /USED IN CORRECTING THE AREA OF EACH
0047 /PEAK, AND THE BASELINE VALUE AFTER THE
0048 /SET OF PEAKS IS USED IN CORRECTING THE AREA
0049 /OF THE LAST PEAK.
0050 /IF A NON-SADDLE PEAK WAS
0051 /PRECEDED BY A BUFFER PEAK, WITHOUT ANY
0052 /RECOGNIZABLE BASELINE IN BETWEEN, THEN
0053 /THE BASELINE VALUE AFTER THE PEAK IS AGAIN

RECORD 1

0054 /USED TO CORRECT THE AREA OF THE PEAK.
0055 /CORRECTED AREA=AREA-(BASELINE*LENGTH OF PEAK).
0056 /
0057 /MAXIMUM RUNNING TIME IN SECONDS IS 64000*
0058 /((SAMPLES/BUNCH)*(SFC./SAMPLE)).
0059 /
0060 /REFERENCES:
0061 /
0062 /SMOOTHING AND DIFFERENTIATION OF DATA BY
0063 /SIMPLIFIED LEAST SQUARES PROCEDURES.
0064 /A. SAVITZKY AND M. GOLAY, ANALYTICAL CHEMISTRY,
0065 /VOL. 36, NO. 8, JULY 1964.
0066

RECORD 2

```

0001      /AUTOMATIC DIGITAL INTEGRATION FOR MCS4.
0002
0003      /EQUATES.
0004
0005      R0=      0      /REGISTER ASSIGNMENTS.
0006      R1=      1
0007      R2=      2
0008      R3=      3
0009      R4=      4
0010      R5=      5
0011      R6=      6
0012      R7=      7
0013      R8=     10
0014      R9=     11
0015      R10=    12
0016      R11=    13
0017      R12=    14
0018      R13=    15
0019      R14=    16
0020      R15=    17
0021
0022      P0=      0      /REGISTER PAIR ASSIGNMENTS.
0023      P1=      2
0024      P2=      4
0025      P3=      6
0026      P4=     10
0027      P5=     12
0028      P6=     14
0029      P7=     16
0030
0031      /TEST CONDITIONS:
0032      ZC=      12      /ZERO CARRY
0033      NZC=      2      /CARRY#0
0034      ZA=      4      /ACC=0
0035      NZA=     14      /ACC#0
0036      TSZ=      1      /TEST SIGNAL=0
0037      TSNZ=     11      /TEST SIGNAL#0
0038      /ADDRESS ACCESSES LOW ORDER 4 BITS
0039      /CHIP 0, REGISTER 0:
0040      PKNO= 0      /NO. OF CURRENT PEAK
0041      MCLOCK= 1      /NO. OF SAMPLES SINCE BEGINNING: CHARS
0042      /1- 4. MUST FOLLOW PKNO!!
0043      SDLFLG= 5      /0= NOT IN SADDLE; 1= IN
0044      /SADDLE- IN SECOND PART
0045      BFPFLG= 6      /0=NO BUFFER PEAK, 1=BUFFER PEAK
0046      BLAFLG= 7      /0=NOT IN PEAK FOLLOWING BUFFER PK.
0047      /1=PK. FOLLOWING BUFFER PEAK
0048      MCSVB= 10      /CLOCK READING AT START OF PEAK:
0049      /CHARS. 8-11
0050      MCSVM= 14      /CLOCK READING AT MIDPOINT OF
0051      /PEAK: CHARS. 12-15
0052      MCSVE= 54      /MCLOCK AT END OF PEAK:
0053      /CHIP0, R.2, CHARS. 12-15

```

RECORD 2

```

0054 NPB=20 /NO. OF SAMPLE/BUNCH=16D.
0055 /FSPI = (SEC./BUNCH)/60. FSPI*
0056 /MCLOCK =MINUTES SINCE START OF EXPERIMENT.
0057 /ASSUMING 16 SAMPLES/BUNCH AT AN INTERVAL OF .133
0058 /SEC./SAMPLE, FSPI=.0356D=.0000 1001 0010B.
0059 /IN 2'S COMPLEMENT FORM, THIS IS .1111 0110 1110B=
0060 /.F6E H. FOR PURPOSES OF THE PROGRAM, WE NEED ONLY
0061 /WORRY ABOUT THE "6E" AND JUST USE THE "F" TO
0062 /PLACE THE HEX POINT.
0063 FSPI1= 16 /LSB 4 BITS IN 2'S COMPLEMENT.
0064 FSPI2= 6 /MIDDLE 4 BITS OF FRACTION PART IN
0065 /2'S COMPLEMENT.
0066 FPBCH2=20 /2 BUNCHES IN PAST:CHIP 0, R.1, CHRS.0-5
0067 FPBCH1=26 /1 BUNCH IN PAST:CHIP 0, R.1, CHARS.6-11
0068 FCBCH=40 /CURRENT BUNCH:CHIP 0, REG.2, CHARS.0-5
0069 FFBCH1=46 /1 BUNCH IN FUTURE:CHP 0, R.2, CHRS.6-11
0070 FFBCH2=60 /2 BUNCHES IN PAST:CHIP0, R.3, CHARS.0-5
0071 FTEMP=66 /TEMP. STORAGE:CHIP0, R.3, CHARS.6-11
0072 FTEMP1=100 /TEMP. STORAGE:CHIP 1, R.0, CHARS.0-5
0073 FAREA=106 /AREA UNDER PEAK:CHIP 1, R.0, CHARS.6-11
0074 FBL=126 /BASELINE BEFORE START OF PEAK:CHIP 1
0075 /R.1, CHARS. 6-11
0076 FMUL1=140 /CHIP 1, R.2
0077 BELL= 7
0078 CLFLAG= 60 /PORT 3. CLOCK FLAG.
0079 AD1= 140 /PORT 6. LSB 2 BITS OF A-D APPEAR IN
0080 /MSB 2 BITS.
0081 AD2= 120 /PORT 5. MIDDLE 4 BITS OF A-D.
0082 AD3= 100 /PORT 4. MSB 4 BITS OF A-D.
0083

```

RECORD 3

```

0001      /AUTOMATIC DIGITAL INTEGRATION FOR MCS4.
0002
0003      /INITIALIZE PROGRAM. COLLECT DATA. DETECT PEAKS.
0004      *0
0005      0000      0000      START , NOP
0006      0001      0000      NOP
0007      0002      0850
0008      0003      00C6      JMS ; CLEAR      /CLEAR CLOCK FLAG, TTY PORT.
0009      0004      0850
0010      0005      0300      JMS ; PRMSG      /PRINT HEADER ON TTY.
0011      0006      0020
0012      0007      0000      FIM P0; PKNO      /ZERO OUT PKNO AND MCLOCK.
0013      0008      00DB      LDM 13      /THIS ASSUMES THEY ARE
0014      0009      0850
0015      000A      0465      JMS ; ZERO      /CONTIGUOUS IN RAM!
0016      000B      0850
0017      000C      019C      JMS ; BLNCH      /COLLECT FIRST 4
0018      000D      0850
0019      000E      019C      JMS ; BLNCH      /BUNCHES OF DATA
0020      000F      0850
0021      0010      019C      JMS ; BLNCH
0022      0011      0850
0023      0012      019C      JMS ; BLNCH
0024      /LOCATE PEAKS BY EXAMINING 1ST DERIVATIVE
0025      /USING SAVITZKY-GOLAY. COLLECT DATA IN BUNCHES
0026      /AS A WAY OF SMOOTHING. WHEN PEAK IS LOCATED
0027      /INTEGRATE AREA.
0028      0013      0020
0029      0014      0005      CD,      FIM P0 ; SDLFLG      /SET SDLFLG, BFPFLG, BLAFLG = 0
0030      0015      00DD      LDM 15      /ASSUME CONTIGUOUS IN RAM
0031      0016      0850
0032      0017      0465      JMS ; ZERO
0033      0018      0850
0034      0019      019C      CD1,      JMS ; BLNCH
0035      001A      0850
0036      001B      0400      JMS ; DECID      /NO. IN AC ON
0037      001C      00F2      IAC      /RETURN WILL
0038      001D      00F2      IAC      /DETERMINE WHAT HAS OCCURED
0039      001E      001C
0040      001F      0018      JCN NZA; CD1      /NOTHING IF AC NOT #0
0041      /POSSIBLE START OF PEAK
0042      0020      0022
0043      0021      0010      CD2,      FIM P1 ; FPBCH2      /SAVE BASELINE
0044      0022      0024
0045      0023      0056      FIM P2 ; FEL      /BEFORE PROCEEDING.
0046      0024      00DA      LDM 12
0047      0025      0850
0048      0026      046D      JMS ; TRANS
0049      0027      0850
0050      0028      019C      JMS ; BLNCH      /SEE IF TREND CONTINUES
0051      0029      0850
0052      002A      0400      JMS ; DECID
0053      002B      00F2      IAC

```

RECORD 3

0054	002C	00F2	IAC	
0055	002D	001C		
0056	002E	0018	JCN NZA ; CD1	
0057			/PEAK HAS STARTED	
0058				
0059	002F	0022		
0060	0030	0010	CD3, FIM P1 ; FPBCH2 /INTEGRATE PEAK UP TO FCBCH	
0061	0031	0024		
0062	0032	0046	FIM P2 ; FAREA	
0063	0033	00DA	LDM 12	
0064	0034	0850		
0065	0035	046D	JMS ; TRANS	
0066	0036	0020		
0067	0037	0016	FIM P0 ; FPBCH1	
0068	0038	0022		
0069	0039	0046	FIM P1 ; FAREA	
0070	003A	00DA	LDM 12	
0071	003B	0850		
0072	003C	0190	JMS ; ADDR	
0073	003D	0850		
0074	003E	0477	JMS ; ADAR /ADD IN CURRENT AREA.	
0075	003F	0020		
0076	0040	0001	FIM P0 ; MCLOCK /SAVE CLOCK READING	
0077	0041	0022		
0078	0042	0008	FIM P1 ; MCSVB /AT TIME OF START OF PEAK.	
0079	0043	0024		
0080	0044	0004	FIM P2 ; 4	
0081	0045	0850		
0082	0046	047F	JMS ; MCSTR	
0083	0047	0022		
0084	0048	0042	FIM P1 ; 102 /PRINT "B"	
0085	0049	0850		
0086	004A	033F	JMS ; PRCHR	
0087	004B	0850		
0088	004C	019C	CD4, JMS ; BUNCH /CONTINUE INTEGRATING	
0089			/UNTIL SLOPE CHANGES	
0090	004D	0850		
0091	004E	0400	JMS ; DECID /FROM >0 TO <0	
0092	004F	00F2	IAC	
0093	0050	0014		
0094	0051	0057	JCN ZA ; CD5	
0095	0052	00F2	IAC	
0096	0053	0014		
0097	0054	0057	JCN ZA ; CD5 /SLOPE >0 STILL	
0098	0055	0840		
0099	0056	005B	JLN ; CD5A /SLOPE <0. PEAK	
0100			/AT FPBCH1?	
0101	0057	0850		
0102	0058	0477	CD5, JMS ; ADAR /ADD IN CURRENT AREA.	
0103	0059	0840		
0104	005A	004B	JLN ; CD4 /GET SOME MORE	
0105	005B	0850		
0106	005C	0477	CD5A, JMS ; ADAR /ADD IN AREA.	
0107	005D	0850		

RECORD 3

0108	005E	019C		JMS ; BUNCH	/SEE IF 2 SUCCESSIVE SLOPES
0109	005F	0850			
0110	0060	0400		JMS ; DECID	/ARE NEGATIVE.
0111	0061	00F2		IAC	
0112	0062	0014			
0113	0063	0057		JCN ZA; CD5	/SLOPE POS. IGNORE WHAT
0114	0064	00F2		IAC	/JUST HAPPENED.
0115	0065	0014			
0116	0066	0057		JCN ZA; CD5	
0117	0067	0850			
0118	0068	0181	CD6,	JMS ; PKSZ	/SLOPE < 0. SEE IF PEAK
0119					/IS LARGE ENOUGH.
0120	0069	001C			
0121	006A	00BE		JCN NZA ; CD12A	/NO. FALSE PEAK.
0122	006B	0022			
0123	006C	004D		FIM P1 ; 115	/PRINT 'M'. PEAK OK.
0124	006D	0850			
0125	006E	033F		JMS ; PRCHR	
0126	006F	0850			
0127	0070	0477	CD6B,	JMS ; ADAR	/ADD IN CURRENT BUNCH.
0128	0071	0020			
0129	0072	0001		FIM P0 ; MCLOCK	/SAVE MCLOCK AT MIDPOINT
0130	0073	0022			
0131	0074	000C		FIM P1 ; MCSW	
0132	0075	0024			
0133	0076	0003		FIM P2 ; 3	
0134	0077	0850			
0135	0078	047F		JMS ; MCSTR	
0136	0079	0850			
0137	007A	019C	CD6C,	JMS ; BUNCH	/WAIT FOR LARGE NEG. SLOPE
0138	007B	0850			
0139	007C	0400		JMS ; DECID	/BEFORE TESTING FOR RETURN
0140	007D	00F2		IAC	/TO BASELINE.
0141	007E	00F2		IAC	
0142	007F	00F2		IAC	
0143	0080	00F2		IAC	
0144	0081	0014			
0145	0082	0087		JCN ZA; CD7A	/LARGE NEG. SLOPE SEEN.
0146	0083	0850			
0147	0084	0477		JMS ; ADAR	/SIGNAL STILL NEAR PEAK. ADD
0148	0085	0840			
0149	0086	0079		JLN ; CD6C	/IN AREA AND WAIT SOME MORE.
0150	0087	002C			
0151	0088	0000	CD7A,	FIM P6 ; 0	/PUT -16 IN R13=BLCTR
0152	0089	0850			
0153	008A	0477	CD7,	JMS ; ADAR	/ADD IN CURRENT AREA.
0154	008B	0850			
0155	008C	019C		JMS ; BUNCH	/GET NEXT BUNCH.
0156	008D	0850			
0157	008E	0400		JMS ; DECID	/SEE WHAT HAPPENED.
0158	008F	00F2		IAC	
0159	0090	0014			
0160	0091	009A		JCN ZA ; CD8	/RETURN TO BASELINE?
0161	0092	00F2		IAC	

RECORD 3

0162	0093	0014	
0163	0094	00C0	JCN ZA ; CD9A /START OF SADDLE?
0164	0095	00F2	IAC
0165	0096	0014	
0166	0097	009A	JCN ZA ; CD8
0167	0098	0840	
0168	0099	0087	JUN ; CD7A /RESET COUNTER
0169	009A	007D	
0170	009B	0089	CD8, ISZ R13 ; CD7 /16TH CONSECUTIVE SMALL
0171	009C	0020	
0172	009D	0006	FIM P0 ; BFPFLG /SLOPE? YES
0173	009E	0021	SRC P0 /BUFFER PEAK?
0174	009F	00E9	RDM
0175	00A0	001C	
0176	00A1	0013	JCN NZA ; CD /YES. IGNORE
0177	00A2	0022	
0178	00A3	0045	FIM P1 ; 105 /PRINT "E". REAL PEAK. RETURN
0179	00A4	0850	
0180	00A5	033F	JMS ; PRCHR /TO BASELINE
0181	00A6	0020	
0182	00A7	0001	FIM P0 ; MCLOCK /SUBTRACT 2 FROM
0183	00A8	0022	
0184	00A9	002C	FIM P1 ; MCSVE /MCLOCK AND STORE
0185	00AA	0024	
0186	00AB	0002	FIM P2 ; 2 /IN MCSVE
0187	00AC	0850	
0188	00AD	047F	JMS ; MCSTR
0189	00AE	0020	
0190	00AF	0005	FIM P0 ; SDLFLG /2ND PART OF SADDLE?
0191	00B0	0021	SRC P0
0192	00B1	00E9	RDM
0193	00B2	001C	
0194	00B3	00C2	JCN NZA ; CD8B /YES
0195	00B4	0020	
0196	00B5	0007	FIM P0 ; BLAFLG /PEAK FOLLOWING BUFFER
0197	00B6	0021	SRC P0 /PEAK?
0198	00B7	00E9	RDM
0199	00B8	001C	
0200	00B9	00C4	JCN NZA ; CD14A /YES
0201	00BA	0850	
0202	00BB	0200	JMS ; DTOT /NO. OUTPUT DATA
0203	00BC	0840	
0204	00BD	0013	JUN ; CD /START OVER
0205	00BE	0840	
0206	00BF	015F	CD12A, JUN ; CD12 /REROUTE JCN'S TO NEXT PROM
0207	00C0	0840	
0208	00C1	010B	CD9A, JUN ; CD9
0209	00C2	0840	
0210	00C3	0100	CD8B, JUN ; CD8A
0211	00C4	0840	
0212	00C5	0176	CD14A, JUN ; CD14
0213			
0214			
0215			/INITIALIZE OUTPUT PORTS.

RECORD 3

0216				
0217	00C6	00DF	CLEAR,	LDM 17
0218	00C7	0020		
0219	00C8	0000	FIM P0:0	/CLEAR CLOCK ON PORT 0.
0220	00C9	0021	SRC P0	
0221	00CA	00E2	WRR	
0222	00CB	00F0	CLB	/STOP TTY ON PORT 1.
0223	00CC	0060	INC R0	
0224	00CD	0021	SRC P0	
0225	00CE	00E2	WRR	
0226	00CF	00C0	BEL 0	
0227				

RECORD 4

```

0001      /AUTOMATIC DIGITAL INTEGRATION FOR MCS4.
0002
0003      *400
0004 0100 0022
0005 0101 0010      CD8A, FIM P1 ; FPBCH2 /USE FPBCH2 AS
0006 0102 0024
0007 0103 0056      FIM P2 ; FBL      /BASELINE
0008 0104 00DA      LDM 12
0009 0105 0850
0010 0106 046D      JMS ; TRANS
0011 0107 0850
0012 0108 0200      JMS ; DTOT      /OUTPUT 2ND PART OF SADDLE.
0013 0109 0840
0014 010A 0013      JLN ; CD
0015 010B 0850
0016 010C 019C      CD9, JMS ; BUNCH
0017 010D 0850
0018 010E 0400      JMS ; DECID      /2 POS. DERIV. IN A
0019 010F 00F2      IAC      /ROW MEAN SADDLE
0020 0110 00F2      IAC
0021 0111 0014
0022 0112 0115      JCN ZA ; CD10      /SADDLE
0023 0113 0840
0024 0114 0087      JLN ; CD7A      /NO SADDLE
0025 0115 0022
0026 0116 0006      CD10, FIM P1 ; BFPFLG /REAL PEAK STARTING
0027 0117 0023      SRC P1      /AFTER BUFFER PEAK?
0028 0118 00E9      RDM
0029 0119 001C
0030 011A 016A      JCN NZA ; CD13      /YES
0031 011B 0022
0032 011C 0053      FIM P1 ; 123      /NO. SADDLE PEAK. PRINT "S".
0033 011D 0850
0034 011E 033F      JMS ; PRCHR
0035 011F 0020
0036 0120 0001      FIM P0; MCLOCK /STORE CURRENT TIME-2
0037 0121 0022
0038 0122 002C      FIM P1 ; MCSVE /AS END OF PEAK
0039 0123 0024
0040 0124 0002      FIM P2; 2
0041 0125 0850
0042 0126 047F      JMS ;MCSTR
0043 0127 0850
0044 0128 0200      JMS ; DTOT      /OUTPUT DATA ON PREVIOUS PK.
0045
0046      /INTEGRATE CURRENT PEAK:
0047      /COMPENSATE FOR THE TIME USED TO PRINT OUT
0048      /THE DATA (APPROX. 5 SEC.) BY ADDING IN EXTRA
0049      /AREA TO APPROXIMATE THE BUNCHES NOT TAKEN.
0050
0051 0129 0020
0052 012A 0001      CD11, FIM P0; MCLOCK /SAVE START TIME OF PEAK.
0053 012B 0022

```

RECORD 4

0054	012C	0008	FIM P1;MCSVB	
0055	012D	0024		
0056	012E	0002	FIM P2; 2	/=TIME OF FFBCH2-2
0057	012F	0850		
0058	0130	047F	JMS ;MCSTR	
0059	0131	0022		
0060	0132	0020	FIM P1; FCBCH	/APPROXIMATE THE MISSING
0061	0133	0024		
0062	0134	0046	FIM P2; FAREA	/INFO. FROM THE KNOWN
0063	0135	00DA	LDM 12	/INFO.
0064	0136	0850		
0065	0137	046D	JMS ;TRANS	
0066	0138	0850		
0067	0139	019C	JMS ;BUNCH	/DON'T WORRY ABOUT SLOPE
0068	013A	0850		
0069	013B	0477	JMS ;ADAR	/CHANGES HERE.
0070	013C	0850		
0071	013D	019C	JMS ;BUNCH	/FCBCH IS NOW LAST BUNCH
0072	013E	0850		
0073	013F	0477	JMS ;ADAR	/TAKEN BEFORE PRINTOUT.
0074	0140	0850		
0075	0141	0477	JMS ;ADAR	/ADD IT IN TWICE.
0076	0142	0850		
0077	0143	019C	JMS ;BUNCH	/NOW FCBCH=1ST BUNCH
0078	0144	0850		
0079	0145	0477	JMS ;ADAR	/TAKEN AFTER PRINTOUT.
0080	0146	0850		
0081	0147	0477	JMS ;ADAR	/ADD IT IN TWICE.
0082	0148	0020		
0083	0149	0036	FIM P0; FTEMP	/SET FTEMP=2, THEN ADD IT
0084	014A	0021	SRC P0	/IN TO MCLOCK TO COMPENSATE
0085	014B	00D2	LDM 2	/FOR MISSING BUNCHES.
0086	014C	00E0	WRM	
0087	014D	0061	INC R1	/ACCESS NEXT 4 BITS.
0088	014E	00DC	LDM 14	/SET REST OF FTEMP TO 0.
0089	014F	0850		
0090	0150	0465	JMS ;ZERO	
0091	0151	0020		
0092	0152	0036	FIM P0; FTEMP	
0093	0153	0022		
0094	0154	0001	FIM P1; MCLOCK	/SET MCLOCK=MCLOCK+2
0095	0155	00DC	LDM 14	
0096	0156	0850		
0097	0157	0190	JMS ;ADDR	
0098	0158	0022		
0099	0159	0005	FIM P1;SDLFLG	/SET SADDLE FLAG=1 (IN SADDLE)
0100	015A	0023	SRC P1	
0101	015B	00D1	LDM 1	
0102	015C	00E0	WRM	
0103	015D	0840		
0104	015E	004B	JUN ;CD4	/COLLECT DATA; WATCH FOR PFAK
0105	015F	0022		
0106	0160	0006	CD12, FIM P1 ; BFPFLG	/PROBABLY BUFFER PEAK

RECORD 4

```

0107 0161 0023 SRC P1 /SET BFPFLG=1
0108 0162 00D1 LDM 1
0109 0163 00E0 WRM
0110 0164 0022
0111 0165 0046 FIM P1; 106 /OUTPUT "F" FOR FALSE PEAK.
0112 0166 0850
0113 0167 033F JMS ; PRCHR
0114 0168 0840
0115 0169 0087 JUN ; CD7A
0116 016A 0022
0117 016B 0006 CD13, FIM P1 ; BFPFLG /BUFFER PEAK OVER
0118 016C 0023 SRC P1 /SET FLAG TO 0
0119 016D 00F0 CLB
0120 016E 00E0 WRM
0121 016F 0022
0122 0170 0007 FIM P1 ; BLAFLG /USE "BASELINE AFTER"
0123 0171 0023 SRC P1 /AS BASELINE BEFORE AND AFTER
0124 0172 00D1 LDM 1 /SET BLAFLG=1
0125 0173 00E0 WRM
0126 0174 0840
0127 0175 002F JUN ; CD3
0128 0176 0022
0129 0177 0020 CD14, FIM P1 ; FCBCH /REAL PEAK FOLLOWING
0130 0178 0024
0131 0179 0056 FIM P2 ; FEL /SPURIOUS PEAK THAT DIDN'T
0132 017A 00DA LDM 12 /TO BASELINE
0133 017B 0850
0134 017C 046D JMS ; TRANS /USE "BASELINE AFTER" AS B.L.
0135 017D 0850
0136 017E 0200 JMS ; DTOT
0137 017F 0840
0138 0180 0013 JUN ; CD
0139
0140
0141 /SEE IF SIZE OF PEAK IS LARGE
0142 /ENOUGH TO BE A LEGITIMATE PEAK.
0143 /CALLING SEQUENCE:
0144 / JMS ; PKSZ
0145 / BEL 1 /PEAK TOO SMALL
0146 / BEL 0 /PEAK OK
0147
0148 0181 0022
0149 0182 0018 PKSZ, FIM P1 ; FPBCH1+2 /THIS CODE MAY
0150 /HAVE TO BE CHANGED WITH
0151 /EXPERIENCE.
0152 0183 0023 SRC P1 /FPBCH1 HAS PEAK VALUE.
0153 0184 00F1 CLC
0154 0185 00D3 LDM 3 /BINARY PT. IS BETWEEN CHARS.
0155 0186 00E8 SBM /2 & 3. IF 16 SAMPLES ARE
0156 0187 001A
0157 0188 018F JCN ZC ; PKSZ1 /TAKEN PER BUNCH, THE SUM
0158 /IS BETWEEN 0 AND 16.
0159 /IF EACH OF THE A-D READINGS IS AT LEAST
0160 /.05 ON THE OPTICAL DENSITY SCALE,

```

RECORD 4

```

0161          /WHICH RUNS FROM 0 TO 2, THEN THE SUM
0162          /WILL BE AT LEAST .4. TEST FOR THIS AND
0163          /RETURN ACCORDINGLY.
0164
0165 0189 0063      INC R3          /GET NEXT HIGHER 4 BITS.
0166 018A 0023      SRC P1
0167 018B 00E9      RDM          /IF HIGH 4 BITS ARE NOT 0,
0168 018C 001C
0169 018D 018F      JCN NZA; PKS Z1 /THEN WE HAVE A PEAK.
0170 018E 00C1      BBL 1        /NO PEAK
0171 018F 00C0      PKS Z1, BBL 0 /PEAK.
0172
0173
0174          /ADDITION ROUTINE. P0 CONTAINS ADDRESS OF NO.
0175          /TO ADD IN. P1 CONTAINS DESTINATION ADDRESS
0176          /AC=-(NO. OF CHARS.=HEX DIGITS).
0177
0178 0190 00B7      ADDR, XCH R7      /SAVE COUNT IN R7.
0179 0191 00F0      CL B
0180 0192 0021      ADDR1, SRC P0
0181 0193 00E9      RDM          /GET NO.
0182 0194 0023      SRC P1
0183 0195 00EB      ADM          /ADD IN.
0184 0196 00E0      WRM          /STORE.
0185 0197 0061      INC R1        /POINT TO NEXT NO.
0186 0198 0063      INC R3
0187 0199 0077
0188 019A 0192      ISZ R7; ADDR1    /GET NEXT SUM.
0189 019B 00C0      BBL 0
0190

```

RECORD 5

```

0001      /AUTOMATIC DIGITAL INTEGRATION FOR MCS4.
0002
0003
0004      /COLLECT DATA FOR FFBCH2 IN BUNCHES OF "NPB". SAMPLE
0005      /A-D WHEN CLOCK FLAG COMES UP. USE FFBCH1, FCBCH,...
0006      /AS PUSHDOWN STACK. CLOCK RUNNING ON ENTRY.
0007
0008      019C      0022
0009      019D      0016      BUNCH,      FIM P1;FPBCH1      /SAVE FPBCH1, FCBCH, ETC.
0010      019E      0024
0011      019F      0010      FIM P2;FPBCH2
0012      01A0      00DA      LDM 12
0013      01A1      0850
0014      01A2      046D      JMS ;TRANS
0015      01A3      0022
0016      01A4      0020      FIM P1;FCBCH      /SAVE FCBCH
0017      01A5      0024
0018      01A6      0016      FIM P2;FPBCH1
0019      01A7      00DA      LDM 12
0020      01A8      0850
0021      01A9      046D      JMS ;TRANS
0022      01AA      0022
0023      01AB      0026      FIM P1;FFBCH1      /SAVE FFBCH1
0024      01AC      0024
0025      01AD      0020      FIM P2;FCBCH
0026      01AE      00DA      LDM 12
0027      01AF      0850
0028      01B0      046D      JMS ;TRANS
0029      01B1      0022
0030      01B2      0030      FIM P1;FFBCH2
0031      01B3      0024
0032      01B4      0026      FIM P2;FFBCH1
0033      01B5      00DA      LDM 12
0034      01B6      0850
0035      01B7      046D      JMS ;TRANS
0036      01B8      0020
0037      01B9      0030      FIM P0;FFBCH2      /SET FFBCH2=0
0038      01BA      00DA      LDM 12
0039      01BB      0850
0040      01BC      0465      JMS ;ZERO
0041      01BD      0020
0042      01BE      0036      FIM P0;FTEMP      /SET FTEMP=0
0043      01BF      00DA      LDM 12
0044      01C0      0850
0045      01C1      0465      JMS ;ZERO
0046      01C2      002E
0047      01C3      0FF0      FIM P7;-NPB      /GET NPB=NO. OF SAMPLES/BUNCH.
0048      01C4      0022
0049      01C5      0030      BCH1,      FIM P1;CLFLG      /CLOCK FLAG.
0050      01C6      0023      SRC P1      /WAIT FOR FLAG, CLEAR
0051      01C7      00EA      BCH1A,      RDR      /WITH CONVERT COMMAND.
0052      01C8      00F6      RAR
0053      01C9      0000      NOP

```


RECORD 5

```

0054 01CA 0012
0055 01CB 01C7          JCN NZC;BCH1A
0056
0057          /SAMPLE A-D AND STORE 10 BIT RESULT IN LSB 12
0058          /BITS OF FTEMP. LSB 2 BITS OF FTEMP ARE UNUSED.
0059          /0=BASELINE; 1777=HIGHEST POSSIBLE PEAK.
0060
0061 01CC 0022
0062 01CD 0000          FIM P1;0          /START A-D CONVERSION
0063 01CE 0023          SRC P1          /WITH CONVERT COMMAND
0064 01CF 00DC          BCH1B, LDM 14
0065 01D0 00E2          WRR          /AND CLEAR CLOCK
0066 01D1 00DF          LDM 17
0067 01D2 00E2          WRR
0068 01D3 0024
0069 01D4 0036          FIM P2;FTEMP          /READ A-D AND STORE IN FTEMP.
0070 01D5 0022
0071 01D6 0060          FIM P1;AD1          /PLS=60 H, LSB 2 BITS
0072 01D7 0850
0073 01D8 01F5          JMS ;RDSTR
0074 01D9 0022
0075 01DA 0050          FIM P1;AD2          /PLS=50 H, MIDDLE 4 BITS
0076 01DB 0850
0077 01DC 01F5          JMS ;RDSTR
0078 01DD 0022
0079 01DE 0040          FIM P1;AD3          /PLS=40 H, MSB 4 BITS
0080 01DF 0850
0081 01E0 01F5          JMS ;RDSTR
0082 01E1 0020
0083 01E2 0036          FIM P0;FTEMP          /ADD IN A-D READING
0084 01E3 0022
0085 01E4 0030          FIM P1;FFBCH2
0086 01E5 00DA          LDM 12
0087 01E6 0850
0088 01E7 0190          JMS ;ADDR
0089 01E8 007F
0090 01E9 01C4          ISZ R15;BCH1          /GET ANOTHER SAMPLE.
0091
0092          /ALL SAMPLES COLLECTED FOR THIS BUNCH. UPDATE
0093          /MASTER CLOCK.
0094 01EA 0022
0095 01EB 0001          FIM P1;MCLOCK
0096 01EC 00F0          CLB
0097 01ED 00D1          LDM 1          /ADD 1 TO LOW 4 BITS.
0098 01EE 0023          BCH3, SRC P1
0099 01EF 00EB          ADM
0100 01F0 00E0          WRM
0101 01F1 0063          INC R3          /GET NEXT 4 BITS AND ADD IN
0102 01F2 0012
0103 01F3 01EE          JCN NZC;BCH3          /CARRY IF NECESSARY. ASSUME NO
0104 01F4 00C0          BBL 0          /OVERFLOW FROM MSB 4 BITS!!!
0105
0106
0107          /GET A-D BITS POINTED TO BY P1 AND STORE

```

RECORD 5

0108			/IN CHAR. POINTED TO BY P2. INCR. P2 FOR	
0109			/NEXT TIME AROUND.	
0110				
0111	01F5	0023	RDSTR,	SRC P1
0112	01F6	00EA	RDR	/READ BITS.
0113	01F7	0025	SRC P2	
0114	01F8	00F4	OMA	
0115	01F9	00E0	WRM	/STORE
0116	01FA	0065	INC R5	
0117	01FB	00C0	BBL 0	
0118				

RECORD 6

```

0001      /AUTOMATIC DIGITAL INTEGRATION FOR MCS4.
0002
0003      /MISC. ROUTINES.
0004
0005      *1000
0006
0007      /OUTPUT DATA TO TTY. MCSVB HAS START TIME OF PEAK;
0008      /MCSVE HAS STOP TIME; FBL HAS BEFORE AND AFTER
0009      /BASELINE; MCSVM HAS MIDPOINT TIME OF PEAK.
0010      /FAREA HAS AREA OF PEAK.
0011
0012 0200 0850
0013 0201 0336      DTO T,      JMS ;CRLF      /OUTPUT CR-LF
0014 0202 0022
0015 0203 0000      FIM P1; PKNO      /OUTPUT PEAK NO. TO TTY.
0016 0204 0023      SRC P1
0017 0205 00E9      RDM
0018 0206 00F2      IAC
0019 0207 00E0      WRM
0020 0208 0020
0021 0209 0036      FIM P0;FTEMP      /STORE CHAR. IN FTEMP FOR PRNU
                                M.
0022 020A 0021      SRC P0
0023 020B 00E0      WRM
0024 020C 0061      INC R1      /ZERO OUT REST OF FTEMP.
0025 020D 00DD      LDM 15
0026 020E 0850
0027 020F 0465      JMS ;ZERO
0028 0210 0850
0029 0211 037B      JMS ;PRNLM
0030 0212 00DC      LDM 14      /PRINT SOME SPACES.
0031 0213 0850
0032 0214 049D      JMS ;PRSPC
0033
0034      /FSPI=(SECONDS/BLNCH)/60. SO MCSVM*FSPI=
0035      /MINUTES SINCE BEGINNING OF EXPERIMENT.
0036      /FORM PRODUCT AND OUTPUT.
0037
0038 0215 0020
0039 0216 0036      FIM P0;FTEMP      /PUT MCSVM IN FTEMP BECAUSE
0040 0217 00DA      LDM 12      /FTEMP AND FTEMP1 HAVE 6
0041 0218 0850
0042 0219 0465      JMS ;ZERO      /HEX CHARS.
0043 021A 0022
0044 021B 000C      FIM P1;MCSVM
0045 021C 0024
0046 021D 0036      FIM P2;FTEMP
0047 021E 00DC      LDM 14      /USE LSB 4 CHARS.
0048 021F 0850
0049 0220 046D      JMS ;TRANS
0050 0221 00DE      LDM FSPI1      /GET LSB 4 BITS IN 2'S COM-
0051 0222 00B5      XCH R5      /PLEMENT FORM.
0052 0223 00D6      LDM FSPI2      /MIDDLE 4 BITS OF FRACTIONAL

```

RECORD 6

```

0053 0224 00B4      XCH R4          /PART. MSB 4 BITS=1111, SO
0054 0225 0850
0055 0226 02A8      JMS ;MILLAD      /THEY JUST SHIFT HEX POINT.
0056
0057                /HEX POINT NOW BETWEEN FTEMP1+2 AND FTEMP1+3.
0058 0227 0022
0059 0228 0043      FIM P1;FTEMP1+3 /STORE INTEGER PART OF PRODUCT
0060 0229 0024
0061 022A 0036      FIM P2;FTEMP      /IN FTEMP.
0062 022B 00DD      LDM 15
0063 022C 0850
0064 022D 046D      JMS ;TRANS
0065 022E 0850
0066 022F 037B      JMS ;PRNUM
0067
0068                /NOW OUTPUT FRACTIONAL PART TO 1 PLACE.
0069 0230 0020
0070 0231 0043      FIM P0;FTEMP1+3 /SET REST OF FTEMP1 TO 0.
0071 0232 00DD      LDM 15
0072 0233 0850
0073 0234 0465      JMS ;ZERO
0074 0235 0850
0075 0236 04A5      JMS ;MUL10      /ADD FTEMP1 TO ITSELF 10 TIMES
0076 0237 0022
0077 0238 002E      FIM P1;56      /OUTPUT "."
0078 0239 0850
0079 023A 033F      JMS ;PRCHR
0080 023B 0022
0081 023C 0043      FIM P1;FTEMP1+3 /GET 1ST DEC. PLACE.
0082 023D 0023      SRC P1
0083 023E 00E9      RDM
0084 023F 0022
0085 0240 0030      FIM P1;60
0086 0241 00B3      XCH R3          /CONVERT TO ASCII
0087 0242 0850
0088 0243 033F      JMS ;PRCHR
0089 0244 00DC      LDM 14          /PRINT SPACES
0090 0245 0850
0091 0246 049D      JMS ;PRSPC
0092 0247 0020
0093 0248 0036      DTO T5, FIM P0;FTEMP /COMPUTE STOP TIME-START
0094 0249 00DA      LDM 12          /TIME. STORE RESULT IN FTEMP
0095 024A 0850
0096 024B 0465      JMS ;ZERO
0097 024C 0022
0098 024D 0008      FIM P1;MCSVB      /STORE MCSVB
0099 024E 0024
0100 024F 0036      FIM P2;FTEMP      /IN FTEMP
0101 0250 00DC      LDM 14
0102 0251 0850
0103 0252 046D      JMS ;TRANS
0104 0253 0022
0105 0254 0036      FIM P1;FTEMP      /PUT -MCSVB IN FTEMP.
0106 0255 00DA      LDM 12

```

RECORD 6

0107	0256	0850		
0108	0257	0492	JMS ; COMP	
0109	0258	0020		
0110	0259	002C	FIM P0; MCSVE	/NOW SUBTRACT MCSVB
0111	025A	0022		
0112	025B	0036	FIM P1; FTEMP	
0113	025C	00DA	LDM 12	
0114	025D	0850		
0115	025E	0190	JMS ; ADDR	
0116			/NOW MULTIPLY TIME OF PEAK BY BASELINE TO FIND	
0117			/AREA TO SUBTRACT FROM FAREA.	
0118	025F	0022		
0119	0260	0058	FIM P1; FEL+2	
0120	0261	0023	SRC P1	
0121	0262	00F0	CLB	
0122	0263	00E8	SBM	/GET - FEL
0123	0264	00B5	XCH R5	/LOW ORDER 4 BITS.
0124	0265	0063	INC R3	/GET HIGH ORDER 4 BITS.
0125	0266	0023	SRC P1	
0126	0267	00F3	CMC	
0127	0268	00D0	LDM 0	
0128	0269	00E8	SBM	
0129	026A	00B4	XCH R4	/SAVE IN R4.
0130	026B	0850		
0131	026C	02A8	JMS ; MLAD	/FORM PRODUCT AND STORE IN FTE
			MP1	
0132				

RECORD 7

```

0001      /AUTOMATIC DIGITAL INTEGRATION FOR MCS4.
0002
0003      /DATA OUTPUT. CONTINUED.
0004
0005 026D 0022
0006 026E 0040      FIM P1:FTEMP1      /SET FTEMP1=-FTEMP1
0007 026F 00DA      LDM 12
0008 0270 0850
0009 0271 0492      JMS ;COMP
0010 0272 0022
0011 0273 0048      FIM P1:FAREA+2      /AND FIND FAREA-AREA UNDER
0012 0274 0020
0013 0275 0040      FIM P0:FTEMP1      /BASELINE. NOTE THAT BINARY
0014 0276 00DC      LDM 14      /POINTS ARE ALIGNED, SINCE BIN.
0015 0277 0850
0016 0278 0190      JMS ;ADDR      /POINT OF FAREA IS BETWEEN
0017                                /FAREA+2 AND FAREA+3.
0018 0279 0024
0019 027A 0036      DTO T6, FIM P2:FTEMP
0020 027B 0022
0021 027C 0049      FIM P1:FAREA+3
0022 027D 00DD      LDM 15
0023 027E 0850
0024 027F 046D      JMS ;TRANS
0025 0280 00F0      CLB
0026 0281 0850
0027 0282 037B      JMS ;PRNLM      /OUTPUT AREA
0028
0029      /MULTIPLY FBL BY 10 AND THEN BY 10 AGAIN.
0030      /PICK OFF THE DECIMAL CHARS. GENERATED AND
0031      /OUTPUT.
0032
0033 0283 0020
0034 0284 0040      FIM P0:FTEMP1      /SET FTEMP1=0
0035 0285 00DA      LDM 12
0036 0286 0850
0037 0287 0465      JMS ;ZERO
0038 0288 0022
0039 0289 0058      FIM P1:FBL+2      /FBL+2 AND FBL+3 CONTAIN MSB
0040 028A 0024
0041 028B 0040      FIM P2:FTEMP1      /8 BITS OF FRACTIONAL PART OF
0042 028C 00DE      LDM 16      /AVERAGE
0043 028D 0850
0044 028E 046D      JMS ;TRANS      /BASELINE VALUE.
0045 028F 00DC      LDM 14
0046 0290 0850
0047 0291 049D      JMS ;PRSPC      /SPACE OVER.
0048 0292 0022
0049 0293 002E      FIM P1:56      /". "
0050 0294 0850
0051 0295 033F      JMS ;PRCHR
0052 0296 00DE      LDM 16      /GO THROUGH LOOP TWICE
0053 0297 00BF      XCH R15

```

RECORD 7

```

0054 0298 0850
0055 0299 04A5 DTOT6A, JMS ;MUL10 /MULTIPLY BY 10
0056 029A 0020
0057 029B 0042 FIM P0;FTEMP1+2 /GET WHAT'S TO LEFT OF "."
0058 029C 0021 SRC P0
0059 029D 00E9 RDM
0060 029E 0022
0061 029F 0030 FIM P1;60 /CONVERT TO ASCII
0062 02A0 00B3 XCH R3
0063 02A1 00F0 CLB /SET THIS CHAR. TO 0 SO THAT
0064 02A2 00E0 WRM /NEXT OUTPUT IS NOT AFFECTED
0065 02A3 0850
0066 02A4 033F JMS ;PRCHR /OUTPUT
0067 02A5 007F
0068 02A6 0298 ISZ R15;DTOT6A
0069 02A7 00C0 BBL 0 /ALL DONE.
0070
0071 /MULTIPLY FTEMP BY NO. IN P2, WHICH IS IN 2'S
0072 /COMPLEMENT FORM. R5=LSB 4 BITS; R4=MSB 4 BITS.
0073 /STORE RESULT IN FTEMP1. FIND RESULT BY ADDING.
0074
0075 02A8 0020
0076 02A9 0040 MULAD, FIM P0;FTEMP1 /STORE RESULT HERE.
0077 02AA 00DA LDM 12
0078 02AB 0850
0079 02AC 0465 JMS ;ZERO
0080 02AD 00A5 LD R5 /MAKE SURE P2#0.
0081 02AE 001C
0082 02AF 02B4 JCN NZA; MULAD1
0083 02B0 00A4 LD R4
0084 02B1 001C
0085 02B2 02B4 JCN NZA; MULAD1
0086 02B3 00C0 BBL 0 /0*FTEMP=0. RETURN.
0087 02B4 0020
0088 02B5 0036 MULAD1, FIM P0;FTEMP /ADD UP FTEMP NO. OF TIMES
0089 02B6 0022
0090 02B7 0040 FIM P1;FTEMP1 /INDICATED BY P2.
0091 02B8 00DA LDM 12
0092 02B9 0850
0093 02BA 0190 JMS ;ADDR
0094 02BB 0075
0095 02BC 02B4 ISZ R5;MULAD1
0096 02BD 0074
0097 02BE 02B4 ISZ R4;MULAD1
0098 02BF 00C0 BBL 0 /ALL DONE.
0099

```

```

0001      /AUTOMATIC DIGITAL INTEGRATION FOR MCS4.
0002
0003      /MISC. ROUTINES.
0004
0005      *1400
0006
0007      /PRINT MESSAGE ON TTY. P0 POINTS TO START OF
0008      /MESSAGE. 1 CHAR. PER LOCATION. -1 ENDS THE
0009      /STRING.
0010
0011      0300      0020
0012      0301      0310      PRMSG,      FIM P0;MHDR
0013      0302      0032      PRMS2,      FIN P1      /GET CHAR.
0014      0303      0073
0015      0304      0308      ISZ R3;PRMS1      /TEST FOR - 1
0016      0305      0072
0017      0306      0308      ISZ R2;PRMS1
0018      0307      00C0      BBL 0      /END OF STRING
0019      0308      0032      PRMS1,      FIN P1      /RESTORE CHAR.
0020      0309      0850
0021      030A      033F      JMS ;PRCHR
0022      030B      0071
0023      030C      0302      ISZ R1;PRMS2      /GET NEXT CHAR.
0024      030D      0060      INC R0
0025      030E      0840
0026      030F      0302      JLN ;PRMS2
0027      0310      0000
0028      0311      0000
0029      0312      000D
0030      0313      000D      MHDR,      00;00;15;15      /"NO. TIME AREA B.L."
0031      0314      000D
0032      0315      000A
0033      0316      000A
0034      0317      000A
0035      0318      0020
0036      0319      0020      15;12;12;12;40;40
0037      031A      004E
0038      031B      004F
0039      031C      002E
0040      031D      0020
0041      031E      0020
0042      031F      0020
0043      0320      0020
0044      0321      0020
0045      0322      0054
0046      0323      0049
0047      0324      004D
0048      0325      0045      116;117;56;40;40;40;40;40;124;111;115;105
0049      0326      0020
0050      0327      0020
0051      0328      0020
0052      0329      0020
0053      032A      0041

```


RECORD 8

0054	032B	0052		
0055	032C	0045		
0056	032D	0041		
0057	032E	0020		
0058			40; 40; 40; 40; 101; 122; 105; 101; 40;	
0059	032F	0020		
0060	0330	0020		
0061	0331	0042		
0062	0332	002E		
0063	0333	004C		
0064	0334	002E	40; 40; 102; 56; 114; 56	
0065	0335	0FFF	- 1	
0066				
0067			/PRINT CR-LF ON TTY.	
0068				
0069	0336	0022		
0070	0337	000D	CRLF,	FIM P1;15
0071	0338	0850		
0072	0339	033F	JMS ;PRCHR	
0073	033A	0022		
0074	033B	000A	FIM P1;12	
0075	033C	0850		
0076	033D	033F	JMS ;PRCHR	
0077	033E	00C0	BBL 0	
0078				
0079			/PRINT CHAR IN P1 ON TTY.	
0080				
0081	033F	0024		
0082	0340	0010	PRCHR,	FIM P2;20 /PORT 1. TTY OUTPUT.
0083	0341	0025		SRC P2
0084	0342	00E2		WRR /START BIT.
0085	0343	00D8		LDM 10
0086	0344	00B6		XCH R6 /R6=BIT COUNTER.
0087	0345	00F0	TO 1,	CLB
0088	0346	0840		
0089	0347	0362		JUN ;SBR2
0090	0348	00F1	TO 1A,	CLC
0091	0349	00B2		XCH R2 /SHIFT 8 BITS RIGHT
0092	034A	00F6		RAR
0093	034B	00B2		XCH R2
0094	034C	00B3		XCH R3
0095	034D	00F6		RAR
0096	034E	00B3		XCH R3
0097	034F	00F7		TCC /CARRY TO ACC
0098	0350	00E2		WRR /ASCII BIT.
0099	0351	0076		
0100	0352	0345		ISZ R6;TO1 /GET REST OF BITS
0101	0353	00D1		LDM 1 /FLAG FOR WHERE TO RETURN.
0102	0354	0840		
0103	0355	0362		JUN ;SBR2
0104	0356	00D1	TO 1B,	LDM 1 /STOP BIT 1
0105	0357	00E2		WRR /STOP BIT.
0106	0358	00D2		LDM 2 /FLAG.
0107	0359	0840		

RECORD 8

0108	035A	0362	JUN ; SBR2	
0109	035B	00D3	TO 1C,	LDM 3
0110	035C	0840		
0111	035D	0362	JUN; SBR2	
0112	035E	00D4	TO 1E,	LDM 4
0113	035F	0840		
0114	0360	0362	JUN; SBR2	
0115	0361	00C0	TO 1D,	BEL 0
0116	0362	0024		
0117	0363	003C	SBR2,	FIM P2; 74 /9.09 MS DELAY
0118	0364	0075		
0119	0365	0364	L2,	ISZ R5; L2
0120	0366	0074		
0121	0367	0364		ISZ R4; L2
0122	0368	0024		
0123	0369	003C	SBR1,	FIM P2; 74
0124	036A	0075		
0125	036B	036A	L1,	ISZ R5; L1
0126	036C	0074		
0127	036D	036A		ISZ R4; L1
0128	036E	0014		
0129	036F	0348	JCN ZA; TO 1A	/AC IS FLAG. USE IT TO RETURN.
0130	0370	00F8	DAC	
0131	0371	0014		
0132	0372	0356	JCN ZA; TO 1B	
0133	0373	00F8	DAC	
0134	0374	0014		
0135	0375	035B	JCN ZA; TO 1C	
0136	0376	00F8	DAC	
0137	0377	0014		
0138	0378	035E	JCN ZA; TO 1E	
0139	0379	0840		
0140	037A	0361	JUN ; TO 1D	
0141				
0142				
0143			/PRINT NO. IN FTEMP TO FTEMP+2. OUTPUT 4 CHARS.:	
0144			/EITHER LEADING SPACES OR DIGITS. NOTE THAT THE INPUT	
0145			/MUST OCCUPY 12 BITS AND BE < 4000 AND THAT	
0146			/THE INPUT NO. IS DESTROYED.	
0147				
0148	037B	00D6	PRNUM,	LDM 6 /GO THROUGH BCD LOOP 10 TIMES
0149	037C	00BE		XCH R14
0150	037D	00DE		LDM 16 /AFTER 1ST 2 CARRIES ARE IN,
0151	037E	00BD		XCH R13 /OUTPUT LEADING CHAR.
0152	037F	00F0		CLB
0153	0380	00BC		XCH R12 /CARRIES STORED HERE.
0154	0381	00F0		CLB /USE R11 AS FLAG: 0=NO DIGITS
0155	0382	00BB		XCH R11 /OUTPUT YET. #0 MEANS A DIGIT
0156				/HAS BEEN OUTPUT.
0157	0383	0020		
0158	0384	03D8	FIM P0; TBL	/SET UP ACCESS TO TABLE OF
0159			/CONSTANTS FOR BCD CONVERSION. GET THE 3 HEX CONSTANTS	
0160			/REPRESENTING ONE OF THE VALUES 2000, 1000, 800, ...	
0161			/10. SUBTRACT FROM THE NO. POINTED TO IN P1 AND	

RECORD 8

```

0162      /SAVE THE RESULT IN FMUL1.
0163 0385 00F0 PRNM1,  CLB
0164 0386 0026
0165 0387 0060      FIM P3;FMUL1      /SAVE RESULT HERE.
0166 0388 0022
0167 0389 0036      FIM P1;FTEMP
0168 038A 0034      FIN P2      /LOW ORDER 4 BITS IN R5
0169 038B 0023      SRC P1      /MIDDLE 4 IN R4
0170 038C 00E9      RDM
0171 038D 0095      SUB R5
0172 038E 00F3      OMC      /SET CARRY FOR CORRECT BORROWI
                                NG
0173 038F 0027      SRC P3      /SAVE RESULT
0174 0390 00E0      WRM
0175 0391 0063      INC R3      /ACCESS NEXT CHAR.
0176 0392 0067      INC R7      /STORE IN NEXT PLACE.
0177 0393 0023      SRC P1      /GET NEXT 4 BITS OF NO.
0178 0394 00E9      RDM
0179 0395 0094      SUB R4
0180 0396 00F3      OMC
0181 0397 0027      SRC P3
0182 0398 00E0      WRM
0183 0399 0071
0184 039A 039C      ISZ R1; PRNM2      /GET HIGH 4 BITS FROM TBLF.
0185 039B 0060      INC R0
0186 039C 0034 PRNM2,  FIN P2      /R5=HIGH 4 BITS
0187 039D 0063      INC R3      /GET NEXT NO.
0188 039E 0067      INC R7      /STORE IN NEXT PLACE.
0189 039F 0023      SRC P1
0190 03A0 00E9      RDM
0191 03A1 0095      SUB R5
0192 03A2 0027      SRC P3
0193 03A3 00E0      WRM
0194 03A4 00AC      LD R12      /GET CARRIES STORED SO FAR.
0195 03A5 00F5      RAL
0196 03A6 00BC      XCH R12
0197 03A7 00AC      LD R12
0198 03A8 00F6      RAR      /PUT CARRY BACK IN TO TEST IT.
0199 03A9 001A
0200 03AA 03B2      JCN ZC;PRNM3      /IF CARRY=0, THEN THIS CONSTAN
                                T
0201 03AB 0022
0202 03AC 0060      FIM P1;FMUL1      /IS TOO LARGE.
0203 03AD 0024
0204 03AE 0036      FIM P2;FTEMP      /CONSTANT<INPUT.
0205 03AF 00DD      LDM 15      /REPLACE INPUT WITH INPUT-
0206 03B0 0850
0207 03B1 046D      JMS ;TRANS      /CONSTANT.
0208 03B2 007D
0209 03B3 03C9 PRNM3,  ISZ R13;PRNM5A      /OUTPUT NOT READY.
0210 03B4 00AC      LD R12      /OUTPUT NO. IN R12
0211 03B5 001C
0212 03B6 03BF      JCN NZA;PRNM4      /DON'T OUTPUT LEADING ZEROES.
0213 03B7 00AB      LD R11

```

RECORD 8

0214	03B8	001C			
0215	03B9	03BE		JCN NZA; PRNM3A	
0216	03BA	0022			
0217	03BB	0020		FIM P1; 40	/LEADING 0. OUTPUT SPACE
0218	03BC	0840			
0219	03BD	03C3		JUN ; PRNM5	/INSTEAD.
0220	03BE	00F0	PRNM3A,	CLB	
0221	03BF	0022			
0222	03C0	0030	PRNM4,	FIM P1; 60	
0223	03C1	00B3		XCH R3	
0224	03C2	006B		INC R11	/SET FLAG.
0225	03C3	0850			
0226	03C4	033F	PRNM5,	JMS ; PRCHR	
0227	03C5	00F0		CLB	
0228	03C6	00BC		XCH R12	/SET R12 TO 0 AGAIN.
0229	03C7	00DC		LDM 14	/RESET COUNTER TO 4 BITS.
0230	03C8	00BD		XCH R13	
0231	03C9	0071			
0232	03CA	03CC	PRNM5A,	ISZ R1; PRNM6	/UPDATE POINTER TO CONSTANTS
0233	03CB	0060		INC R0	
0234	03CC	007E			
0235	03CD	0385	PRNM6,	ISZ R14; PRNM1	/GET ALL 10.
0236					
0237					/LAST DIGIT TO OUTPUT IS WHAT REMAINS IN LSB
0238					/4 BITS IN FTEMP.
0239					
0240	03CE	0024			
0241	03CF	0036		FIM P2; FTEMP	
0242	03D0	0025		SRC P2	
0243	03D1	00E9		RDM	
0244	03D2	0022			
0245	03D3	0030		FIM P1; 60	/CONVERT TO ASCII
0246	03D4	00B3		XCH R3	
0247	03D5	0850			
0248	03D6	033F		JMS ; PRCHR	
0249	03D7	00C0		BBL 0	
0250					
0251	03D8	00D0	TBLE,	320	/2000. LSB 8 BITS FIRST, THEN
0252	03D9	0007		7	/MSB 4 BITS. = 7D0H
0253	03DA	00E8		350	/1000= 3E8H
0254	03DB	0003		3	
0255	03DC	0020		40	/800= 320H
0256	03DD	0003		3	
0257	03DE	0090		220	/400= 190H
0258	03DF	0001		1	
0259	03E0	00C8			
0260	03E1	0000		310; 0	/200= C8H
0261	03E2	0064			
0262	03E3	0000		144; 0	/100= 64H
0263	03E4	0050			
0264	03E5	0000		120; 0	/80= 50H
					0265 03E6 0028
0266	03E7	0000		50; 0	/40= 28H
0267	03E8	0014			

RECORD 8

0268	03E9	0000	24:0	/20=14H
0269	03EA	000A		
0270	03EB	0000	12:0	/10
0271				

RECORD 9

```

0001      /AUTOMATIC DIGITAL INTEGRATION FOR MCS4.
0002
0003      /MISC. ROUTINES.
0004
0005      *2000
0006
0007      /EVALUATE 1ST DERIVATIVE OF THIS SIGNAL AT FCBCH,
0008      /F'(FCBCH) BY THE SAVITZKY-GOLAY LEAST SQUARES
0009      /ALGORITHM FOR 5 POINTS AND RETURN AS FOLLOWS:
0010      /
0011      /BBL 17: 0<F'(FCBCH)<FMPS=MINIMUM SLOPE FOR START OF PE
                AK
0012      /BBL 16: F'>=FMPS
0013      /BBL 15: -FMPS<F'<0: SIGNAL PROBABLY NEAR
0014      /                BASELINE OR PEAK.
0015      /BBL 14: F'<-FMPS: SIGNAL PROBABLY DESCENDING AFTER PE
                AK
0016      /
0017      /CONVOLUTE THE SEQUENCE FFBCH2,FFBCH1,FCBCH,FPBCH1,
0018      /FPBCH2, WITH THE SEQUENCE 2,1,0,-1,-2.
0019
0020      0400      0022
0021      0401      0010      DECID,      FIM P1;FPBCH2      /TRANSFER FPBCH2 TO FTEMP
0022      0402      0024
0023      0403      0036      FIM P2;FTEMP
0024      0404      00DA      LDM 12
0025      0405      0850
0026      0406      046D      JMS ;TRANS
0027      0407      0022
0028      0408      0036      FIM P1;FTEMP      /NEGATE FTEMP
0029      0409      00DA      LDM 12
0030      040A      0850
0031      040B      0492      JMS ;COMP
0032      040C      0020
0033      040D      0030      FIM P0;FFBCH2      /FIND FFBCH2-FPBCH2
0034      040E      0022
0035      040F      0036      FIM P1;FTEMP
0036      0410      00DA      LDM 12
0037      0411      0850
0038      0412      0190      JMS ;ADDR
0039      0413      0020
0040      0414      0036      FIM P0;FTEMP      /DOUBLE RESULT
0041      0415      0022
0042      0416      0036      FIM P1;FTEMP
0043      0417      00DA      LDM 12
0044      0418      0850
0045      0419      0190      JMS ;ADDR
0046      041A      0020
0047      041B      0026      FIM P0;FFBCH1      /ADD IN FFBCH1
0048      041C      0022
0049      041D      0036      FIM P1;FTEMP
0050      041E      00DA      LDM 12
0051      041F      0850

```

RECORD 9

```

0052 0420 0190      JMS ;ADDR
0053 0421 0022
0054 0422 0016      FIM P1;FPBCH1
0055 0423 0024
0056 0424 0040      FIM P2;FTEMP1 /AND SUBTRACT FPBCH1.
0057 0425 00DA      LDM 12
0058 0426 0850
0059 0427 046D      JMS ;TRANS
0060 0428 0022
0061 0429 0040      FIM P1;FTEMP1
0062 042A 00DA      LDM 12
0063 042B 0850
0064 042C 0492      JMS ;COMP
0065 042D 0020
0066 042E 0040      FIM P0;FTEMP1
0067 042F 0022
0068 0430 0036      FIM P1;FTEMP
0069 0431 00DA      LDM 12
0070 0432 0850
0071 0433 0190      JMS ;ADDR /SLOPE NOW IN FTEMP.
0072 0434 0022
0073 0435 0038      FIM P1;FTEMP+5 /SEE IF SLOPE>0
0074 0436 0023      SRC P1
0075 0437 00E9      RDM
0076 0438 00F5      RAL
0077 0439 0012
0078 043A 0441      JCN NZC;DCD2 /<0
0079 043B 0850
0080 043C 044C      JMS ;TEST />0. SEE IF LARGE OR SMALL.
0081 043D 0014
0082 043E 0440      JCN ZA;DCD1
0083 043F 00CF      BEL 17 /SMALL POS. SLOPE.
0084 0440 00CE      DCD1, BEL 16 /LARGE POS. SLOPE
0085 0441 0022
0086 0442 0036      DCD2, FIM P1;FTEMP
0087 0443 00DA      LDM 12
0088 0444 0850
0089 0445 0492      JMS ;COMP /NEGATE SLOPE, THEN
0090 0446 0850
0091 0447 044C      JMS ;TEST /TEST FOR LARGE OR
0092 0448 001C
0093 0449 0448      JCN NZA;DCD3 /SMALL VALUE
0094 044A 00CC      BEL 14 /LARGE NEG. SLOPE
0095 044B 00CD      DCD3, BEL 15 /SMALL NEG. SLOPE.
0096
0097
0098 /ROUTINE TO TEST IF SLOPE IN FTEMP>SOME MINIMUM.
0099 /RETURN AS FOLLOWS:
0100 / BEL 17 /SMALL SLOPE
0101 / BEL 0 /LARGE SLOPE
0102
0103 044C 0022
0104 044D 0037      TEST, FIM P1;FTEMP+1 /THESE INSTRUCTIONS MAY HAVE T

```

0

0105			/BE CHANGED WITH EXPERIENCE. ASSUME FOR NOW
0106			/THAT A SLOPE>.0000 1000(2) INDICATES ONSET OF
			A
0107			/PEAK. THE BINARY POINT OF FTEMP IS LOCATED
0108			/BETWEEN CHARS. 2 AND 3.
0109	044E	0023	SRC P1 /LOOK AT MIDDLE 4 BITS OF
0110	044F	00F1	CLC /OF FRACTIONAL PART.
0111	0450	00D8	LDM 10 /=.0000 1000
0112	0451	00E8	SBM
0113	0452	001A	
0114	0453	0464	JCN ZC; TEST1 /BORROW GENERATED.
0115	0454	0063	INC R3 /NO BORROW. CHECK HIGHER ORDER
0116	0455	0023	SRC P1 /BITS TO SEE IF THEY ARE NON-0
0117	0456	00E9	RDM
0118	0457	001C	
0119	0458	0464	JCN NZA; TEST1 /LARGE SLOPE
0120	0459	0063	INC R3 /GET FTEMP+3
0121	045A	0023	SRC P1
0122	045B	00E9	RDM
0123	045C	001C	
0124	045D	0464	JCN NZA; TEST1
0125	045E	0063	INC R3 /GET HIGH 4 BITS OF SLOPE.
0126	045F	0023	SRC P1
0127	0460	00E9	RDM
0128	0461	001C	
0129	0462	0464	JCN NZA; TEST1
0130	0463	00CF	BBL 17 /SMALL POS. SLOPE.
0131	0464	00C0	TEST1, BBL 0 /LARGE POS. SLOPE.
0132			
0133			
0134			/SET N CHARS. TO 0. AC=-N ON ENTRY. ADDRESS OF
0135			/LEAST SIG. CHAR. IN P0 ON ENTRY.
0136			
0137	0465	00B2	ZERO, XCH R2 /KEEP COUNT HERE.
0138	0466	00F0	CLB
0139	0467	0021	SRC P0
0140	0468	00E0	WRM /SET CHAR TO 0.
0141	0469	0061	INC R1 /SET UP NEXT ONE.
0142	046A	0072	
0143	046B	0467	ISZ R2; ZERO+2
0144	046C	00C0	BBL 0 /ALL DONE.
0145			
0146			
0147			/TRANSFER N CHARS. FROM ADDRESS POINTED TO BY
0148			/P1 TO ADDRESS POINTED TO BY P2. -N IN AC
0149			/ON ENTRY.
0150			
0151	046D	00B6	TRANS, XCH R6 /SAVE COUNT HERE.
0152	046E	0023	SRC P1
0153	046F	00E9	RDM /GET CHAR.
0154	0470	0025	SRC P2
0155	0471	00E0	WRM /STORE HERE.
0156	0472	0063	INC R3 /GO TO NEXT CHAR.
0157	0473	0065	INC R5

RECORD 9

0158	0474	0076	
0159	0475	046E	ISZ R6; TRANS+1
0160	0476	00C0	BEL 0
0161			
0162			/ADD IN FCBCH TO FAREA.
0163			
0164	0477	0020	
0165	0478	0020	ADAR, FIM P0; FCBCH
0166	0479	0022	
0167	047A	0046	FIM P1; FAREA
0168	047B	00DA	LDM 12
0169	047C	0850	
0170	047D	0190	JMS ; ADDR
0171	047E	00C0	BEL 0
0172			

```

0001      /AUTOMATIC DIGITAL INTEGRATION FOR MCS4.
0002
0003      /MISC. ROUTINES.
0004
0005      /STORE MCLOCK-N IN CHARS. POINTED TO BY P1.
0006      /ADDRESS OF MCLOCK IN P0 ON ENTRY. R4=0, R5=N ON
0007      /ENTRY.
0008
0009      047F 00F1 MCSTR,  OLC      /SUBTRACT NO. IN R5 FROM LOW
0010      0480 00DD      LDM 15    /4 BITS OF MCLOCK. PUT -3
0011      0481 00B6      XCH R6    /IN R6.
0012      0482 0021      SRC P0
0013      0483 00E9      RDM
0014      0484 0095      SUB R5    /SUBTRACT N.
0015      0485 0023      SRC P1
0016      0486 00E0      WRM      /WRITE LOW 4 BITS.
0017      0487 0061 MCST1, INC R1
0018      0488 0063      INC R3
0019      0489 0021      SRC P0    /GET NEXT 4 BITS.
0020      048A 00E9      RDM
0021      048B 00F3      OMC
0022      048C 0094      SUB R4    /R4=0, SO JUST BORROW IS TAKEN
0023      048D 0023      SRC P1    /INTO ACCOUNT.
0024      048E 00E0      WRM
0025      048F 0076
0026      0490 0487      ISZ R6;MCST1 /GET ALL CHARS.
0027      0491 00C0      BBL 0
0028
0029
0030      /COMPLEMENT NO. P1 CONTAINS ADDRESS OF NO.
0031      /TO NEGATE. AC=-NO. OF CHARS.
0032
0033      0492 00B4 COMP,  XCH R4    /STORE COUNT HERE.
0034      0493 00FA      STC
0035      0494 0023 COMP1, SRC P1    /GET 4 BITS.
0036      0495 00D0      LDM 0
0037      0496 00F3      OMC
0038      0497 00E8      SBM
0039      0498 00E0      WRM      /STORE BACK
0040      0499 0063      INC R3    /GO TO NEXT CHAR.
0041      049A 0074
0042      049B 0494      ISZ R4;COMP1
0043      049C 00C0      BBL 0
0044
0045
0046      /PRINT N SPACES. ENTER WITH -N IN AC.
0047
0048      049D 00B0 PRSPC, XCH R0    /SAVE COUNT HERE.
0049      049E 0022
0050      049F 0020      FIM P1;40
0051      04A0 0850
0052      04A1 033F      JMS ;PRCHR
0053      04A2 0070

```

RECORD 10

```

0054 04A3 049E      ISZ R0;PRSPC+1
0055 04A4 00C0      BBL 0
0056
0057
0058                /MULTIPLY FTEMP1 BY 10 AND STORE IN FTEMP1.
0059                /FTEMP IS USED ALSO.
0060
0061 04A5 0020
0062 04A6 0036      MLL 10,   FIM P0;FTEMP      /SET FTEMP=0.
0063 04A7 00DA      LDM 12
0064 04A8 0850
0065 04A9 0465      JMS ;ZERO
0066 04AA 00D6      LDM 6          /- 10
0067 04AB 00B5      XCH R5
0068 04AC 0020
0069 04AD 0040      MLL P,   FIM P0;FTEMP1
0070 04AE 0022
0071 04AF 0036      FIM P1;FTEMP
0072 04B0 00DA      LDM 12
0073 04B1 0850
0074 04B2 0190      JMS ;ADDR
0075 04B3 0075
0076 04B4 04AC      ISZ R5;MLLP
0077 04B5 0022
0078 04B6 0036      FIM P1;FTEMP      /PUT PRODUCT IN FTEMP1.
0079 04B7 0024
0080 04B8 0040      FIM P2;FTEMP1
0081 04B9 00DA      LDM 12
0082 04BA 0850
0083 04BB 046D      JMS ;TRANS
0084 04BC 00C0      BBL 0

```

RECORD 11

0001	ADAR	0477
0002	ADDR	0190
0003	ADDR1	0192
0004	AD1	0060
0005	AD2	0050
0006	AD3	0040
0007	BCH1	01C4
0008	BCH1A	01C7
0009	BCH1B	01CF
0010	BCH3	01EE
0011	BELL	0007
0012	BFPFLG	0006
0013	BLAFLG	0007
0014	BUNCH	019C
0015	CD	0013
0016	CD1	0018
0017	CD10	0115
0018	CD11	0129
0019	CD12	015F
0020	CD12A	00BE
0021	CD13	016A
0022	CD14	0176
0023	CD14A	00C4
0024	CD2	0020
0025	CD3	002F
0026	CD4	004B
0027	CD5	0057
0028	CD5A	005B
0029	CD6	0067
0030	CD6B	006F
0031	CD6C	0079
0032	CD7	0089
0033	CD7A	0087
0034	CD8	009A
0035	CD8A	0100
0036	CD8B	00C2
0037	CD9	010B
0038	CD9A	00C0
0039	CL EAR	00C6
0040	CLFLG	0030
0041	COMP	0492
0042	COMP1	0494
0043	CRLF	0336
0044	DCD1	0440
0045	DCD2	0441
0046	DCD3	044B
0047	DECID	0400
0048	DTOT	0200
0049	DTOTS	0247
0050	DTOT6	0279
0051	DTOT6A	0298
0052	FAREA	0046
0053	FEL	0056

RECORD 11

0054	FCBCH	0020
0055	FFBCH1	0026
0056	FFBCH2	0030
0057	FMUL1	0060
0058	FPBCH1	0016
0059	FPBCH2	0010
0060	FSPI1	000E
0061	FSPI2	0006
0062	FTEMP	0036
0063	FTEMP1	0040
0064	L1	036A
0065	L2	0364
0066	MCLOCK	0001
0067	MCSTR	047F
0068	MCST1	0487
0069	MCSVB	0008
0070	MCSVE	002C
0071	MCSVM	000C
0072	MHDR	0310
0073	MLLP	04AC
0074	MULAD	02A8
0075	MULAD1	02B4
0076	MUL10	04A5
0077	NPB	0010
0078	NZA	000C
0079	NZC	0002
0080	PKNO	0000
0081	PKSZ	0181
0082	PKSZ1	018F
0083	PRCHR	033F
0084	PRMSG	0300
0085	PRMS1	0308
0086	PRMS2	0302
0087	PRNM1	0385
0088	PRNM2	039C
0089	PRNM3	03B2
0090	PRNM3A	03BE
0091	PRNM4	03BF
0092	PRNM5	03C3
0093	PRNM5A	03C9
0094	PRNM6	03CC
0095	PRNLM	037B
0096	PRSPC	049D
0097	P0	0000
0098	P1	0002
0099	P2	0004
0100	P3	0006
0101	P4	0008
0102	P5	000A
0103	P6	000C
0104	P7	000E
0105	RDSTR	01F5
0106	R0	0000
0107	R1	0001

RECORD 11

0108	R10	000A
0109	R11	000B
0110	R12	000C
0111	R13	000D
0112	R14	000E
0113	R15	000F
0114	R2	0002
0115	R3	0003
0116	R4	0004
0117	R5	0005
0118	R6	0006
0119	R7	0007
0120	R8	0008
0121	R9	0009
0122	SBR1	0368
0123	SBR2	0362
0124	SDL FLG	0005
0125	START	0000
0126	TEL E	03D8
0127	TEST	044C
0128	TEST1	0464
0129	TO 1	0345
0130	TO 1A	0348
0131	TO 1B	0356
0132	TO 1C	035B
0133	TO 1D	0361
0134	TO 1E	035E
0135	TRANS	046D
0136	TSNZ	0009
0137	TSZ	0001
0138	ZA	0004
0139	ZC	000A
0140	ZERO	0465

SAMPLE ØUTPUT FROM "MADI"

NØ.	TIME	AREA	B.L.BMS
1	22.2	65	.00ME
2	31.7	49	.01BME
3	40.7	12	.01BME
4	44.6	7	.02BME
5	52.9	289	.01

NØ.	TIME	AREA	B.L.BMS
1	22.8	66	.00ME
2	31.8	55	.01BME
3	40.8	8	.02BME
4	44.6	15	.01BME
5	49.3	342	.01



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-24☒ 4004 ☐ 8008 ☐ 8080 ☐ 4040

(use additional sheets if necessary)

Program
Title

MULTIPLY/DIVIDE 8 DECIMAL NUMBERS

Function

All numbers occupy 1/2 A RAM, i.e., characters 0-7 or 8-15 with the low order digit in character 0 or 8. Numbers can be positive or negative. Negative numbers are in complementary form (ten's complement) a number with a value 6 in the high order position is assumed negative. (See additional sheets for function.)

Required
Hardware

No additional support.

Required
Software

Subroutine to change an 8 digit number to its complement has been included and can be called prior to entry and upon return to change the sign.

Input
ParametersMultiplication A x B:

B=Multiplier 8 digits
A=Multiplicand 8 digits
AC=Round off place
[Dec. digits in multiplier
+D.D in multiplicand -D.D
in product (16 digit internal
desired.)] Registers RPA and
RPB point to A and B respec-
tively.

Division B/A:

A=Divisor 8 digits
B=Dividend 8 digits
AC=Decimal places in offset
[D.P quotient=D.P. dividend
-D.P. divisor +offset.]
CY=1 to round quotient.
Registers RPA and RPB point to
A and B respectively.

Output
Results

Product replaces multiplicand
A. Multiplier B unchanged.

Remainder replaces Divisor A
Quotient replaces Dividend B
AC=15 on overflow (Div. by 0).
0 otherwise.

Registers Modified: All but register
SRS (equated 15)

Maximum Subroutine Nesting Level:
0

RAM Required: RAM0 and RAM1 are
used for scratch

Assembler/Compiler Used:
Intel 4004 MACRO Assembler

ROM Required:
512 Words of ROM memory (2 ROMS)

Programmer:
Hannah Fox

Company: Caltech
1201 E. California, Pasadena, CA
Astro Electronics 91125

354-33

might mean RAM 0, Reg 0 & RAM 0, Reg 1

FUNCTION:

The multiply/divide subroutine calculates the product/quotient by repeated shifted additions or subtractions and by incrementing/decrementing the multiplier/quotient. (See flowcharts.)

To achieve maximum speed and efficiency, the following features have been incorporated.

- 1.) If the active digit K in the multiplier is 6, the multiplicand is added $(10-k)$ times to the product (rather than subtracted k times).
- 2.) The incrementing/decrementing is continued for only as long as the status of the carry dictates.
- 3.) The decision for continuing the sign-extended addition/subtraction process is also dependent on the carry (see flowchart).
- 4.) Both negative (10's complement) and positive numbers are acceptable.

DEFINITIONS:

SRX, ~~SRY~~, SRZ, SRS:

Registers within CPU

PO:

Register pair 0

R0, R1:

Registers 0 and 1 contained in PO

RPA, RPB: { location of multiplicand
is coded in RPA
location of multiplier
is coded in RPB

CPU register pairs used to point to a RAM

RPAE, RPAØ, RPBE, RPBØ:

Even and odd registers contained in RPA and RPB

RSA, RSAE, RSAØ, RSB, RSBE, RSBØ: Same as above, but used as pointers to scratch RAM registers

RTP:

CPU register pair set to an address for JIN return

RTPE, RTPØ:

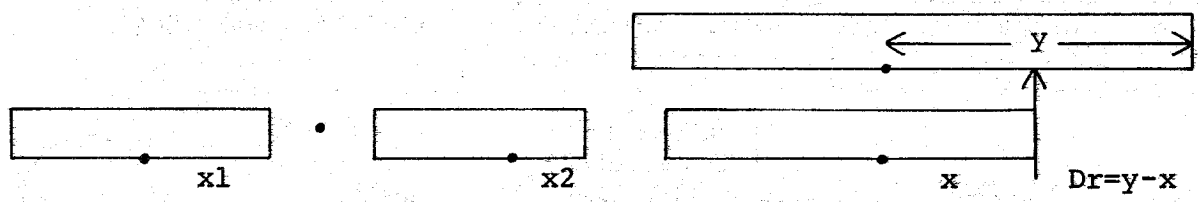
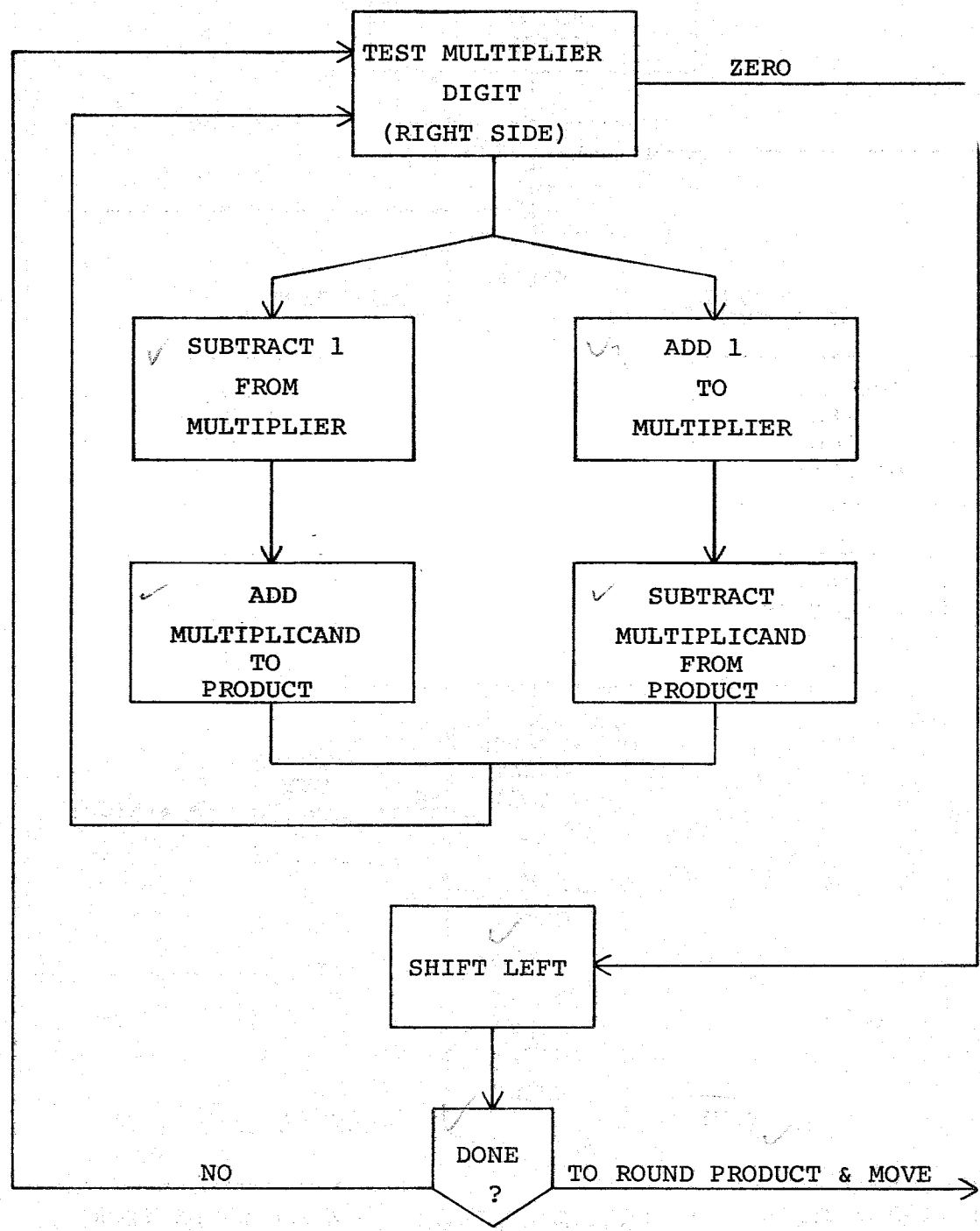
Registers contained in RTP, RTPE contains most significant byte of address RTPØ least sign

RAM RPA, RAM RPB, etc.:

RAM register whose address is in CPU register pairs RPA, RPB, etc.

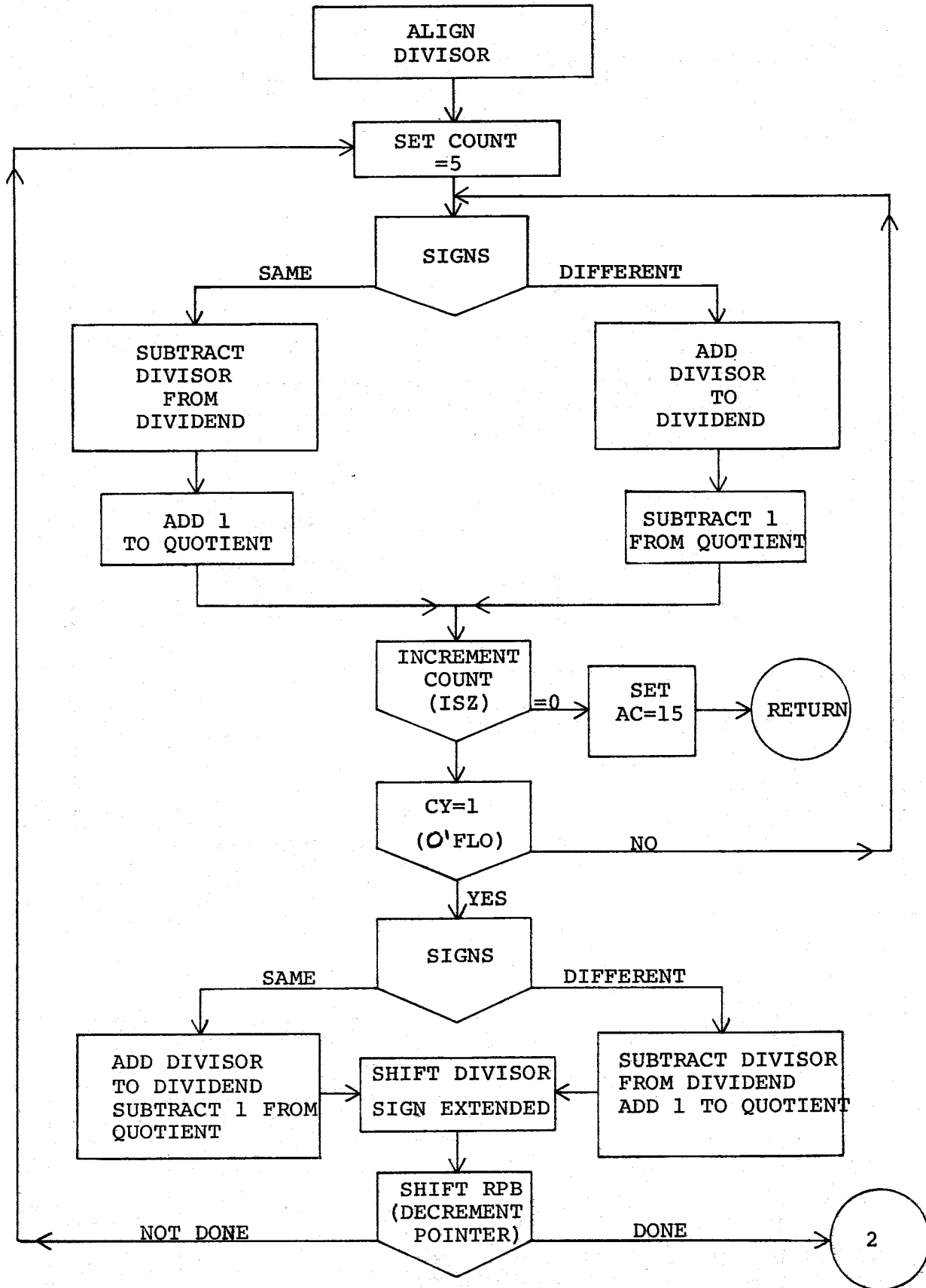
NOTE: Descriptions and comments sometimes refer to a RAM register as RAM N0 or RAM N8 or simply N0 or N8, i.e., RAM register Hex N (N=0 to F) beginning at character 0 or 8. Similarly RAM RPA, RAM RPB or RPA, RPA+8, RPB or RPB=8.

MULTIPLICATION

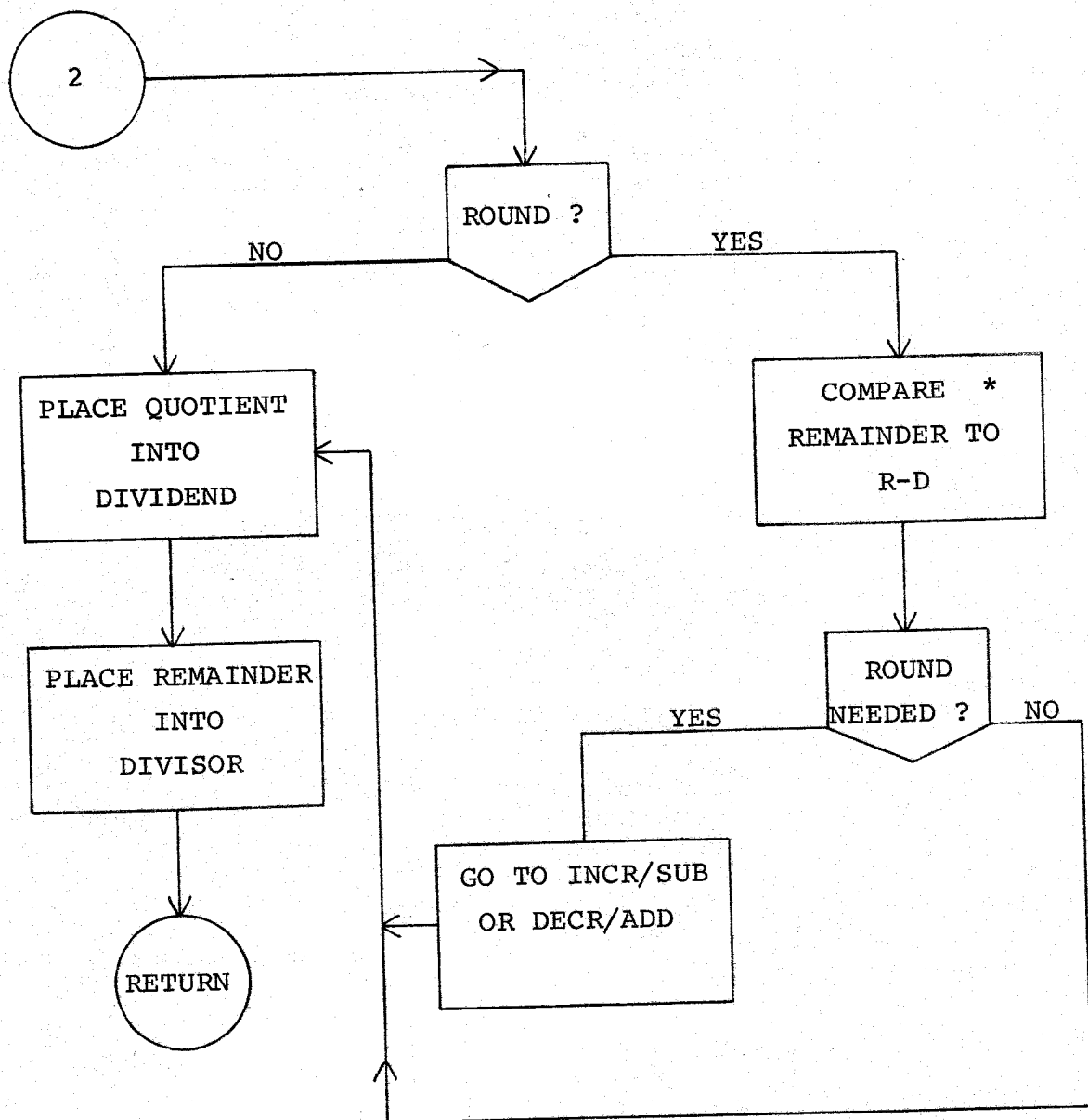


x1 Decimal places to right of digit x1
 x2 Decimal places to right of digit x2
 IN product $y = x1 + x2$ digits are present
 Only x (offset) digits are desired. Thus round at $y - x$.

DIVISION

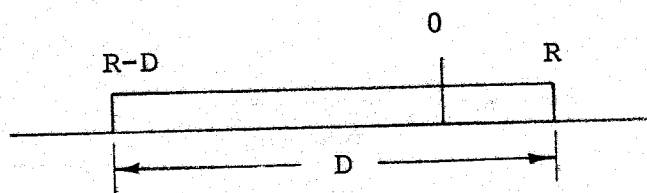


DIVISION (CONTINUED)



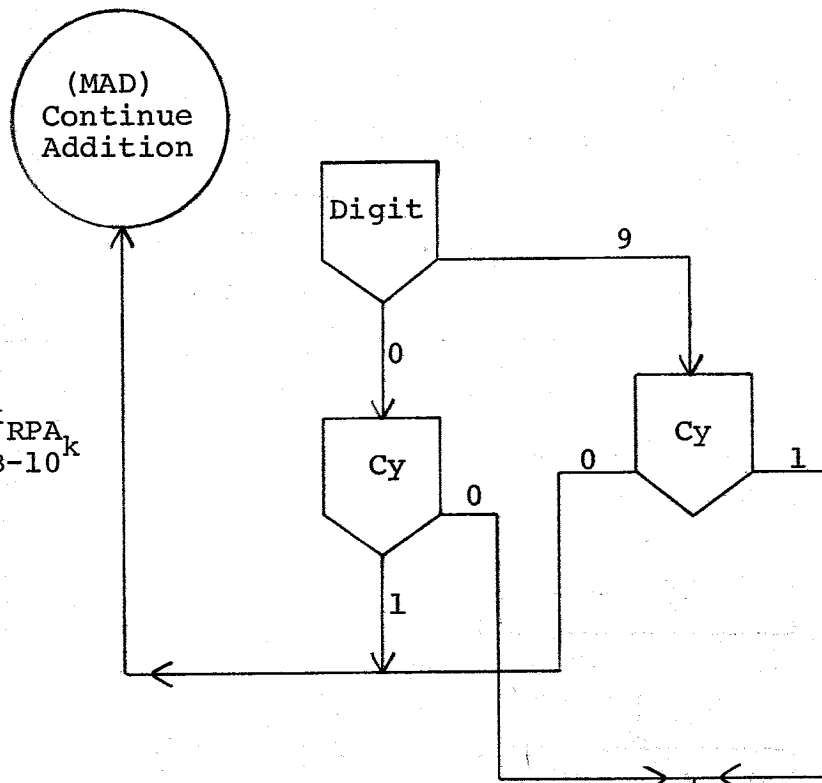
* Take $R-D+R=2R-D$
compare signs of
 $R-D, 2R-D, R$

Round if $\text{sign}(2R-D) = \text{Sign}(R) \neq \text{Sign}(R)$
No round if $\text{sign}(2R-D) = \text{Sign}(R-D) \neq \text{sign}(R)$

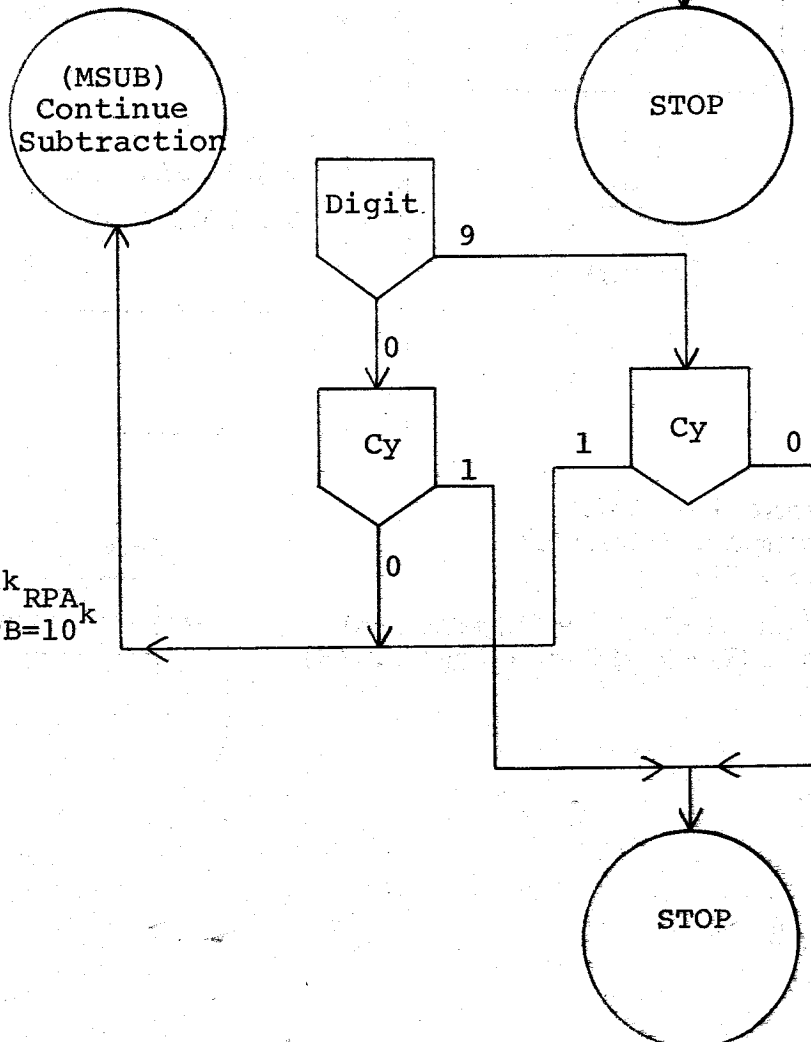


DECISION FOR SIGN EXTENDED ADD/SUBTRACT

MAD: $A = A + 10^k \text{RPA}$
 $\text{RPB} = \text{RPB} - 10^k$



MSUB: $A = A - 10^k \text{RPA}$
 $\text{RPB} = \text{RPB} + 10^k$



RAM 32101598
123 456
low low

0003		ORG 3	assembly starts at location 003
0002	RPA	EQU 2	reg pair 2 ← multiplicand stored at location ended here
0003	RPA0	EQU 3	reg 3
0002	RPAE	EQU 2	reg 2
0004	RPB	EQU 4	reg pair 4 ← multiplier stored at location ended here
0005	RPB0	EQU 5	reg 5
0004	RPBE	EQU 4	reg 4
0006	RSA	EQU 6	reg pair 6
0007	RSA0	EQU 7	reg 7
0006	RSAE	EQU 6	reg 6
0008	RSB	EQU 8	reg pair 8
0009	RSB0	EQU 9	reg 9
0008	RSBE	EQU 8	reg 8
000A	RTP	EQU 10	reg pair 10
000B	RTP0	EQU 11	reg 11
000A	RTP1	EQU 10	reg 10
0000	P0	EQU 0	reg pair 0
0000	R0	EQU 0	reg 0
0001	R1	EQU 1	reg 1
000C	SRX	EQU 12	
000D	SRX	EQU 13	
000E	SRX	EQU 14	
000F	SRX	EQU 15	

} registers within chip ?

Assume
Jump To MULT from
main Program

; MULTIPLICATION (A+B TO A)
; PARMS - RPA = MULTIPLICAND, REPLACED BY PRODUCT
; RPB = MULTIPLIER, SCRATCH REGISTER PAIR(00,10)
; AE CONTAINS ROUND OFF PLACE
; CY = 1 ON OVERFLOW
; RPA0 = RPA0 + 8, RPB0 = RPB0 + 8 AT EXIT

0003	2600	MULT:	FIM RSA,00	Fin 6,00
0005	27		SRC RSA	src 6
0006	E7		WR3	; SAVE FOR ROUNDING
0007	E0		CLB	
0008	27	MULA:	SRC RSA	clear scratch reg 00
0009	E0		WRM	; CLEAR PRODUCT AREA
000A	7708		ISZ RSA0,MULA	when was airt counter set up ?
000C	2618		FIM RSA,18H	0001 1000 chip Reg 1 char B
000E	25	MULB:	SRC RPB	
000F	E9		RDM	45600 ; GET MULTIPLIER B ← INPUT NO.
0010	27		SRC RSA	
0011	E0		WRM	; PUT IN SCRATCH SPACE RAM 18
0012	65		INC RPB0	
0013	770E		ISZ RSA0,MULB	
0015	A5		LD RPB0	; SAVE RPB0+8
0016	E4		WR0	save location
0017	A4		LD RPBE	or multiplier in status char 0,1 of scratch reg
0018	E5		WR1	
0019	2413		FIM RPB,18H	
001B	25	MULC:	SRC RPB	; NEXT MULTIPLIER DIGIT
001C	E9		RDM	4050601 0005

405 06 01 006

400

0010 1430 J# MULD

4004 MACRO ASSEMBLER, VER 2.2 TEST PROGRAM ERRORS = 0 PAGE 2

001F DB LDM 11
 0020 80 XCH SRY ; TO COMPUTE COUNT
 0021 05 LDM 5
 0022 F1 CLC
 0023 E3 SRM ; TEST SIZE 5+16-D OR 16+(5-DIGIT)
 0024 F3 CMC
 0025 1A2E JNC MLOD ; DIGIT LTE 5
 0027 F4 CMA ; DIGIT - 6 = 15 - (16+(5-D))
 0028 80 ADD SRY ; DIGIT + 15 - 9 = 12 - 6 + 0
 0029 80 XCH SRY
 002A 2A01 FIM RTP,01H ; RETURN SIGNAL
 002C 4388 X MSRT1: JUN MSRT1
 002E 80 MLOD: ADD SRY
 002F 80 XCH SRY
 0030 2A00 FIM RTP,00 ; RETURN SIGNAL
 0032 4065 V MART1: JUN MAD
 0034 7020 X MSRT1: ISZ SRY,MSRT1
 0036 401R JUN MULC
 0038 7032 V MART1: ISZ SRY,MART1
 003A 401R JUN MULC
 003C 751R V MULD: ISZ RPB0,MULC
 003E F0 CLB
 003F 36 XCH RSAE
 0040 27 SRC RSA
 0041 EF RD3 ; GET ROUND OFF PLACE
 0042 F8 UAC
 0043 1A51 JNC MULE ; NO ROUNDING
 0045 B7 XCH RSAO
 0046 F1 CLC
 0047 D5 LDM 5 ; ROUND OFF
 0048 27 X MULE: SRC RSA
 0049 ER ADM
 004A FB UAA
 004B E0 WRM
 004C 1A51 JNC MULE
 004E D0 LDM 0
 004F 7748 ISZ RSAO,MULE
 0051 EF MULE: RD3
 0052 B7 XCH RSAO
 0053 D3 LDM 8
 0054 BC XCH SRX ; SET COUNT
 0055 27 MULE: SRC RSA ; MOVE PRODUCT TO RPA
 0056 E9 RDM
 0057 23 SRC RPA

This the one I changed

DONE

ROUND OFF

0058	E0	WRM		
0059	67	INC R5A0	inc 7	6,7
005A	63	INC RPA0	inc 3	2,3
005B	7C55	ISZ SRX,MULG		
005D	2610	FIM RSA,10H	Fin 6, chro 0 Reg 1 char 0	
005F	27	SRC RSA		; EXIT WITH RPA0+8, RPBO+8
0060	EC	RDE		

4004 MACRO ASSEMBLER, VER 2.2 TEST PROGRAM ERRORS = 0 PAGE 3

0061	B5		XCH RPBO XCH 5	
0062	E0		R01	
0063	B4		XCH RPBE XCH 4	
0064	C0		BRL 0	Jump back to MAIN
0065	A4	X MAC:	LD RPBE	; COPY RPB TO R5A FOR
0066	B6		XCH R5AE	; DECREMENT LOOP
0067	A5		LD RPBO	
0068	B7		XCH R5A0	
0069	F1		CLC	
006A	F9	DCLA:	ICS	; DECREMENT MULTIPLIER
006B	27		SRC RSA	
006C	EB		ADM	
006D	FB		UAA	
006E	E1		WRM	
006F	1277		JC DCIB	; STOP IF CARRY
0070	776A		ISZ R5A0,DCLA	
0071	F7	DCIB:	CLC	
0072	B6		XCH R5AE	
0073	08		LDM 0	
0074	3C		XCH SRX	
0075	03		LDM 3	
0076	B5		ADD RPBO	; POINT TO ACTIVE DIGIT OF PRODUCT
0077	B7		XCH R5A0	
0078	F1		CLC	
0079	23	DCIB:	SRC RPA	; ADD MULTIPLICAND TO PRODUCT
007A	29		RDM	
007B	27		SRC RSA	
007C	E3		ADM	
007D	FB		UAA	
007E	E0		WRM	
007F	23		SRC RPA	
0080	03		INC RPA0	
0081	07		INC R5A0	
0082	7C70		ISZ SRX,DCLB	
0083	41C0		JUN DCLJ	(MAD CONT.) → MART (038)
0084	A4	X MAD:	LD RPBE	; COPY RPB TO R5A FOR
0085	B6		XCH R5AE	; INCREMENT LOOP
0086	A5		LD RPBO	

0088	B7	XCH	RSAC	
008C	FA	STC		
008D	27	XICLA:	SRC RSA	
008E	D0	LDM	0	; INCREMENT 0 IN RSA
008F	EB	ADM		
0090	FB	DAA		
0091	E0	WRM		
0092	1A95	JNC	ICIB	; GO IF NO CARRY
0094	7780	ISZ	RSAC,ICLA	
0096	F0	XICIB:	CLB	
0097	36	XCH	RSAC	
0098	D8	LDM	8	
0099	BC	XCH	SRX	

2nd NO INPUT

our prog puts divisor in first

12PB
RPA

DIVIDEND
DIVISOR

= QUOTIENT
replaces dividend in RPB
1st NO INPUT

4004 MACRO ASSEMBLER, VER 2.2 TEST PROGRAM ERRORS = 0 PAGE 4

009A	D8	LDM	8	
009B	85	ADD	RPB0	
009C	B7	XCH	RSAC	; SET PRODUCT PLANE
009D	FA	STC		; TURN ON CY
009E	23	XICLB:	SRC RPA	
009F	F9	ICS		
00A0	E8	SBM		; SUBTRACT MULTIPLICAND (DIVISOR)
00A1	27	SRC	RSA	
00A2	F1	CLC		
00A3	EB	ADM		; FROM PRODUCT (DIVIDEND)
00A4	FB	DAA		12
00A5	E0	WRM		6
00A6	23	SRC	RPA	
00A7	63	INC	RPA0	
00A8	67	INC	RSAC	
00A9	7C9F	ISZ	SRX,ICLB	
00AB	417A	JUN	ICLB	

TALES IN
DIVISOR

we are rdy
nos on acc
here

47A

(MSUB CONT)
MSRT 034

JUN 500 4500

; DIVISION (R/A, A = REMAINDER B = QUOTIENT)
; PARMS - RPA = DIVISOR, REPLACED BY REMAINDER
; RPB = DIVIDEND, REPLACED BY QUOTIENT
; AC CONTAINS DECIMAL OFFSET
; CY = 1 TO ROUND QUOTIENT
; D.P.GUD = D.P.DVD - D.P.DVS + OFFSET AC = 15 FOR O.F.
; RPA0 = RPA0 + 8, RPB0 = RPB0 + 8 AT EXIT

00AD	2600	DIV:	FIM RSA;02	
00AF	27	SRC	RSA	
00B0	E6	WR2		; SAVE DISPLACEMENT 0000 ON ACC.
00B1	F7	ICC		
00B2	E7	WR3		; SAVE ROUNDING 0001
00B3	D0	LDM	0	
00B4	27	DLA:	SRC RSA	
00B5	E0	WRM		; CLEAR DIVIDEND

```
00B6 77B4      (ISZ RSAO,DLA)
00B8 EE        RQ2
00B9 B7        XCH RSAO } counter for loops
00BA 08        LDM 8
00BB BC        XCH SRX
00BC A4        LD RPBE      ; SAVE FOR QUOTIENT STORAGE
00BD E4        WR0
00BE A5        LD RPBO      ; This is copying char of 0011
00BF E5        WR1
00C0 25        DLR: SRC RPB  ; WORKS
00C1 E9        RDM          ; COPY THE DIVIDEND
00C2 27        SRC RSA      ; WITH OFFSET 1000
00C3 E0        WRM
00C4 65        INC RPBO
00C5 77C9      (ISZ RSAO,DLK)
00C7 4179      JUN DOVF     ; WORKS
00C9 7CC0      DLK: ISZ SRX,DLB ; WORKS
00CB F1        CLC
00CC BD        XCH SRY
```

Takes dividend

works

it goes here

4004 MACRO ASSEMBLER, VER 2.2 TEST PROGRAM ERRORS = 0 PAGE 5

```
00CD D9        LDM 9
00CE 8D        ADD SRY      ; LAST DIGIT
00CF 12D4      JC DLL       ; > OR = 6 if zero no jump
00D1 F0        CLB         if 0
00D2 40D5      JUN DLC
00D4 D9        > DLL: LDM 9
00D5 27        > DLC: SRC RSA
00D6 E0        WRM          ; SIGN EXTEND
00D7 77D5      ISZ RSAO,DLC
00D9 BC        XCH SRX
00DA DA        LDM 10
00DB F1        CLC
00DC 8C        ADD SRX
00DD F7        ICC          ; CY = 1 IF MSD GTE 6 (B)
00DE BC        XCH SRX      ; DIVIDEND SIGN
00DF D7        LDM 7
00E0 83        ADD RPAO
00E1 B3        XCH RPAO
00E2 B0        XCH SRY      ; SAVE OLD RPAO
00E3 F1        CLC
00E4 23        SRC RPA
00E5 DA        LDM 10
00E6 EB        ADM
00E7 F7        ICC          ; CY = 1 IF MSD GTE 6 (A)
00E8 8C        ADD SRX
00E9 F6        RAR
```

Take it out

00EA	F7		ICC	
00EB	B0		XCH SRY	; QUOTIENT SIGN
00EC	B3		XCH RPA0	; RESTORE DIVISOR
00ED	2418		FIM RPB;18H	
00EF	F0		CLB	- clears reg RPB used for answer
00F0	25	DLD:	SRC RPB	; CLEAR QUOTIENT
00F1	E0		WRM	
00F2	75F0		ISZ RPB0,DLD	
00F4	241F		FIM RPB;1FH	; SET ACTIVE DIGIT
00F6	4100		JUN DPD	
0100			ORG 100H	
0100	2A0B	DPD:	FIM RTP;DLF AND 0FFH	← we are loading a register with 0B
0102	D5		LDM 5	
0103	BE		XCH SRZ	
0104	AD	DPE:	LD SRY	; SIGN OF QUOTIENT: 1=-,0=+
0105	1409		JZ BTABS	
0107	4065	BTABA:	JUN MAD	; ADD IF QUOTIENT < 0
0109	4088	BTABS:	JUN MSUB	; SUBTRACT IF QUOTIENT > 0
010B	1211	DLF:	JC DLG	; DETECT OVERDRAFT
010D	7E04		ISZ SRZ,DPE	; MAIN DIVISION LOOP
010F	4179		JUN DOVF	; DIV. BY 0?
0111	D7	DLG:	LDM 7	
0112	F1		CLC	
0113	85		ADD RPB0	; TEST FOR END
0114	1A22		JNC DPG	; GO IF END

didn't work

Trid JVN didn't work

not what we thought purposes of it

4004 MACRO ASSEMBLER, VER 2.2 TEST PROGRAM ERRORS = 0 PAGE 6

0116	2A1D		FIM RTP;DPE AND 0FFH
0118	AD	DQF:	LD SRY
0119	1407		JZ BTABA
011B	4088		JUN MSUB
011D	A5	DPF:	LD RPB0
011E	F8		DAC
011F	B5		XCH RPB0
0120	4100		JUN DPD
0122	2600	DPG:	FIM RSA;00
0124	27		SRC RSA
0125	2A2F		FIM RTP;DLH AND 0FFH
0127	EC	DQG:	R00 ; SET RSB TO DIVIDEND REGISTER
0128	B8		XCH RSBF
0129	ED		R01
012A	B9		XCH RSB0
012B	D8		LDM 0
012C	BC		XCH SRX
012D	3B		JIN RTP
012E	27	DLH:	SRC RSA
012F	E9		RDM

~~EIN (not JIN)~~

? thought we only needed (B)
RAM 0,1

0130 29 SRC RSB ; SAVE ENDING REMAINDER-
0131 E0 WRM ; DIVISOR WITH OVERDRAFT
0132 67 INC RSAO
0133 69 INC RSBO
0134 7C2F ISZ SRX,DLH
0136 2A3A FIM RTP,DPH AND 0FFH ; SET CLOSE RETURN
0138 4118 JUN DGF ; GO FIX REMAINDER, REMOVE OVERDRAFT
013A 2600 DPH: FIM RSA,00
013C 27 SRC RSA
013D EF RD3
013E 145F JZ DNR ; TEST FOR ROUNDING REQUEST
0140 F1 CLC
0141 2A45 FIM RTP,DLI AND 0FFH
0143 4127 JUN DGF
0145 29 DLI: SRC RSB
0146 E9 RDM
0147 27 SRC RSA
0148 EB ADM ; SUM R + (R-D)
0149 FB DAA
014A 69 INC RSBO
014B 67 INC RSAO
014C 7C45 ISZ SRX,DLI
014E BE XCH SRZ ; SAVE DIGIT OF SUM
014F F0 CLB
0150 DA LDM 10
0151 BE XCH SRZ ; SAVED DIGIT TO AC, A TO SRZ
0152 8E ADD SRZ
0153 F7 ICC ; SAVE SIGN = 1 IF MSD > 5
0154 BE XCH SRZ ; GET A AGAIN, SIGN TO SRZ
0155 EB ADM ; STILL POINTS TO LAST DIGIT
0156 F7 ICC ; SIGN OF R-D

replaced by Quotient
RPB (B) DIVIDEND = QUOTIENT
RPA (A) DIVISOR
replaced by remainder

4004 MACRO ASSEMBLER, VER 2.2 TEST PROGRAM ERRORS = 0 PAGE 7

0157 8E ADD SRZ
0158 F6 RAR ; CY = 0 IF SIGNS SAME
0159 125F JC DNR
015B 2A5F FIM RTP,DNR AND 0FFH
015D 4104 JUN DPE ; TO INCR/DECR SUB
015F 2A63 DNR: FIM RTP,DPJ AND 0FFH
0161 4127 JUN DGF
0163 2418 DPJ: FIM RPB,18H
0165 2607 FIM RSA,00
0167 25 DLJ: SRC RPB
0168 E9 RDM ; GET QUOTIENT
0169 29 SRC RSB RPB
016A E0 WRM ; TO RESULT
016B 27 SRC RSA ; GET REMAINDER

RAM loaded by reg pair 4 (RPB)

2	016C	E9	RDM	
	016D	23	SRC RPA	; TO DIVISOR FIELD
3	016E	E0	WRM	
	016F	67	INC R5A0	
	0170	63	INC RPA0	
	0171	69	INC R5B0	
5	0172	7567	ISZ RPB0,DLJ	
	0174	A8	LD R5BF	; SET RPB TO RPB + 8
	0175	B4	XCH RPB E	
6	0176	A9	LD R5B0	
	0177	B5	XCH RPB0	
7	0178	C0	BBL 0	<i>end Sub</i>
	0179	CF	BRL 15	
8	017A	D0	ICLH: LDM 0	; MSUB CONTINUED
	017B	BC	XCH SRX	
9	017C	F5	RAL	
	017D	B0	XCH R0	; SAVE CY
10	017E	DA	LDM 10	
	017F	EB	ADM	
11	0180	1288	JC ICLF	; CY=1=>H.O.D.GTE 6=>NEG VALUE
	0182	A0	LD R0	
12	0183	F6	KAK	
	0184	12A4	JC ICEC	
13	0186	4194	JUN ICLC	
	0188	A0	ICLF: LD R0	; TO EXTEND SIGN
14	0189	F6	KAK	
	018A	1AA5	JNC ICEH	
15	018C	D9	LDM 9	
	018D	BC	XCH SRX	
16	018E	4194	JUN ICLC	
17	0190	1AA5	ICLG: JNC ICEH	
	0192	41A0	JUN ICLE	
18	0194	F9	ICLC: ICS	<i>TL5</i> ; SIGN EXTENDED SUBTRACT
	0195	9C	SUB SRX	
19	0196	27	SRC RSA	
	0197	F1	CLC	
20	0198	EB	ADM	

4004 MACRO ASSEMBLER, VER 2.2 TEST PROGRAM ERRORS = 0 PAGE 8

21	0199	FB	DAA
22	019A	E0	WRM
23	019B	AC	LD SRX
24	019C	1C90	JNZ ICLG
25	019E	12A4	JC ICEC
26	01A0	7794	ICLC: ISZ R5A0,ICLC
27	01A2	1CA5	JNZ ICEH
28	01A4	F3	ICEC: CMC

```
01A5 F5 ICEH: RAL
01A6 BC XCH SRX ; SAVE CARRY
01A7 F1 CLC
01A8 D8 LDM 8
01A9 83 ADD RPA0 ; RESTORE RPA
01AA B3 XCH RPA0
01AB F1 CLC
01AC AA LD RTPE
01AD 1CB0 JNZ RETMD
01AF AB LD RTP0
01B0 F6 RAR
01B1 1CB0 JNZ RETMD ; JIN RETURN
01B3 12B9 JC RETM1 ; RETURN IS TO MULT SUB
01B5 AC RETM2: LD SRX ; RESET CARRY
01B6 F6 RAR
01B7 4032 JUN MART
01B9 AC RETM1: LD SRX
01BA F6 RAR
01BB 4034 JUN MSRT
01BD AC RETMD: LD SRX
01BE F6 RAR ; RESET CARRY
01BF 3R JIN RTP
01C0 D0 DCLJ: LDM 0 ; MAD CONTINUED
01C1 BC XCH SRX
01C2 F5 RAL
01C3 B3 XCH R0 ; SAVE CY
01C4 DA LDM 10
01C5 E3 ADM
01C6 12CF JC DCLF ; NEG VALUE
01C8 A0 LD R0
01C9 F6 RAR
01CA 1AA5 JNC ICEH ;+ VALUE, CY = 1
01CC AC DCLG: LD SRX
01CD 4105 JUN DCLC ; EXTEND SIGN
01CE A0 DCLF: LD R0
01D0 F6 RAR
01D1 12A4 JC ICEC
01D3 D9 LDM 9
01D4 BC XCH SRX
01D5 410C JUN DCLG
01D7 12A4 DCLH: JC ICEC
01D9 41E4 JUN DCLF
01DB 27 DCLC: SRC RSA
```

Back to mult.

(X)

```
01DC EB not getting here ADM ; ADD SIGN EXTENDED
01DD FB DAA
```

we don't get here

To use their Test
prog just put
these reg

01DE	E3	WRM	
01DF	AC	LD SRX	
01E0	1CD7	JNZ DCLH	
01E2	1AA5	JNC ICEH	
01E4	7700 DB	ISZ R5A0, DCLC	
01E6	1CA4	JNZ ICEC	
01E8	41A5	JUN ICEH	
0200		ORG 200H	
TITLE ' TEST PROGRAM '			
0200	2280	BEGIN: FIM RPA, 80H	1000
0202	2490	FIM RPB, 98H	1001
0204	23	SRC RPA	
0205	EC	R00	
0206	5003	JMS MULT	
0208	2280	FIM RPA, 80H	
020A	2480	FIM RPB, 88H	
020C	FA	STC	
020D	23	SRC RPA	
020E	EC	R00	
020F	50A0	JMS DIV	
0211	4200	JUN BEGIN	
END			

RAM 2 Reg 0 MULTiplicand
RAM 2 Reg 1 MULTiplicier

NO PROGRAM ERRORS

000050000
1101
1100
1011
1010
1010
1001
1000
1000
1000
1000

BANK 1



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-25☒ 4004 ☐ 4040 ☐ 8008 ☐ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title	COMPLEMENT DECIMAL (CØMPL)
Function	Convert an 8 digit decimal number to its complement. (Change sign of number.)
Required Hardware	
Required Software	
Input Parameters	RPA (Register pointer) to RAM
Output Results	Number in complement form replaces number

Registers Modified: SRX(12),RPAØ(3)	Assembler/Compiler Used: ASF4
RAM Required: Uses Input RAM	Programmer: Hannah Fox
ROM Required: 13 bytes	Company: Caltech
Maximum Subroutine Nesting Level:	Address: 1201 E. Calif., Pasadena, CA

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
 2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
 3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.
 4. A source paper tape of the contributed program is required. This insures that a clear, original copy of the program is available to photo-copy for publication in a User's Library update publication.
-

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

```

      ;
      ;
      ;COMPLEMENT DECIMAL
      ;PARMS = RPA POINTS TO LOW ORDER DIGIT
      ;
      ;
0002      RPA      EQU 2
0003      RPA0     EQU 3
000C      SRX      EQU 12
      ;
      ;
0000      D8       COMPL:          LDM B
0001      BC       ACH SRX        ;SET COUNT
0002      FA       COMPM:         STC      ;SET CARRY
0003      F9       COMPM:         TCS
0004      23       SRC RPA
0005      E8       SBM            ;SUBTRACT FROM 10 OR 9
0006      F3       CMC
0007      FB       DAA
0008      E0       WRM            ;STORE OVER A
0009      63       INC RPA0
000A      7C03     ISZ SRX,COMPM
000C      C0       BBL 0
      END

```

NO PROGRAM ERRORS

M15
RAMWD(20 CH, 0 TO F, -)=000000000000126820000

Complement 00012682

*B1864

*J1860

CY CTR RESET

BREAK EXIT:

1864 FIM P1 80 0 0 0050000000000000 0 15 7 82

*E15

00000000999873180000

*

↓
- 00012682



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-26☒ 4004 ☐ 4040 ☐ 8008 ☐ 8080 ☐ 3000

(use additional sheets if necessary)

Program Title BINARY TO BCD CONVERTER

Function Converts an 8 bit binary number to a BCD number.

Required Hardware MCS4

Required Software None

Input Parameters 8 bit binary number in registers A (MSB) and B (LSB)

Output Results 3 digit BCD number
 hundreds in register 4,
 tens in register 5,
 units in register B.

 To Test: Load any 9 bit number in registers A, B,
 execute and examine registers 4, 5, and B for results.

Registers Modified: 2,3,4,5,7,A,B	Assembler/Compiler Used:
RAM Required: None	Programmer: J. Garner
ROM Required: 36 ₁₀ = 24H	Company: Bell Helicopter
Maximum Subroutine Nesting Level: None	Address: Research Electronics Box 482 Ft. Worth, Texas 76101

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
 2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
 3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.
 4. A source paper tape of the contributed program is required. This insures that a clear, original copy of the program is available to photo-copy for publication in a User's Library update publication.
-

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

```

0001      TZ      EQU 01H
0001      TO      EQU 01H
0009      TH      EQU 09H
0009      TI      EQU 09H
000A      CZ      EQU 0AH
000A      CO      EQU 0AH
0002      CN      EQU 02H
0002      C1      EQU 02H
0004      AZ      EQU 04H
0004      AO      EQU 04H
000C      AN      EQU 0CH
000C      NZA     EQU 0CH
0000      NC      EQU 00H

; 8 BIT BINARY TO BCD CONVERSION
; 8 BIT INPUT IN REGISTERS A (MSB) AND B (LSB)
; RESULT RETURNED IN REGISTERS
; 4 (HUNDREDS)
; 5 (TENS)
; 6 (UNITS)

```

```

0000      2400      FIM 4, 0      ;CLEAR REGISTERS
0002      2240      FIM 2, 64     ;LOAD 100 DECIMAL
0004      F1        HUND:      CLC      ;SUBTRACT 100 FROM NUMBER
0005      A8        LD 11
0006      93        SUB 3         ;SUBTRACT 4 FROM LSB
0007      B7        XCH 7         ;STORE INTERMEDIATE RESULT
0008      F3        CMC
0009      AA        LD 10         ;LOAD MSB
000A      92        SUB 2         ;SUBTRACT 96 FROM MSB
000B      1A12      JCN CO,HOUT    ;TO TENS IF RESIDUE < 100
000D      64        INC 4         ;INCREMENT HUNDREDS
000E      BA        XCH 10
000F      A7        LD 7         ;STORE RESIDUE
;NEXT STATEMENT NOT RECOGNIZED
; XCH
; JCN HUND          ;LOOP UNTIL RESIDUE < 100
0010      4C04      JCN HUND      ;LOAD 10 DECIMAL
0012      2200      HOUT:      FIM 2, 0      ;SUBTRACT TEN FROM RESIDUE
0014      F1        TENS:      CLC
0015      A8        LD 11
0016      93        SUB 3         ;SUBTRACT 10 FROM LSB
0017      B7        XCH 7         ;STORE INTERMEDIATE RESULTS
0018      F3        CMC
0019      AA        LD 10         ;LOAD MSB
001A      92        SUB 2
001B      1A23      JCN CO,TOUT    ;FINISHED IF <10
001D      65        INC 5         ;INCREMENT TENS
001E      BA        XCH 10
001F      A7        LD 7         ;STORE RESIDUE
0020      B8        XCH 11

```

4004 MACRO ASSEMBLER, VER 2.4 ERRORS = 0 PAGE 2

```
0021    4014          JUN TENS          ;LOOP UNTIL RESIDUE <10
0023    00          TOUT:-          NOP
                        END
NO PROGRAM ERRORS
```


SECTION 7

UNCLASSIFIED PROGRAMS

REFERENCE NUMBER		PROGRAMS
1.	4-17 Random Number Generator
2.	4-18 General Purpose ROM
3.	4-19 Translate HEX
4.	4-20 4x8 Keyboard Scanner
5.	40-13 "SEL" Subroutine (selector)
6.	40-14 Paper Tape Conversion, 5 Level TTY to 8 Level ASCII
7.	40-15 John Conway's Solitaire Game of Life



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-17☒ 4004 ☐ 4040 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

Random Number Generator

Function

**Verification Program to run subroutine also included

The subroutine generates a pseudo-random sequence of numbers.
The method used is; $X(J+L) = (A * (X(J) + C) \text{ MOD } 2^{**16})$

This is accomplished by shifting the old value (J) to the left (*2) an appropriate number of times and adding this result to the old value. C, an odd value, is then added giving the new value of (J). This is stored in RAM and made available to the calling program. **The verification program outputs 10 numbers per line after converting to octal.

Required
Hardware

None. **The verification pgm requires TTY on port 0.

Required
Software

A program is needed to call this subroutine. How the generated numbers are used must be determined by the user.

Input
Parameters

The subroutine uses RAM status characters to call the old value (J) and to store the result of its last computation RAM bank 0, Chip 0, Register 4 status characters are used.

Output
Results

Upon exit of this subroutine the new value is stored in the above mentioned status characters with the MSB in status character 0. Also register pairs 12 and 14 contain the new value of (J). The MSB is in register 12.

Registers Modified:	Assembler/Compiler Used:
8, 9, 10, 11, 12, 13, 14, 15	Resident Assembler V3.0
RAM Required:	Programmer:
Status Characters Register 4	Gary A. Winkler
ROM Required:	Company:
74 bytes	Howard Micro Systems
Maximum Subroutine Nesting Level:	Address:
One	C/O Howard Cox & Assoc. 6950 France Ave. S. Minneapolis, MN 55435

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

Your program will be photo-copied for publication in the User's Library. Please send an original, clear, un-marked copy.

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

ASSEMBLER V3.0

:1

:2

00B5

ORG 181

;RANDOM NUMBER GENERATOR

;

;THIS SUBROUTINE GENERATES A PSEUDO-RANDOM
;SEQUENCE OF NUMBERS IN THE RANGE OF
; $0 \leq N \leq 2^{16}-1$.

;

;A LINEAR CONGRUENTIAL SEQUENCE IS DERIVED
;FROM $X(J+1) = (A * (X(J) + C) \text{MOD } 2^{16})$.

;

;WHEN USING LESS THAN 16 BITS, USE THE MOST
;SIGNIFICANT BITS AS THEY ARE MOST RANDOM.
;THE CARRY BIT SHOULD BE USED FOR RANDOM
;N MOD 2.

;

;THE STATUS CHARACTERS OF RAM BANK 0,
;CHIP 0, REGISTER 4 ARE USED TO STORE THE
;OLD VALUE (J) WITH THE MSB IN STATUS
;CHARACTER 0. THE NEW VALUE (J+1) IS
;RETURNED TO THESE STATUS CHARACTERS AND
;IN REGISTER PAIRS 12 AND 14 FOR USE BY
;THE CALLING PROGRAM.

;

;EXECUTION TIME IS APPROXIMATELY 2.6 MS.

;

;

;

;

00B5 2C30
00B7 2D

RAND: FIM 12,48 ;GET ADDRESS OF OLD (J)
SRC 12

;

;COMPUTE A*J

;

00B8 EF
00B9 BF
00BA EE
00BB BE
00BC ED
00BD BD
00BE EC
00BF BC
00C0 D7
00C1 BB
00C2 50E2
00C4 50F2
00C6 DE
00C7 BB
00C8 50E2
00CA 50F2
00CC 2836

RD3 ;FETCH OLD VALUE

XCH 15

RD2

XCH 14

RD1

XCH 13

RD0

XCH 12

LDM 7 ;SET LOOP CNT. FOR 9

XCH 11 ;ITERATIONS (J*2+9)

JMS SHIFT

JMS PLUS ;ADD J TO RESULT OF MULT.

LDM 14 ;SET LOOP CNT. FOR 2

XCH 11 ;ITERATIONS (2+2)

JMS SHIFT

JMS PLUS ;ADD J TO THIS RESULT

FIM 8,54 ;FETCH C FOR ADDITION

00CE	2A23	FIM 10,35 ;TO FIND NEW VALUE OF J
00D0	F1	CLC
00D1	AB	LD 11
00D2	8F	ADD 15
00D3	E7	WR3 ;STORE NEW VALUE (J)
00D4	BF	XCH 15
00D5	AA	LD 10
00D6	8E	ADD 14
00D7	E6	WR2
00D8	BE	XCH 14
00D9	A9	LD 9
00DA	8D	ADD 13
00DB	E5	WR1
00DC	BD	XCH 13
00DD	A8	LD 8
00DE	8C	ADD 12
00DF	E4	WR0
00E0	BC	XCH 12 ;RETURN WITH (J) IN RP 12&14
00E1	C0	BBL 0 ;MSB IN REG. 12

;
;
;
;

00E2	F1	SHIFT: CLC ;ROUTINE TO MULT. BY 2
00E3	AF	LD 15
00E4	F5	RAL
00E5	BF	XCH 15
00E6	AE	LD 14
00E7	F5	RAL
00E8	BE	XCH 14
00E9	AD	LD 13
00EA	F5	RAL
00EB	BD	XCH 13
00EC	AC	LD 12
00ED	F5	RAL
00EE	BC	XCH 12
00EF	7BE2	ISZ 11,SHIFT
00F1	C0	BBL 0

;
;
;
;

00F2	F1	PLUS: CLC ;ROUTINE TO ADD (J) TO
00F3	EF	RD3 ;INTERMEDIATE RESULTS
00F4	8F	ADD 15
00F5	BF	XCH 15
00F6	EE	RD2
00F7	8E	ADD 14
00F8	BE	XCH 14
00F9	ED	RD1
00FA	8D	ADD 13
00FB	BD	XCH 13
00FC	EC	RD0
00FD	8C	ADD 12
00FE	BC	XCH 12
00FF	C0	BBL 0

;
;
;
;

END

ASSEMBLER V3.0

:1

:2

0000	5078	START:	JMS CRLF	;OUTPUT CR&LF
0002	D6		LDM 6	
0003	B7		XCH 7	;SET LOOP FOR PRINT OF 10
0004	50B5	NEXT:	JMS RAND	;NUMBERS PER LINE
0006	F1	CONV:	CLC	;CONVERT TO OCTAL
0007	AF		LD 15	
0008	F5		RAL	
0009	BF		XCH 15	
000A	AE		LD 14	
000B	F5		RAL	
000C	BE		XCH 14	
000D	AD		LD 13	
000E	F5		RAL	
000F	BD		XCH 13	
0010	AC		LD 12	
0011	F5		RAL	
0012	BC		XCH 12	
0013	F7		TCC	
0014	B3		XCH 3	;OUTPUT CHARACTER TO
0015	F0		CLB	;TTY AFTER EACH DIGIT
0016	B2		XCH 2	;IS CONVERTED
0017	506D		JMS DIGIT	
0019	5043		JMS PRINT	
001B	DB		LDM 11	;SET LOOP FOR PRINTING
001C	B8		XCH 8	;REMAINING 5 DIGITS
001D	DD	C01:	LDM 13	;SET LOOP FOR THREE
001E	B9		XCH 9	;SHIFTS (OCTAL)
001F	F0		CLB	
0020	B3		XCH 3	
0021	F1	C02:	CLC	
0022	AF		LD 15	
0023	F5		RAL	
0024	BF		XCH 15	
0025	AE		LD 14	
0026	F5		RAL	
0027	BE		XCH 14	
0028	AD		LD 13	
0029	F5		RAL	
002A	BD		XCH 13	
002B	AC		LD 12	
002C	F5		RAL	
002D	BC		XCH 12	
002E	A3		LD 3	
002F	F5		RAL	
0030	B3		XCH 3	
0031	7921		ISZ 9,C02	
0033	F0		CLB	
0034	B2		XCH 2	
0035	506D		JMS DIGIT	
0037	5043		JMS PRINT	
0039	781D		ISZ 8,C01	
003B	2220		FIM 2,32	;CONVERSION COMPLETE
003D	5043		JMS PRINT	;OUTPUT SPACE

```

003F 7704      ISZ 7,NEXT ;CHECK IF 10 NUMBERS PRINTED
0041 4000      JUN START  ;YES-START NEW LINE
00B5          RAND EQU 181
;
;
;
;
0043 2000      PRINT: FIM 0,0
0045 21        SRC 0
0046 F0        CLB
0047 E1        WMP
0048 D8        LDM 8
0049 B4        XCH 4
004A 5060      T01: JMS SBR2
004C F1        CLC
004D B2        XCH 2
004E F6        RAR
004F B2        XCH 2
0050 B3        XCH 3
0051 F6        RAR
0052 B3        XCH 3
0053 F7        TCC
0054 E1        WMP
0055 744A      ISZ 4,T01
0057 5060      JMS SBR2
0059 D1        LDM 1
005A E1        WMP
005B 5060      JMS SBR2
005D 5060      JMS SBR2
005F C0        BBL 0
0060 203C      SBR2: FIM 0,60
0062 7162      L2:  ISZ 1,L2
0064 7062      ISZ 0,L2
0066 203C      FIM 0,60
0068 7168      L1:  ISZ 1,L1
006A 7068      ISZ 0,L1
006C C0        BBL 0
;
;
;
;
006D 2430      DIGIT: FIM 4,48 ;CHANGE OCTAL CODE TO
006F F1        CLC           ;ASCII EQUIVALENT
0070 A5        LD 5
0071 83        ADD 3
0072 B3        XCH 3
0073 F7        TCC
0074 A4        LD 4
0075 82        ADD 2
0076 B2        XCH 2
0077 C0        BBL 0
;
;
;
;
0078 220D      CRLF: FIM 2,13
007A 5043      JMS PRINT
007C 220A      FIM 2,10
007E 5043      JMS PRINT
0080 C0        BBL 0
END

```

RUN OF VERIFICATION PROGRAM NUMBERS ARE IN OCTAL

```

.
.
033043 056322 113075 006524 014307 165006 174101 013550 165453 022772
142005 031074 010517 002256 136611 030720 127463 035042 064315 125044
124327 035126 004721 167470 121073 034512 122225 012414 177537 025376
176431 167640 162103 141562 113535 011364 032347 073246 133541 151410
112513 074232 060445 041734 064557 136516 054251 034560 152523 174302
020755 043704 136367 117366 000361 141330 142133 161752 174665 137254
047577 135636 150071 007500 101143 155022 004175 044224 040407 131506
163201 137250 027553 075472 067105 102574 130617 022756 061711 070420
165563 063542 045415 012544 140427 131626 064021 143170 153173 037212
137325 114114 107637 176076 011531 057340 010203 120262 164635 127064
036447 117746 102641 155110 134613 026732 165545 173434 164657 037216
157351 154260 170623 103002 162055 011404 132467 074066 037461 175030
154233 044452 171765 120754 137677 166336 143171 157200 107243 013522
035275 041724 024507 036206 112301 022750 031653 110172 154205 114274
010717 003456 145011 070120 163663 052242 166515 040244 114527 166326
103121 056670 145273 001712 114425 155614 157737 106576 164631 107040
176303 036762 175735 004564 002547 104446 011741 120610 116713 121432
032645 065134 024757 077716 022451 033760 146723 151502 063155 117104
066567 010566 036561 170530 126333 067152 127065 042454 167777 157036
076271 066700 055343 012222 026375 177424 150607 102706 001401 046450
173753 062672 001305 065774 031017 124156 170111 027620 121763 000742
047615 025744 030627 163026 062221 132370 077373 104412 031525 157314
170037 157276 077731 076540 124403 115462 147035 022264 106647 031146
061041 024310 041013 154132 037745 116634 025057 100416 025551 053460
065023 160202 124255 164604 162667 065266 175661 124230 040433 051652
024165 124154 160077 107536 171371 136400 163443 150722 157475 075124
034707 107406 030501 032150 076053 175372 166405 177474 011117 004656
153211 127320 020063 067442 070715 153444 104727 117526 001321 146070
171473 147112 106625 121014 140137 167776 153031 026240 012503 134162
060135 177764 152747 115646 070141 070010 123113 146632 005045 110334
165157 041116 170651 033160 143123 126702 125355 172304 016767 101766
074761 017730 112533 174352 061265 145654 110177 000236 024471 146100
031543 047422 050575 132624 061007 054106 017601 155650 140153 050072
113505 051174 131217 025356 076311 167020 056163 116142 052015 041144
121027 014226 060421 121570 023573 151612 123725 022514 050237 140476
166131 115740 040603 112662 131235 115464 157047 142346 037241 073510
145213 101332 112145 042034 065257 141616 073751 152660 161223 035402
066455 140004 013067 056466 134061 053430 124633 057052 056365 127354
000277 030736 017571 115600 037643 106122 101675 130324 045107 160606
146701 041350 142253 062572 000605 062674 011317 006056 161411 166520
054263 104642 173115 066644 075127 050726 077521 035270 015673 114312

```


101025	064214	120337	051176	141231	145440	026703	031362	142335	173164
123147	127046	146341	037210	127313	174032	157245	133534	125357	002316
137051	032360	137323	104102	167555	045504	147167	173166	133161	047130
076733	101552	013465	051054	030377	021436	152671	025300	005743	104622
072775	066024	171207	025306	036001	065050	104353	035272	025705	034374
031417	126556	004511	126220	012363	033342	054215	054344	011227	045426
056621	110770	147773	017012	016125	065714	130437	121676	054331	135140
155003	110062	113435	010664	027247	053546	015441	142710	051413	026532
164345	165234	125457	003016	142151	052060	055423	112602	030655	113204
043267	047666	072261	002630	011033	064252	110565	132554	020477	152136
045771	075000	114043	043322	024075	163524	055307	032006	065101	050550
006453	147772	013005	146074	011517	007256	167611	025720	110463	122042
075315	002044	065327	002126	175721	124470	042073	061512	073225	027414
100537	132376	127431	064640	043103	126562	024535	166364	073347	140246
024541	006410	133513	021232	131445	156734	065557	143516	105251	031560
133523	061302	031755	120704	077367	064366	171361	076330	063133	006752
145665	154254	150577	042636	101071	104500	162143	142022	115175	021224
101407	176506	054201	174250	050553	022472	140105	017574	131617	027756
112711	065420	146563	150542	056415	067544	101427	076626	055021	100170
074173	064212	110325	131114	010637	103076	142531	154340	071203	105262
075635	104064	077447	164746	173641	012110	155613	153732	036545	110434
165657	044216	010351	151260	151623	170002	173055	066404	073467	041066
030461	132030	075233	071452	142765	135754	040677	073336	074171	054200
170243	000522	146275	016724	065507	103206	003301	057750	052653	035172
025205	031274	011717	010456	176011	065120	144663	137242	177515	115244
055527	133326	074121	013670	066273	026712	065425	172614	060737	013576
115631	004040	057303	023762	106735	161564	043547	151446	102741	155610
137713	046432	103645	002134	025757	104716	053451	030760	127723	036502
074155	174104	027567	155566	027561	125530	047333	114152	100065	057454
070777	064036	027271	163700	136343	177222	137375	154424	011607	147706
072401	103450	014753	007672	052305	002774	032017	131156	021111	024620
102763	065742	060615	102744	171627	130026	053221	067370	020373	131412
002525	174314	071037	064276	030731	173540	005403	102462	060035	177264
147647	076146	152041	061310	062013	101132	110745	033634	026057	105416
056551	050460	046023	045202	135255	041604	123667	032266	166661	061230
161433	076652	175165	141154	061077	014536	122371	033400	044443	135722
070475	052124	075707	154406	121501	067150	117053	122372	037405	114474
012117	011656	004211	124320	001063	154442	101715	030444	045727	064526
172321	103070	112473	174112	057625	136014	041137	074776	104031	123240
073503	121162	171135	154764	013747	162646	161141	125010	144113	073632
056045	025334	166157	046116	021651	030160	124123	013702	136355	047304
157767	046766	065761	154730	033533	021352	032265	162654	011177	105236
155471	043100	112543	034422	161575	107624	122007	121106	110601	012650
161153	175072	164505	166174	132217	032356	127311	164020	037163	003142
063015	116144	062027	161226	051421	056570	144573	176612	074725	037514
151237	045476	117131	012740	121603	077662	042235	072464	020047	007346
130241	130510	166213	026332	163145	157034	066257	146616	124751	147660
142223	122402	077455	015004	154067	023466	125061	010430	045633	104052
027365	144354	101277	135736	150571	012600	120643	073122	012675	105324
106107	025606	037701	076350	163253	007572	051605	177674	012317	013056
012411	163520	035263	171642	004115	143644	036127	015726	070521	172270
136673	141312	052025	101214	021337	156176	072231	042440	107703	016362
053335	150164	164147	174046	037341	074210	150313	121032	030245	050534
126357	007316	170051	027360	120323	171102	000555	122504	110167	140166
124161	004130	017733	126552	164465	066054	131377	126436	103671	122300
066743	071622	003775	043024	032207	072306	127001	122050	125353	162272
076705	151374	032417	133556	035511	123220	173363	120342	065215	131344
152227	012426	047621	045770	070773	044012	167125	102714	031437	026676
005331	032140	036003	075062	024435	165664	070247	120546	106441	177710

072413	153532	035345	102234	126457	010016	173151	047060	036423	177602
041655	170204	004267	014666	063261	137630	132033	111252	061565	147554
121477	057136	176771	172000	175043	030322	135075	140524	116307	077006
156101	105550	027453	074772	064005	063074	012517	014256	020611	022720
071463	007042	106315	057044	026327	147126	166721	061470	163073	106512
044225	044414	001537	037376	060431	161640	124103	113562	135535	143364
134347	005246	115541	043410	154513	146232	002445	073734	066557	150516
136251	026560	114523	146302	042755	175704	040367	031366	162361	033330
004133	033752	116665	171254	051577	147636	032071	001500	043143	127022
026175	176224	142407	043506	145201	031250	071553	147472	011105	134574
132617	034756	143711	062420	127563	035542	067415	144544	042427	043626
046021	035170	015173	111212	061325	146114	111637	010076	073531	051340
152203	072262	006635	061064	140447	031746	064641	047110	176613	100732
107545	025434	166657	051216	041351	146260	132623	055002	004055	143404
034467	006066	021461	067030	016233	116452	113765	152754	141677	000336
025171	151200	051243	165522	057275	173724	126507	150206	074301	114750
073653	162172	076205	146274	012717	015456	027011	062120	125663	024242
010515	172244	016527	100326	065121	150670	007273	053712	036425	007614
161737	120576	046631	101040	140303	010762	017735	136564	104547	016446
173741	012610	160713	173432	154645	117134	026757	111716	104451	025760
110723	123502	105155	051104	170567	122566	020561	062530	170333	141152
051065	074454	171777	171036	160271	060700	017343	164222	050375	131424
052607	014706	163401	140450	035753	134672	123305	117774	033017	136156
052111	021620	063763	152742	071615	157744	132627	075026	044221	024370
141373	156412	153525	011314	172037	171276	161731	070540	066403	067462
171035	154264	010647	143146	043041	116310	103013	026132	161745	150634
027057	112416	107551	045460	027023	132202	146255	116604	064667	177266
157661	016230	102433	123652	146165	156154	162077	121536	053371	130400
125443	122722	001475	027124	136707	021406	012501	124150	140053	047372
110405	031474	013117	016656	035211	121320	162063	041442	112715	105444
006727	031526	163321	040070	033473	021112	030625	153014	142137	001776
035031	020240	154503	106162	102135	131764	054747	027646	052141	162010
165113	020632	127045	142334	167157	053116	052651	025160	105123	100702
147355	124304	120767	013766	056761	111730	154533	046352	003265	177654
112177	012236	106471	140100	173543	021422	072575	064624	163007	166106
001601	047650	002153	122072	035505	103174	133217	037356	160311	161020
020163	070142	074015	173144	023027	126226	042421	013570	065573	023612
045725	054514	052237	152476	050131	107740	002603	064662	153235	047464
061047	054346	021241	165510	007213	153332	034145	074034	067257	153616
155751	144660	123223	007402	110455	072004	115067	170466	116061	145430
166633	131052	000365	161354	002277	042736	101571	107600	001643	060122
123675	062324	147107	072606	130701	133350	004253	134572	122605	114674
013317	020056	043411	160520	016263	056642	015115	020644	177127	162726
061521	127270	057673	166312	023025	116214	122337	063176	023231	137440
170703	003362	164335	125164	025147	041046	130341	131210	171313	046032
101245	165534	127357	014316	021051	024360	101323	056102	011555	177504
051167	105166	115161	141130	140733	153552	135465	103054	032377	033436
034671	017300	147743	056622	114775	020024	073207	137306	020001	157050
146353	107272	147705	066374	033417	140556	066511	120220	154363	005342
076215	006344	113227	157426	040621	002770	011773	071012	140125	117714
132437	133676	136331	127140	117003	062062	135435	142664	131247	165546
177441	034710	113413	100532	106345	017234	127457	015016	024151	044060
017423	064602	052655	045204	145267	161666	054261	074630	053033	136252
032565	164554	022477	164136	127771	067000	056043	015322	046075	115524
157307	144006	047101	142550	050453	021772	135005	000074	013517	021256
051611	017720	052463	074042	117315	134044	167327	114126	157721	016470
104073	133512	015225	061414	102537	144376	011431	056640	005103	100562
046535	120364	175347	052246	006541	100410	175513	073232	053445	010734
067557	155516	167251	023560	075523	033302	053755	052704	001367	176366
153361	170330	125133	060752	067665	006254	152577	054636	163071	076500
124143	114022	137175	153224	003407	110506	036201	066250	112553	074472
062105	051574	133617	041756	174711	057420	110563	122542	100415	021544
003427	010626	037021	172170	136173	136212	032325	163114	012637	115076
024531	146340	033203	057262	117635	036064	001447	076746	155641	104110

017613	025732	160545	142434	167657	056216	072351	143260	113623	142002
015055	020404	175467	153066	012461	024030	137233	143452	064765	167754
042677	105336	156171	046200	132243	152522	170275	150724	167507	015206
165301	151750	114653	107172	147205	063274	013717	022456	060011	057120
106663	111242	021515	047244	157527	045326	056121	105670	130273	100712
007425	024614	062737	025576	177631	176040	021303	175762	130735	113564
145547	063446	064741	047610	001713	120432	025645	034134	027757	116716
135451	022760	071723	010502	116155	126104	131567	067566	011561	017530
111333	166152	022065	111454	072777	076036	111271	155700	100343	151222
161375	106424	113607	061706	054401	175450	056753	061672	174305	034774
034017	143156	103111	016620	044763	037742	102615	034744	073627	042026
035221	161370	062373	003412	124525	026314	073037	076276	112731	165540
147403	054462	102035	131264	051647	010146	134041	153310	124013	153132
032745	065634	030057	117416	140551	042460	010023	017202	157255	173604
025667	144266	150661	153230	023433	150652	117165	173154	063077	026536
004371	025400	006443	107722	112475	004124	177707	066406	103501	161150
161053	174372	161405	146474	014117	023656	066211	116320	143063	126442
123715	162444	147727	176526	154321	175070	154473	046112	001625	170014
043137	106776	166031	115240	035503	073162	013135	106764	115747	074646
143141	017010	006113	145632	000045	057334	170157	060116	103651	022160
066123	165702	160355	001304	061767	160766	047761	046730	075533	073352
154265	014654	013177	117236	037471	035100	054543	006422	003575	041624
024007	033106	072601	104650	023153	047072	106505	020174	134217	044356
011311	156020	001163	155142	105015	050144	164027	073226	033421	150570
006573	050612	016725	071514	153237	057476	001131	004740	063603	051662
064235	024464	122047	121346	112241	022510	030213	100332	105145	011034
070257	160616	006751	141660	104223	074402	121455	147004	056067	135466
107061	102430	107633	156052	151365	176354	103277	147736	032571	004600
062643	045122	034675	037324	010107	137606	021701	170350	025253	061572
173605	031674	014317	025056	074411	155520	177263	143642	026115	075644
140127	127726	052521	064270	000673	013312	174025	133214	023337	170176
154231	034440	051703	170362	075335	102164	066147	106046	021341	166210
012313	173032	152245	102534	130357	021316	052051	021360	062323	143102
022555	054504	012167	052166	106161	076130	061733	000552	106465	120054
133377	140436	165671	114300	030743	043622	025775	175024	134207	004306
111001	014050	167353	034272	020705	003374	034417	145556	117511	115220
135363	072342	107215	063344	054227	124426	031621	137770	132773	116012
111125	134714	033437	040676	067331	024140	000003	047062	046435	117664
172247	032546	070441	071710	134413	025532	157345	134234	130457	022016
055151	041060	000423	151602	063655	122204	106267	126666	045261	031630
174033	163252	003565	001554	123477	071136	060771	164000	137043	002322
157075	072524	020307	011006	140101	177550	071453	146772	006005	115074
014517	026256	102611	014720	033463	161042	130315	011044	130327	061126
150721	153470	025073	160512	166225	076414	003537	051376	142431	153640
066103	065562	157535	075364	036347	117246	077541	135410	016513	020232
124445	125734	070557	162516	020251	020560	056523	120302	064755	127704
142367	143366	144361	125330	046133	105752	040665	023254	053577	161636
114071	173500	005143	101022	050175	130224	044407	155506	127201	123250
133553	021472	133105	166574	134617	046756	025711	054420	071563	007542
111415	076544	144427	155626	030021	127170	057173	163212	003325	000114



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-18XX 4004 ☐ 4040 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title	General Purpose ROM
Function	30 cps terminal I/O 1702A programming Hex I/O Frees test line Single calls for CR, LF, Bell No JUN's or JMS's; may be put in any socket Standardized entry points on top of page
Required Hardware	SIM4 or Intellec 4 with 30 cps terminal or equivalent configuration
Required Software	Any driving routine
Input Parameters	See attachment; all entry points designed to chain and interwork easily
Output Results	Terminal I/O, PROMs, etc.

Registers Modified: see attachment	Assembler/Compiler Used: Hand assembled (listing supplied)
RAM Required: one	Programmer: John M. Harrison
ROM Required: Program requires one ROM	Company: Northeast Electronics
Maximum Subroutine Nesting Level: One	Address: Concord, New Hampshire

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

Your program will be photo-copied for publication in the User's Library. Please send an original, clear, un-marked copy.

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

GENERAL PURPOSE SYSTEMS ROM

John M. Harrison

November 23, 1974

Features

- Runs on SIM or Intellec 4 with standard I/O
- 30 CPS terminal I/O
- Frees test line for other use
- No JUN'S or JMS'S;
May be put in any ROM socket
- Standardized register usage
- Includes and, compare
- Includes HEX in and out
- Saves 50 locations compared to TTY I/O of 0540, 0541, 0543
- Programs and reads 1702 A PROMS
- Single calls for CR, LF, BELL
- Makes test routines easy to write
- Includes ROMPORT clear routine
- Standardized entry points on top of page; easy
to modify routines (i.e., FC, FA,...)

General Comments

Define JMP \equiv JCN 8 ; this is a same page jump. May be used when writing socket (page) independent code. Also JCN 0 is a skip instruction.

CPU PROBLEM? When "JCN 8 XX" is in locations, FFE & FFF execution continues at 0XX instead of FXX (on Intellec IV).
Some 1702A ROMS pass through programmer but forget in minutes

REGISTER USAGE AND CALLING MAP

<u>ROUTINE</u>	<u>NON-JMS CALLS</u>	<u>USES</u>	<u>P0</u>	<u>P1</u>	<u>P2</u>	<u>P3</u>	<u>P4</u>	<u>P5</u>	<u>P6</u>	<u>P7</u>	<u>RETURNS DATA IN ACC.</u>	<u>ENTRY POINT</u>
BELL	TTY OUT	----	X	X	X							FO
CR	TTY OUT	----	X	X	X							EE
LF	TTY OUT	----	X	X	X							EC
HEX OUT	TTY OUT	----	X	X/IN	X							FA
TTY OUTPUT	- - - -	----	X	IN	X							E4
TTY INPUT	- - - -	----	X	OUT	X	-	-	-	-	-	FAIL FLAG	FC
HEX CONVERT	- - - -	----	X	IN X/OUT	X	-	-	-	-	-	FAIL FLAG	F8
COMPARE	- - - -	----	X	IN	X	IN	-	-	-	-	PASS/ FAIL	F2
AND	- - - -	----	X	-/IN -/OUT		/IN						F4
CLEAR	- - - -	----	X									F6
ROM PGM	- - - -	----	X	DATA IN	X	ADDRESS IN	-	-	-	-	PASS/ FAIL	EA
ROM READ	- - - -	----	X	DATA OUT		ADDRESS						E8

X - Modifies Data in These Register Pairs

- or Blank Unused By Subroutine

NOTE: TTY INPUT ECHOES TO PRINTER & RETURNS ZEROED PARITY BIT. IF
HEXCON RETURNS WITH FAIL FLAG = 1, REGISTER 1 IS UNMODIFIED;
OTHERWISE R3 = HEX

REGISTER DEFINITIONS FOR SUBROUTINE USAGE

ACC	=	0	NO ERRORS BINARY CONDITION PASSES
ACC	=	1	BINARY CONDITION FAILS
ACC	=	0	ERROR
REGISTER PAIR	0	SCRATCH USAGE (MUST BE USED WITH FIN)	
	1	8 BIT ACCUMULATOR	
	2	SCRATCH USAGE	
	3	SECOND OPERATOR TO USE WITH P1	
	4	SCRATCH USAGE	
	5	CALLING ROUTINE	
	6	CALLING ROUTINE	
	7	MAIN ROUTINE	

ALL SUBROUTINE ENTRY POINTS IN TOP OF ROM (FC, FA, F8, ETC.)

CARRY BIT IS NOT PROTECTED BY SUBS OR CALLS, THUS GOOD PRACTICE SUGGESTS CLB UPON ENTRY TO SUBROUTINE.

.T100
.I100,1FF
.D100,1FF

GP ROM DEC. 6, 1974

0100	F0	D3	B2	D6	83	1A	0B	F2	B3	62	00	20	00	21	20	57
0110	A2	F6	B2	A3	F6	B3	A1	F6	B1	A1	E1	24	B7	74	1D	75
0120	1D	70	10	D1	E1	C0	20	00	21	24	67	20	BA	EA	F6	1A
0130	2D	71	31	70	31	74	3A	D1	E1	C0	EA	F4	E1	75	40	F0
0140	F6	A2	F6	B2	A3	F6	B3	20	78	18	31	F0	DC	82	14	59
0150	F2	14	54	C1	D5	83	12	53	C0	A3	14	53	D8	83	12	53
0160	B3	C0	20	00	21	F0	E2	60	A0	1C	64	C0	20	0B	D3	B3
0170	F5	B3	71	75	C0	F6	B0	B7	F5	B7	F6	80	18	6E	00	00
0180	00	00	00	00	00	F0	A3	97	1C	53	F0	A2	96	1C	53	C0
0190	20	00	21	A7	E2	60	21	A6	E2	60	21	EA	F4	E3	60	21
01A0	EA	F4	B2	C0	20	00	21	A7	E2	60	21	A6	E2	60	21	A3
01B0	F4	E2	60	21	A2	F4	E2	60	21	D2	E1	24	E0	71	BD	70
01C0	BD	75	BD	74	BD	F0	E1	71	C7	70	C7	20	20	21	EA	83
01D0	F2	1C	53	60	21	EA	82	1C	53	C0	22	07	18	0B	22	0D
01E0	18	0B	22	0A	18	0B	40	00	18	90	18	A4	18	E2	18	DE
01F0	18	DA	18	85	18	6C	18	62	18	4B	18	00	18	26	00	00

```

HEXOUT, CLB / GENERAL PURPOSE ROM
LDM 3 / BY J. M. HARRISON
XCH 2 / NORTHEAST ELECTRONICS
LDM 6 / CONCORD, N.H. 03301
ADD 3 / DECEMBER 3, 1974
JCN CZ TTYOUT / IS IT 0-9?
IAC / NO A TO F INCREMENT HI WORD
XCH 3 / ADD OFFSET
INC 2 /
NCP / GOODIES IN P1; CHAIN TO TTYOUT
TTYOUT, FIM P0 0
SRC P0
FIM P0 0101 0111/ 57 # OF HITS; STOP, STOP, START
LOOP, LD 2 / ONE BIG SHIFT REG.
RAR / OUTPUT IS STOP, STOP, START, B1,...,B8, RETURN
XCH 2
LD 3
RAR
XCH 3
LD 1
RAR
XCH 1
LD 1
WMP
FIM P2 1011 0111/ B7 1 BIT TIME AT 30CPS
L2, ISZ 4 L2
ISZ 5 L2
ISZ 0 LOCP / 11 HITS
LDM 1 / SUPPRESS TTY
WMP
RBL 0
TTYIN, FIM P0 0
SRC P0
FIM P2 0110 0111/ 67 CONSTANTS FOR PASS COUNTERS
FIM P0 1011 1010/RA 1/2 BIT TIME DELAY
ST2, RDR / DO WE HAVE START BIT YET?
RAR
JCN CZ ST2
ST1, ISZ 1 ST1 / YES 1/2 BIT DELAY
ISZ 0 ST1
ISZ 4 BY / ARE WE DONE?
LDM 1 / YES STOP TTY
WMP
RBL 0
BY, RDR / GET BIT
CMA
WMP / PUT IT ENT
ISZ 5 BY / IS THIS PARITY BIT
CLB / YES, ZERO IT
BYE, RAR / PUT BIT INTO SHIFT REG.
LD 2 / THIS IS AN 8 BIT S-R
RAR
XCH 2
LD 3
RAR
XCH 3
FIM P0 0111 1000/ 78 1 BIT TIME DELAY
JMP ST1 / ANOTHER BIT TIME
HEXCON, CLB / CLEAR CARRY BIT
LDM 12
ADD 2
JCN AZ H / IS THIS A-F MAYBE?
IAC
JCN AZ L / IS THIS 0-9 MAYBE?
RBL 1 / NOT HEX RETURN WITH ERROR
TILT, LD 5 / DESCRIPTION FOR USE
L,
JCN ZFRIT
JMP RORFAL
JMP ROMPGF
JCN 1

```

```

TILT,      JCN AZ 11      / IS THIS RTT MAILED
L,          IAC
           JCN AZ L      / IS THIS 0-9 MAYBE?
           BBL 1        / NOT HEX RETURN WITH ERROR
           LDM 5
           ADD 3
           JCN CN TILT   / IS IT 0-9
H,          BBL 0        / YES, DONE
           LD 3
           JCN AZ TILT   / IS IT 0?
           LDM 8        / ADD OFFSET
           ADD 3
           JCN CN TILT   / >1?
           XCH 3        / NO, MAKE DATA GOOD
           BBL 0        / GET OUT
CLEAR,      FIM P0 0     / SET UP P0
A,          SRC P0
           CLB          / CLEAR AC
           WRR          / CLEAR PCRT
           INC 0        / NEXT PCRT
           LD 0
           JCN AN A      / IS THIS LAST PORT?
           BBL 0        / YES, GET OUT!
AND,        FIM P0 0000 1011 / 08 COOKBOOK AND
           / FROM INTELLEC IV
LI,         LDM 3        / EXCEPT FOR
           XCH 3
           RAL
           XCH 3
           ISZ 1 L2      / THIS ISZ
           / DUNNO WHY
L2,         BBL 0        / THEY DIDN'T
           RAR          / USE IT?
           XCH 0
           XCH 7
           PAL
           XCH 7
           RAR
           ADD 0
           JMP L1
           NOP
           NOP
           NOP
           NOP
           NOP
           NOP
COMPARE,    CLB          / CARRY BIT MUST BE ZERO
           LD 3
           SUB 7
           JCN AN TILT   / WHOOPS
           CLB          / CARRY BIT MUST BE ZERO
           LD 2
           SUB 6
           JCN AN TILT
           BBL 0
ROMREAD,    FIM P0 0000 0000 / 00 ADDRESS ON PORTS 1,0
           / DATA OUT ON PORTS 3,2
           / DATA IN ON PORTS 3,2
           SRC P0
           LD 7
           WRR          / SET UP LOW ADDRESS
           INC 0
           SRC P0
           LD 6
           WRR          / SET UP HIGH ADDRESS
           INC 0        / GET FOGGIES
           SRC P0
           RDR
           CMA
           XCH 3        / RESELECT LOW PORT
           INC 0        / GET FOGGIES

```

```

SRC P0
RDR
CMA
XCH 7
BHL 0 / DONE
WRITE, FIM P0 0
SRC P0
LD 7
WRR
JNC 0
SRC P0
LD 6
WRR
INC 0
SRC P0
LD 3
CMA
WRR
INC 0
SRC P0
LD 2
CMA
WRR
INC 0
SRC P0
LD 2
WMP
FIM P2 1101 0000
L1, ISZ 1 L1
ISZ 0 L1
/ THIS IS A 01 SECOND TIME OUT
JSZ 5 L1
ISZ 4 L1
CLH
WMP / TURN OFF BURN
L2, ISZ 1 L2
ISZ 0 L2
FIM P0 00100000
SRC P0
RDR / CHECK LOW DIGIT
ADD 3
IAC
JCN AN TILT
INC 0
SRC P0
RDR / CHECK HIGH DIGIT
ADD 2
JCN AN TILT
BBL 0
BELL, FIM P1 0000 0111
JMS TTYOUT
CR, FIM P1 0000 1101
JMS TTYOUT
LF, FIM P1 0000 1010
* 1110 0100
JMP TTYOUT / VARIOUS ENTRY POINTS SEE
/ DESCRIPTION FOR USE
JUN ZERO
JMP R0 READ
JMP ROMPGM
JMP LF
JMP CR
JMP BELL
JMP COMPARE
JMP AND
JMP CLEAR
JMP HEXCONVERT
JMP HEXOUT
JMP ITYIN
/ CANNOT USE CPU BUG JMP GOES TO 000 INSTEAD OF 400

```



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-20☒ 4004 ☐ 4040 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program Title	"KEYCN"
Function	Refer to Part A of attached sheet.
Required Hardware	None, except for pull up resistors on ROM input port lines. Refer to Part B of attached sheet.
Required Software	None
Input Parameters	A zero on a ROM input line means a switch has been actuated on that line.
Output Results	KEYCN returns row and column identification (in binary) as follows: RAM CHIP 0, REG. 0, CHAR. 0 = Row Identification RAM CHIP 0, REG. 0, CHAR. 1 = Column Identification

Registers Modified: 0,1,2,3,13,14 & 15	Assembler/Compiler Used: Resident MCS-4 Assembler, V3.0
RAM Required: 2 bytes	Programmer: K.J. Fisher
ROM Required: 151 bytes	Company: Reserve Mining Company
Maximum Subroutine Nesting Level: One	Address: Silver Bay, Minnesota 55614

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

Your program will be photo-copied for publication in the User's Library. Please send an original, clear, un-marked copy.

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

KEYCN

PART A:

"KEYCN" is a subroutine which scans, reads, debounces, and stores the row and column identification of an actuated (normally open) switch wired into a 4x8 Matrix as shown in Part B below.

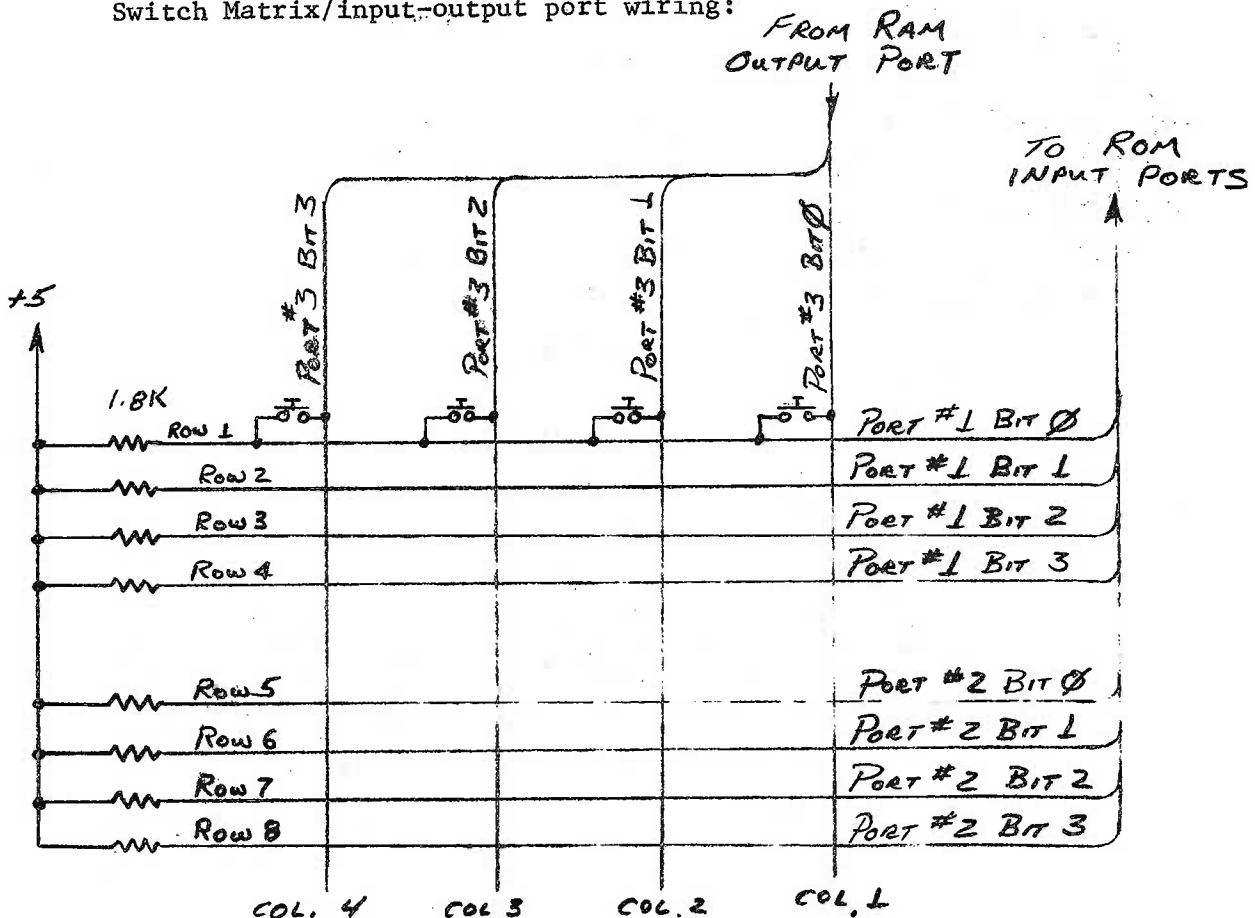
Debounce time is 15 MS and is easily changeable by modifying the register initialization instructions of the "Wait 15 MS." portion of the program.

The "KEY RELEASED" flag is necessary so that the program does not assume a new entry in the case of the entire program (of which "KEYCN" is a part) running to completion and coming back to the keyboard scan subroutine before the operator gets his hand off the switch.

Note that the column lines (RAM output ports) are negative true. ie. Program 1 = 0 volts and Program 0 = +5 volts
Therefore to "Assert" a column, that is set that column line to 0 volts, one writes a Program "1" to that bit on the RAM output port. Actuated switches will then show up as 0 volts = Program 0, on the ROM input port lines, since ROM input ports use positive true logic.

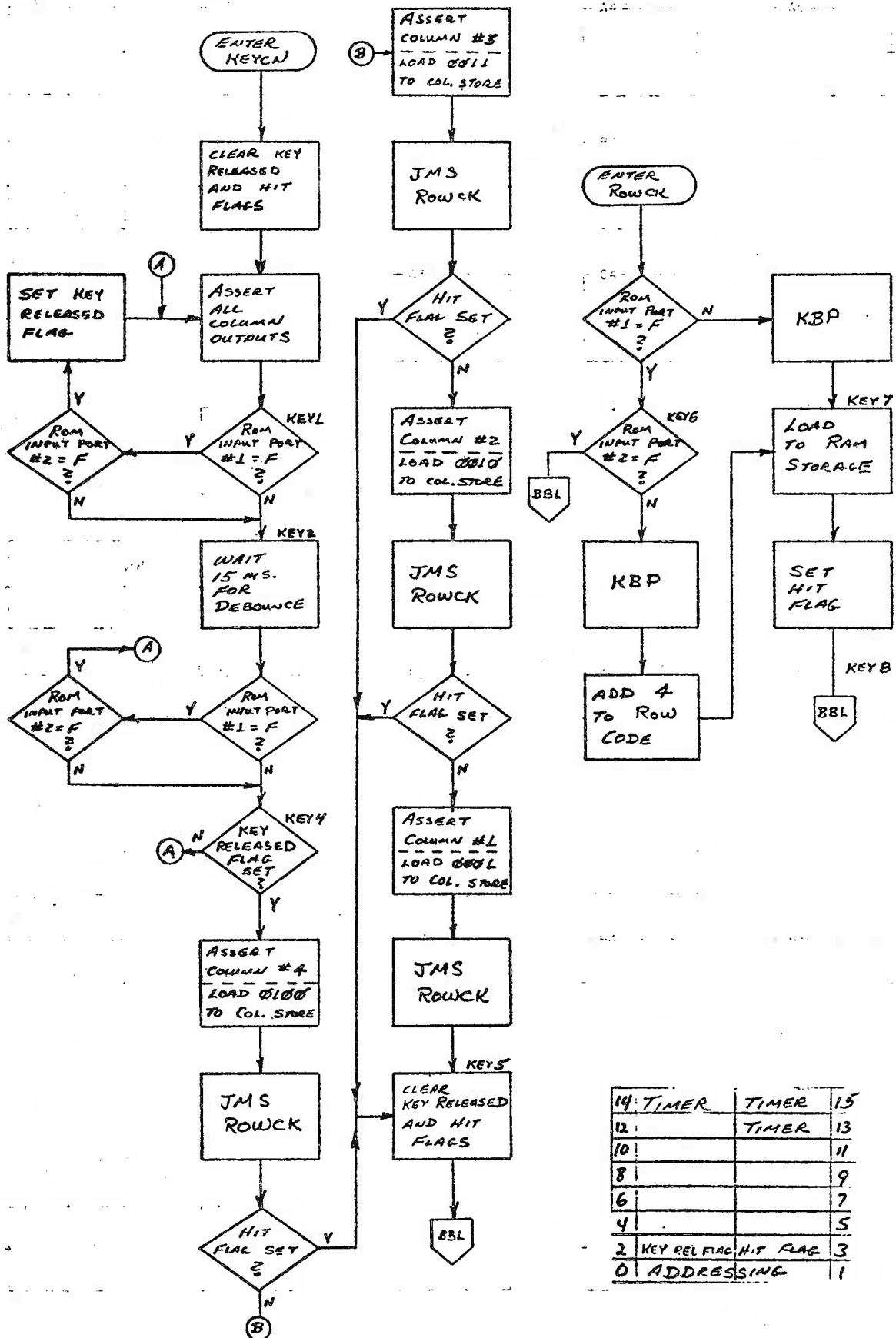
PART B:

Switch Matrix/input-output port wiring:



NOTE: Only 4 of 32 possible switches are shown.

Programmer: K. FISHER Program No.: _____ Date: 3/3/75 Page: _____
Chart ID: _____ Chart Name: _____ Program Name: KEYCN



14	TIMER	TIMER	15
12		TIMER	13
10			11
8			9
6			7
4			5
2	KEY REL FLAG	HIT FLAG	3
0	ADDRESSING		1

;*****

```

;
; SUBROUTINE "KEYCN" ( FOR X8 SWITCH MATRIX)
;
; "KEYCN" SCANS, FEELS, DEF UNCES, AND STOPES
; THE ROW AND COLUMN IDENTIFICATION OF AN
; ACTUATED SWITCH WIPED TO CONNECT BETWEEN
; A ROW (ROM INPUT LINE) AND A COLUMN (PAM
; OUTPUT LINE).
;
;   ROW 1=PCM INPUT PCPT #1, EIT 0
;   ROW 2=PCM INPUT PCPT #1, EIT 1
;   ROW 3=PCM INPUT PCPT #1, EIT 2
;   ROW 4=PCM INPUT PCPT #1, EIT 3
;   ROW 5=PCM INPUT PCPT #2, EIT 0
;   ROW 6=PCM INPUT PCPT #2, EIT 1
;   ROW 7=PCM INPUT PCPT #2, EIT 2
;   ROW 8=PCM INPUT PCPT #2, EIT 3
;
;   COL 1=RAM CUPPUT PORT #3, EIT 0
;   COL 2=RAM OUTPUT PCPT #3, EIT 1
;   COL 3=RAM OUTPUT PCPT #3, EIT 2
;   COL 4=RAM OUTPUT PORT #3, EIT 3
;
; THE SUBROUTINE RETURNS SWITCH IDENTIFICATION
; AS FOLLOWS:
;   RAM CHIP 0, REG.0, CHAR 0 = ROW IDENT.
;   RAM CHIP 0, REG.0, CHAR 1 = COL IDENT.
; ROW AND COL IDENT. ARE IN BINARY.
;
; NOTE: ROM INPUT PORTS ARE POSITIVE TRUE
; LOGIC AND PAM OUTPUT PORTS ARE NEGATIVE TRUE
; LOGIC.

```

;*****

```

NCP
NCP
FIM 2,0      ; CLEAR KEY REL. AND HIT FLAGS
KEYCN: FIM 0,0C0H      ; ASSEPT ALL COLUMN OUTPUTS
SRC 0
LIM 15
WMP
FIM 0,10H      ; ROM INPUT PCPT #1=F?
SRC 0
RDP
CMA
JNZ KEY2
FIM 0,20H      ; ROM INPUT PORT #2=F?
SRC 0
RDR
CMA
JNZ KEY2
LIM 1      ; SET KEY PELEASED FLAG
XCH 2
JUN KEYCN
KEY2: FIM 14,40H      ; WAIT 15 MS.
LIM 10
XCH 13
KEY3: ISZ 13,KEY3
ISZ 14,KEY3
ISZ 15,KEY3

```

~~151~~ WORDS

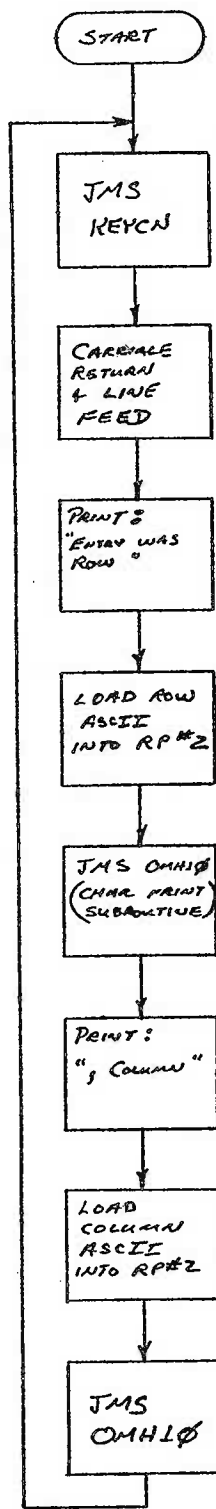
```
FIM 0, 10H      ; RCM INPUT PORT #1=F?
SPC 0
RIR
CMA
JNZ KEY4
FIM 0, 20H      ; RCM INPUT PORT #2=F?
SRC 0
RIR
CMA
JZ KEYCN
JUN KEY4
NOP
NOP
NOP
NOP
KEY4: LD 2      ; KEY RELEASED FLAG SET?
JZ KEYCN
FIM 0, 0C0H     ; ASSERT ONLY COL. #4 LINE (LOW)
SRC 0
LIM 8
WMP
FIM 0, 01H     ; LOAD 0100 TO COL. FAM STORAGE
SRC 0
LIM 4
WRM
JMS ROWCK
LD 3           ; HIT FLAG SET?
JNZ KEY5
FIM 0, 0C0H     ; ASSERT ONLY COL. #3 LINE(LOW)
SRC 0
LIM 4
WMP
FIM 0, 01H     ; LOAD 0011 TO COL. FAM STORAGE
SRC 0
LIM 3
WRM
JMS ROWCK
LD 3           ; HIT FLAG SET?
JNZ KEY5
FIM 0, 0C0H     ; ASSERT ONLY COL. #2 LINE(LOW)
SRC 0
LIM 2
WMP
FIM 0, 01H     ; LOAD 0010 INTO COL. FAM STORAGE
SRC 0
LIM 2
WRM
JMS ROWCK
LD 3           ; HIT FLAG SET?
JNZ KEY5
NOP
NOP
FIM 0, 0C0H     ; ASSERT ONLY COL. 1 LINE(LOW)
SRC 0
LIM 1
WMP
FIM 0, 01H     ; LOAD 0001 INTO COL. FAM
                ; STORAGE
```

```
      SRC 0
      LDM 1
      WPM
      JMS POWCK
KEY5:  FIM 2,00H      ; CLEAR KEY RELEASED
      NOP      ; AND HIT FLAGS
      EEL 0
ROWCK: FIM 0,10H      ; ROM INPUT PCRT #1=F?
      SRC 0
      RDR
      CMA
      JZ KEY6
      KEP      ; CONVERT 1 OF 4 TO BINARY
KEY7:  FIM 0,00H      ; LOAD TO RAM STORAGE
      SRC 0
      WPM
      LDM 1      ; SET HIT FLAG
      XCH 3
KEY8:  EEL 0
KEY6:  FIM 0,20H      ; ROM INPUT PCRT #2=F?
      SRC 0
      RDR
      CMA
      JZ KEY8
      KEP      ; CONVERT 1 OF 4 TO BINARY
      FIM 0,04H      ; ADD 4 TO ROW CODE.
      ADD 1
      JUN KEY7
```

IBM Flowcharting Worksheet

PRINTED IN U.S.A.
S.C. 211 1050

Programmer: K. FISHER Program No.: 4PTP-5 Date: 2/10/75 Page:
Chart ID: Chart Name: Program Name: KEYCN TEST



↑ Fold under at dotted line.

↑ Fold under at dotted line.

ASSEMBLER V3.0

:1

:2

```

; MPTP-5.....KEYCN TEST
;
;
000D      CR EQU 0DH
000A      LF EQU 0AH
;
;
0000      00      NOP
0001      00      NOP
0002      401E     JUN BEGIN      ; JUMP AROUND CANNED MESSAGES
0004      454E5452 M1: DB 'ENTRY WAS: ROW @'
          59205741
          533A2052
          4F572040
0014      2C20434F M2: DB ', COLUMN @'
          4C554D4E
          2040
001E      50E9     BEGIN: JMS KEYCN
0020      220D     FIM 2,CR      ; CARRIAGE RETURN
0022      5089     JMS CMH10
0024      220A     FIM 2,LF      ; LINE FEED
0026      5089     JMS CMH10
0028      2004     FIM 0,M1      ; PRINT M1
002A      5044     JMS CMH
002C      2000     FIM 0,00H     ; RP2 GETS ROW ASCII
002E      21       SRC 0
002F      E9       RDM
0030      E3       XCH 3
0031      D3       LDM 3
0032      E2       XCH 2
0033      5089     JMS CMH10     ; PRINT ROW IDENT.
0035      2014     FIM 0,M2     ; PRINT M2
0037      5044     JMS CMH
0039      2001     FIM 0,01H    ; RP2 GETS COLUMN ASCII
003E      21       SRC 0
003C      E9       RDM
003D      E3       XCH 3
003E      D3       LDM 3
003F      E2       XCH 2
0040      5089     JMS CMH10     ; PRINT ROW IDENT.
I 0042      401E     JUN BEGIN

```

```

;*****
;
; OUTPUT MESSAGE AND TTY HANDLER, "CMH".
;
; LOAD MESSAGE S.A. INTO RP 0
; ENTER AT "CMH"
;

```

```
; TO PRINT ONE CHAR. LOAD ASCII INTO
; R2 & R3 AND ENTER AT "CMH10"..
;
; USES REG. 0, 1, 2, 3, 4, 6 & 7.
;
;
```

0044	32	CMH:	FIN 2	; R2 AND R3 GET ASCII CHAR.
0045	F1	CMH1:	CLC	; IS IT END CHAR? I.E. =e=40H?
0046	D4		LIM 4	
0047	92		SUB 2	
0048	1C50		JCN 12, CMH2	
004A	F1		CLC	
004B	D0		LIM 0	
004C	93		SUB 3	
004D	1C50		JCN 12, CMH2	
004F	C0		BFL 0	; END OF MESSAGE
0050	5089	CMH2:	JMS CMH10	; PRINT CHAR.
0052	F0	CMH3:	CLB	; INC ADDR
0053	E1		XCH 1	
0054	F2		IAC	
I 0055	B1		XCH 1	
I 0056	80		ADD 0	
0057	E0		XCH 0	
0058	40,44		JUN CMH	

```
; THE FOLLOWING SUBROUTINES COMPRISE THE TTY
; HANDLER. ASCII CHAR. ARE LOADED INTO OR READ
; FROM R2 & R3.
;
;
```

```
INPUT A CHAR. FROM THE TTY
;
```

005A	F0	CMH6:	CLB	
005B	2640	CMH7:	FIM 6, 40H	
005D	27		SRC 6	
005E	E1		WMP	
005F	D8		LIM 8	
0060	E4		XCH 4	; REG 4=BIT COUNTER
0061	50A4		JMS CMH12	
0063	EA	CMH8:	RDR	
0064	F6		RAR	; SHIFT TO CARRY
0065	1A63		JNC CMH8	; LOOP WAITING FOR START BIT
0067	50AE		JMS CMH15	; 4.55 MS DELAY
0069	27		SRC 6	
006A	E1		WMP	; TURN READER OFF
006B	21		SRC 0	
006C	E1		WMP	; ECHO START BIT
006D	50A8	CMH9:	JMS CMH13	; 9.09 MS DELAY
006F	EA		RDR	; INPUT DATA
0070	F4		QMA	; COMPLEMENT ACC
0071	E1		WMP	; ECHO DATA BIT
0072	F6		RAR	; BIT TO LINK
0073	A2		LD 2	; GET UPPER NIBBLE
0074	F6		RAR	; SHIFT IN CARRY
0075	B2		XCH 2	; SAVE UPPER NIBBLE
0076	A3		LD 3	; GET LOWER NIBBLE
0077	F6		RAR	; SHIFT IN CARRY
0078	B3		XCH 3	; SAVE LOWER NIBBLE
0079	746D		ISZ 4, CMH9	; GET ALL 8 BITS

```

007E 50A8 JMS OMH13 ; STOP BIT 1
I 007D D1 LDM 1
007E E1 WMP ; ECHO STOP BIT 1
007F 50A8 JMS OMH13
0081 50AE JMS OMH15 ; 4.55 MS DELAY
0083 E2 XCH 2 ; ELIMINATE PARITY BIT
0084 F5 RAL
0085 F1 CLC
0086 F6 RAP
0087 E2 XCH 2
0088 C0 BEL 0
;
; OUTPUT A CHAR TO TTY
;
0089 50A4 OMH10: JMS OMH12 ; SELECT TTY PORT
008B E1 WMP ; START BIT
008C D8 LDM 8
008D B4 XCH 4 ; REG 4=BIT COUNTER
008E 50A8 OMH11: JMS OMH13 ; 9.09 MS DELAY
0090 F1 CLC
0091 B2 XCH 2 ; SHIFT 8 BITS RIGHT
0092 F6 RAR
0093 E2 XCH 2
0094 E3 XCH 3
0095 F6 RAR
0096 E3 XCH 3
0097 F7 TCC ; CARRY TO ACC
0098 E1 WMP ; OUTPUT TO ACC
0099 748E ISZ 4, OMH11 ; CONTINUE LOOPING
009B 50A8 JMS OMH13
*18C
009D D1 LDM 1 ; STOP BIT 1
009E E1 WMP
009F 50A8 JMS OMH13 ; 9.09 MS DELAY
00A1 50A8 JMS OMH13 ; AND ANOTHER
00A3 C0 BEL 0 ; RETURN
;
; TIMING SUBROUTINES
;
00A4 2600 OMH12: FIM 6, 0
00A6 27 SRC 6
00A7 C0 BEL 0
00A8 263C OMH13: FIM 6, 60 ; 9.09 MS DELAY
00AA 77AA OMH14: ISZ 7, OMH14
00AC 76AA ISZ 6, OMH14
00AE 263C OMH15: FIM 6, 60 ; 4.55 MS DELAY
00B0 77E0 OMH16: ISZ 7, OMH16
00B2 76E0 ISZ 6, OMH16
I 00B4 C0 BEL 0

```

```

;*****

```

```

;
; SUBROUTINE "KEYCN" ( FOR 4X8 SWITCH MATRIX)
;
; "KEYCN" SCANS, READS, DEBOUNCES, AND STORES
; THE ROW AND COLUMN IDENTIFICATION OF AN
; ACTUATED SWITCH WIRED TO CONNECT BETWEEN
; A ROW (FROM INPUT LINE) AND A COLUMN (FROM
; OUTPUT LINE).
;

```

```

; ROW 1=PCM INPUT PORT #1, EIT 0
; ROW 2=PCM INPUT PORT #1, EIT 1
; ROW 3=PCM INPUT PORT #1, EIT 2
; ROW 4=PCM INPUT PORT #1, EIT 3
; ROW 5=PCM INPUT PORT #2, EIT 0
; ROW 6=PCM INPUT PORT #2, EIT 1
; ROW 7=PCM INPUT PORT #2, EIT 2
; ROW 8=PCM INPUT PORT #2, EIT 3
;
; COL 1=RAM OUTPUT PORT #3, EIT 0
; COL 2=RAM OUTPUT PORT #3, EIT 1
; COL 3=RAM OUTPUT PORT #3, EIT 2
; COL 4=RAM OUTPUT PORT #3, EIT 3
;
; THE SUBROUTINE RETURNS SWITCH IDENTIFICATION
; AS FOLLOWS:
;   RAM CHIP 0, REG. 0, CHAR 0 = ROW IDENT.
;   RAM CHIP 0, REG. 0, CHAR 1 = COL IDENT.
; ROW AND COL IDENT. ARE IN BINARY.
;
; NOTE: ROM INPUT PORTS ARE POSITIVE TRUE
; LOGIC AND RAM OUTPUT PORTS ARE NEGATIVE TRUE
; LOGIC.

```

```

00E5 00 NOP
00E6 00 NOP
00E7 2200 FIM 2,0 ; CLEAR KEY REL. AND HIT FLAGS
I 00E9 2000 2000 KEYCN: FIM 0,0 CH ; ASSERT ALL COLUMN OUTPUTS
00EB 21 SRC 0
00EC DF LDM 15
00ED E1 WMP
00EE 2010 FIM 0,10H ; ROM INPUT PORT #1=F?
00C0 21 SRC 0
00C1 EA RDR
00C2 F4 CMA
00C3 1CD0 JNZ KEY2
00C5 2020 FIM 0,20H ; ROM INPUT PORT #2=F?
00C7 21 SRC 0
00C8 EA RDR
00C9 F4 CMA
00CA 1CD0 JNZ KEY2
00CC D1 LDM 1 ; SET KEY RELEASED FLAG
00CD B2 XCH 2
00CE 40E9 JUN KEYCN
00D0 2E4D KEY2: FIM 14,4DH ; WAIT 15 MS.
00D2 DA LDM 10
00D3 BD XCH 13
00D4 7DD4 KEY3: ISZ 13,KEY3
00D6 7ED4 ISZ 14,KEY3
00D8 7FD4 ISZ 15,KEY3
I 00DA 2010 FIM 0,10H ; ROM INPUT PORT #1=F?
00DC 21 SRC 0
00DD EA RDR
00DE F4 CMA
00DF 1CEE JNZ KEY4
00E1 2020 FIM 0,20H ; ROM INPUT PORT #2=F?
00E3 21 SRC 0
I 00E4 EA RDR
00E5 F4 CMA
00E6 14D0 ISZ KEYCN

```


00E8	40EE	JUN KEY4
00EA	00	NCP
00EE	00	NOP
00EC	00	NOP
00ED	00	NOP
00EE	A2	KEY4: LD 2 ; KEY RELEASED FLAG SET?
00EF	14E9	JZ KEYCN
00F1	20C0	FIM 0,0C0H ; ASSERT ONLY COL. #4 LINE (LOW)
00F3	21	SRC 0
00F4	D3	LIM 8
00F5	E1	WMP
00F6	2001	FIM 0,01H ; LOAD 0100 TO COL. RAM STORAGE
00F8	21	SRC 0
00F9	D4	LIM 4
00FA	E0	WRM
00FE	5130	JMS ROWCK
00FD	A3	LD 3 ; HIT FLAG SET?
V 00FE	1B2C 1C2C	JNZ KEY5
0100	20C0	FIM 0,0C0H ; ASSERT ONLY COL. #3 LINE (LOW)
0102	21	SRC 0
0103	D4	LIM 4
0104	E1	WMP
0105	2001	FIM 0,01H ; LOAD 0011 TO COL. RAM STORAGE
0107	21	SRC 0
0108	D3	LIM 3
0109	E0	WRM
010A	5130	JMS ROWCK
010C	A3	LD 3 ; HIT FLAG SET?
010D	1C2C	JNZ KEY5
010F	20C0	FIM 0,0C0H ; ASSERT ONLY COL. #2 LINE (LOW)
0111	21	SRC 0
0112	D2	LIM 2
0113	E1	WMP
0114	2001	FIM 0,01H ; LOAD 0010 INTO COL. RAM STORAGE
0116	21	SRC 0
0117	D2	LIM 2
0118	E0	WRM
0119	5130	JMS ROWCK
011E	A3	LD 3 ; HIT FLAG SET?
011C	1C2C	JNZ KEY5
011E	00	NOP
011F	00	NOP
0120	20C0	FIM 0,0C0H ; ASSERT ONLY COL. 1 LINE (LOW)
0122	21	SRC 0
0123	D1	LIM 1
0124	E1	WMP
0125	2001	FIM 0,01H ; LOAD 0001 INTO COL. RAM STORAGE
0127	21	SRC 0
0128	D1	LIM 1
0129	E0	WRM
012A	5130	JMS ROWCK
012C	2200	KEY5: FIM 2,00H ; CLEAR KEY RELEASED
012E	00	NOP ; AND HIT FLAGS
012F	C0	REL 0
0130	2010	ROWCK: FIM 0,10H ; RCI INPUT PORT #1=F?
0132	21	SRC 0
0133	EA	RDR
0134	F4	QMA
0135	143F	JZ KEY6

```

0137 FC      KEY:      ; CONVERT 1 OF 4 TO BINARY
0138 2000    KEY7:    FIM 0,00H      ; LOAD TO RAM STORAGE
013A 21      SRC 0
013B E0      WPM
013C D1      LDM 1      ; SET HIT FLAG
013D B3      XCH 3
013E C0      KEY8:    BEL 0
013F 2020    KEY6:    FIM 0,20H      ; ROM INPUT PORT #2=F?
0141 21      SRC 0
0142 EA      RDR
0143 F4      CMA
0144 143E    JZ KEY8
0146 FC      KEY:      ; CONVERT 1 OF 4 TO BINARY
0147 2004    FIM 0,04H      ; ADD 4 TO ROW CODE.
0149 81      ADD 1
014A 4138    JUN KEY7

```

END

:3

```

:0F0000000000401E454E545259205741533A209C
:0F000F00524F5720402C20434F4C554D4E204010
:0F001E0050E9220D5089220A50892004504420E5
:0F002D000021E9E3D3E25089201450442001219F
:0F003C00E9E3D3E25089401E32F1D4921C50F177
:0F004E00D0931C50C05039F0E1F2E130E0404446
:0F005A00F0264027E1D8E450A4EAF61A6350AE5E
:0F00690027E121E150A8EAF4E1F6A2F6F2A3F6EE
:0F007800B3746D50A8D1E150A850AEF2F5F1F6E7
:0F008700E2C050A4E1D8F450A8F1E2F6E2E3F6AB
:0F009600B3F7E1748E50A8D1E150A850A8C0264E
:0F00A5000027C0263C77AA76AA263C77E076E013
:0F00B400C000002200200C21DFE1201021EAF41F
:0F00C3001CD0202021EAF41CD0D1E240E92E4D20
:0F00D200DAED7DD47ED47FD4201021EAF41CEE59
:0F00E100202021EAF414E940EE00000000A21420
:0F00F000B920C021D8E1200121D4E05130A31D57
:0F00FF002C20C021D4E1200121D3E05130A31CDB
:0F010E002C20C021D2E1200121D2E05130A31CCE
:0F011D002C000020C021D1E1200121D1E0513080
:0F012C00220000C0201021EAF4143FFC20002123
:0F013E00E0D1E3C0202021EAF4143EFC2004215F
:02014A0041383A
:0000000000

```

SAMPLE OUTPUT
FROM MPTA-5

ENTFY	WAS:	PCW	1,	CCLUMN	1
ENTFY	WAS:	PCW	2,	CCLUMN	1
ENTFY	WAS:	PCW	3,	CCLUMN	1
ENTFY	WAS:	PCW	4,	CCLUMN	1
ENTFY	WAS:	PCW	5,	CCLUMN	1
ENTFY	WAS:	PCW	6,	CCLUMN	1
ENTFY	WAS:	PCW	7,	CCLUMN	1
ENTFY	WAS:	PCW	8,	CCLUMN	1
ENTFY	WAS:	PCW	1,	CCLUMN	2
ENTFY	WAS:	PCW	2,	CCLUMN	2
ENTFY	WAS:	PCW	3,	CCLUMN	2
ENTFY	WAS:	PCW	4,	CCLUMN	2
ENTFY	WAS:	PCW	5,	CCLUMN	2
ENTFY	WAS:	PCW	6,	CCLUMN	2
ENTFY	WAS:	PCW	7,	CCLUMN	2
ENTFY	WAS:	PCW	8,	CCLUMN	2
ENTFY	WAS:	PCW	1,	CCLUMN	3
ENTFY	WAS:	PCW	2,	CCLUMN	3
ENTFY	WAS:	PCW	3,	CCLUMN	3
ENTFY	WAS:	PCW	4,	CCLUMN	3
ENTFY	WAS:	PCW	5,	CCLUMN	3
ENTFY	WAS:	PCW	6,	CCLUMN	3
ENTFY	WAS:	PCW	7,	CCLUMN	3
ENTFY	WAS:	PCW	8,	CCLUMN	3
ENTFY	WAS:	PCW	1,	CCLUMN	4
ENTFY	WAS:	PCW	2,	CCLUMN	4
ENTFY	WAS:	PCW	3,	CCLUMN	4
ENTFY	WAS:	PCW	4,	CCLUMN	4
ENTFY	WAS:	PCW	5,	CCLUMN	4
ENTFY	WAS:	PCW	6,	CCLUMN	4
ENTFY	WAS:	PCW	7,	CCLUMN	4
ENTFY	WAS:	PCW	8,	CCLUMN	4



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 40-13☐ 4004 ☒ 4040 ☐ 8008 ☐ 8080 ☐ 3000

(use additional sheets if necessary)

Program
Title

"SEL" SUBROUTINE (Selector)

Function

To recognize 5 individual character sequences up to four characters each and set appropriate flags (see accompanying diagram)

Required
Hardware

- 1, Device selector board
- 2, UART (serial data) card
- 3, Matrix card

Required
Software

This program is used in data communication control application, a simple sample program is enclosed but any program may call this routine.

Input
Parameters

Serial data is received by a UART and assembled in parallel format on ROM input ports 2 & 3

UART is controlled by ROM port 0

When sequence is recognized flags are set in SBO registers 4 and 6

Output
Results

			EOT
EOA	BC	RSC	TSC

← 4

← 6

Registers Modified: Register Bank 0, R8, R9, R10	Assembler/Compiler Used: Intellec 4 Version 3.0
RAM Required: CH0 REG 0 & 1 CHAR 0 → 7 2 & 3	Programmer: Laszlo I. Szerenyi
ROM Required: (Sample Program 0000 → 0050) SUBROUTINE 0061 → 00C2	Company: Frederick Electronics Co.
Maximum Subroutine Nesting Level: 1	Address: Box 502 Frederick Maryland 21701

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
 2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
 3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.
 4. A source paper tape of the contributed program is required. This insures that a clear, original copy of the program is available to photo-copy for publication in a User's Library update publication.
-

end completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

:1

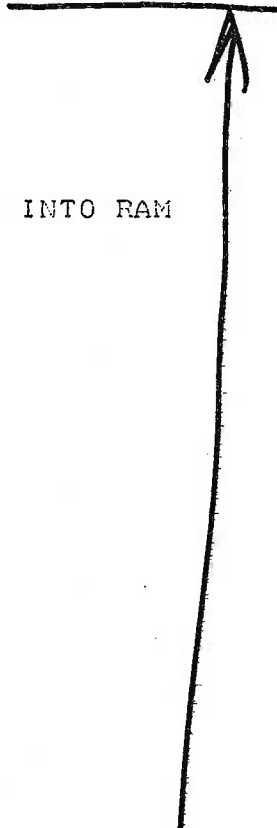
:2

*204C

```
;SELECTOR SUBROUTINE AND SAMPLE
;PROGRAM 8BIT PARALELLED DATA ON
;ROM IN/OUT PORTS 2&3,CONTROL
;OUTPUT ROM PORT 0
;DEVICE SELECT USES RPM INST-
;RUCTION TO GET THE "PM"PULSE
;FOR THE HARDWARE ONLY.THE DEVICE
;IS SPECIFIED BY DATA ON ROM
;OUTPUT PORT 2.CHARACTER MATRIX
;IS EXTERNAL DIODE PROGRAMMABLE
;FOR EASY FIELD CHANGE.
;THE COMPLETE CIRCUIT USES
;DEVICE CONTROLLER BOARD(WITH
;"PM"PULSE)UART BOARD=DS1,
;MATRIX BOARD=DS2
;OUTPUT BOARD=DS4
;TOTAL SYSTEM USES A NUMBER
;OF OTHER BOARDS ALSO.
;THE OBJECT OF THE "SEL
;SUBROUTINE" IS TO FIND 5
;INDIVIDUAL SEQUENCES OF UP
;TO FOUR CHARACTERS EACH,AN
;END OF SEQUENCE IS MARKED
;BY AN END OF SEQUENCE SIGNAL
;ON ROM IN PORT 0,BIT 0.
;THE PROGRAM IS ONLY AN
;INDICATION OF THE POSSIBLE
;USES OF THE SUBROUTINE.
```

```
0000 03 DE0
0001 0A SB0
0002 2000 FIM 0,0
0004 2A20 FIM 10,32 ;ROMP 2(3)
0006 D4 LDM 4
0007 B3 XCH 3
0008 29 SETRAM: SRC 8 ;SET MATRIX ADDR INTO RAM
; (CH0 ,R0,CH0-7)

0009 AB LD 11
000A E0 WRM
000B F1 CLC
000C 83 ADD 3
000D BB XCH 11
000E 68 INC 8 ;R1
000F A0 LD 0 ;R0 = 0
0010 29 SRC 8
0011 E0 WRM ;CLEAR RAM R1
0012 D0 LDM 0
0013 B3 XCH 8
0014 69 INC 9
0015 F1 CLC
0016 D3 LDM 3
0017 99 SUB 9 ;C=0 ALREADY
0018 1C08 JNZ SETRAM
001A B9 XCH 9 ;ACC=0,RESET R9
001B 29 RDATA: SRG 8 ;ROMP 0
```



001C	D7	LDM 7	;0111=CH1 CONTR
001D	E2	WRR	
001E	2B	SRC 10	;ROMP 2
001F	D1	LDM 1	
0020	E2	WRR	
0021	0E	RPM	;DS1=UART
0022	29	SRC 8	
0023	EA	RDR	;READ STAT
0024	F6	RAR	
0025	1A1B	JNC RDATA	;C=0=NO DP
0027	5061	JMS SEL	
0029	D7	LDM 7	
002A	29	SRC 8	
002B	E2	WRR	;ROMP 0=0111
002C	2B	SRC 10	
002D	D4	LDM 4	
002E	E2	WRR	
002F	0E	RPM	;DS 4=OUTPUT
0030	2810	FIM 8,16	
0032	29	READ: SRC 8	
0033	F1	CLC	
0034	E9	RDM	
0035	BB	XCH 11	;R11=FLAG
0036	F1	CLC	
0037	D3	LDM 3	;0-3=LSB (R9)
0038	99	SUB 9	
0039	AB	LD 11	;ACC =FLAG,C=SIGN
003A	1A43	JNC MSB	;C=0==MSB
003C	F1	CLC	
003D	F6	RAR	
003E	A6	LD 6	
003F	F6	RAR	
0040	B6	XCH 6	
0041	4047	JUN NEWF	
0043	F6	MSB: RAR	
0044	A4	LD 4	
0045	F6	RAR	
0046	B4	XCH 4	
0047	F1	NEWF: CLC	
0048	D7	LDM 7	
0049	99	SUB 9	
004A	144F	JZ OUTPUT	
004C	69	INC 9	
004D	4032	JUN READ	
004F	F0	OUTPUT: CLB	
0050	B8	XCH 8	;RESET R8
0051	2B	SRC 10	
0052	A6	LD 6	
0053	E2	WRR	;ROMP 2=LSB

Sample program to call
subroutine and light

LED's

0054	6A	INC 10	
0055	2B	SPC 10	
0056	A4	LD 4	
0057	E2	WRR	;ROMP 3 =MSB
0058	29	SRC 8	
0059	D4	LDM 4	
005A	E2	WRR	;STROBE OUTPUT
005B	D7	LDM 7	
005C	E2	WRR	;END STROBE
005D	D2	LDM 2	
005E	BA	XCH 10	;RESET R10
005F	401B	JUN RDATA	

;SEL SUBROUTINE

0061	2B	SEL:	SRC 10	
0062	EA		RDR	
0063	B1		XCH 1	
0064	6A		INC 10	;ROMP 3
0065	2B		SRC 10	
0066	EA		RDR	
0067	B0		XCH 0	;DATA IN R0,1
0068	F0		CLB	;CLEAR ROMPS-MSB MATPIX ADDR
0069	E2		WRR	
006A	D2		LDM 2	
006B	BA		XCH 10	
006C	29		SRC 8	
006D	D6		LDM 6	;0110=CH1 DRR
006E	E2		WRR	;START OF SELECT SEQ
006F	D7		LDM 7	
0070	E2		WRR	;DRR COMPL
0071	2B		SPC 10	;ROMP 2
0072	D2		LDM 2	
0073	E2		WRR	
0074	0E		RPM	;DS2=MATRIX
0075	F0		CLB	
0076	B9		XCH 9	;INIT RAM ADDR, LOOP COUNTERS
0077	29	LCOP:	SRC 8	;RAM CH0, R0
0078	E9		RDM	;CURRENT MATRIX ADDR
0079	2B		SRC 10	
007A	E2		WRR	;LSB ADDR
007B	B3		XCH 3	;TENP ADDR STORE
007C	F0		CLB	
007D	D3		LDM 3	
007E	99		SUB 9	;CHECK LOOP NO
007F	1287		JC COMP	
0081	D1		LDM 1	
0082	6A		INC 10	
0083	2B		SPC 10	;ROMP 3
0084	E2		WRR	
0085	D2		LDM 2	
0086	BA		XCH 10	;RESET R10
0087	2B	COMP:	SPC 10	;ADDR COMPL, PEAD DATA
0088	EA		RDR	
0089	F1		CLC	
008A	91		SUB 1	;COMPARE LSB
008B	1CB4		JNZ FAIL	

008D	6A		INC 10	
008E	2B		SRC 10	
008F	D2		LDM 2	
0090	BA		XCH 10	;RESET R10
0091	EA		RDR	
0092	F1		CLC	
0093	90		SUB 0	;COMPARE MSB
0094	1CB4		JNZ FAIL	
0096	29		SPC 8	
0097	EA		RDR	
0098	F6		RAR	;CHECK FOR LAST CHAR OF SEQ
0099	12A9		JC SEQ	;C=1 GO TO NEXT CHAR IN SEQ
009B	68		INC 8	;RAM CH0 R1
009C	29		SRC 8	
009D	D1		LDM 1	
009E	E0		WRM	;STORE FLAG
009F	F0		CLC	
00A0	B8		XCH 8	;RAM CH0 R0
00A1	29		SRC 8	
00A2	DC		LDM 12	;1100
00A3	B7		XCH 7	
00A4	A3		LD 3	;R3=MATRIX LSB ADDR
00A5	07		AN7	
00A6	E0		WRM	;INIT CHAR SEQ
00A7	40AC		JUN NEXT	;GET NEXT SEQ
00A9	63	SEQ:	INC 3	;NEXT MATRIX ADDR
00AA	A3		LD 3	;ADDR INTO ACC
00AB	E0		WRM	;STILL RAM CH0,R0
00AC	69	NEXT:	INC 9	
00AD	D4		LDM 4	;5 SEQUENCES ONLY
00AE	F1		CLC	
00AF	99		SUB 9	
00B0	1AC2		JNC GOUT	;C=0==END OF SEC
00B2	4077		JUN LOOP	
00B4	D3	FAIL:	LDM 3	;0011
00B5	B7		XCH 7	
00B6	A3		LD 3	
00B7	07		AN7	;CHECK IF FIRST CHAR OF SEQ
00B8	14AC		JZ NEXT	
00BA	DC		LDM 12	;1100
00BB	B7		XCH 7	
00BC	A3		LD 3	
00BD	07		AN7	;INITIALIZE MATRIX LSB ADDR
00BE	29		SRC 8	
00BF	E0		WRM	;STORE IN RAM
00C0	4077		JUN LOOP	;CHECK AGAIN
		*\$S		
		*?		
		*A		
00C2	C7	GOUT:	BEL 7	
			END	

:



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 40-14☒ 4004 ☐ 4040 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
Title

PAPER TAPE CONVERSION 5-LEVEL TELETYPE TO 8-LEVEL ASCII

Function

This program converts information (programs, etc.) originally on 5-level (TELEX on TWX) paper tape to ASCII 8-level paper tape. In this way programs can be sent over telex or TWX and then converted without retyping the information.

Required
Hardware

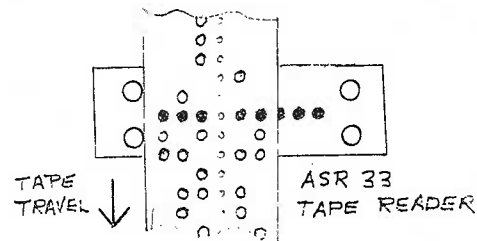
TTY on Port 0 and 1

Required
Software

None (modified TTY routines included in program)

Input
Parameters

The 5-level tape to be converted to ASCII is loaded in the paper tape reader of the teletype so that the tape covers the 5 leftmost pins as shown in illustration.

Output
Results

The program will convert the character and punch the equivalent ASCII character. No rubouts will appear on the ASCII tape but blanks will be copied.

NOTE: The source listing included was typed on a Model 19 5-level TTY and converted to ASCII before assembly.

Registers Modified: 0,1,6,7,8,9,15,4	Assembler/Compiler Used: ASSEMBLER VERSION 2.0
RAM Required: NONE	Programmer: Mark D. Hansen
ROM Required: 207 Bytes	Company: INSTRUMENTATION SPECIALTIES CO.
Maximum Subroutine Nesting Level: 1 level	Address: P.O. BOX 5347 LINCOLN, NEBRASKA 68505

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
 2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps is not necessary.
 3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.
-

Your program will be photo-copied for publication in the User's Library. Please send an original, clear, un-marked copy.

Send completed documentation to:

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051

:1

:2

```

; PAPER TAPE CONVERSION 5 TO 8 LEVEL
; BY MARK HANSEN 5-1-75
;

```

```

0000 D5          LDM 5    ; SET REG 15 FOR LTRS LOOKUP
0001 BF          XCH 15
0002 D1          MARK:  LDM 1    ; INPUT 5 LEVEL CHARACTER
0003 2640        TIX:    FIM 6, 64
0005 27          SRC 6
0006 E1          WMP
0007 D8          LDM 8
0008 B4          XCH 4
0009 507E        JMS SR0
000B EA          TIO:    RDR
000C F6          RAR
000D 1A0B        JCN 10, TIO AND 0FFH
000F 5088        JMS SBR1
0011 27          SRC 6
0012 E1          WMP
0013 29          SRC 8
0014 5082        TI1:    JMS SBR2
0016 EA          RDR
0017 F4          CMA
0018 F6          RAR
0019 A0          LD 0
001A F6          RAR
001B B0          XCH 0
001C A1          LD 1
001D F6          RAR
001E B1          XCH 1
001F 7414        ISZ 4, TI1 AND 0FFH
0021 5082        JMS SBR2
0023 5082        JMS SBR2
0025 5088        JMS SBR1
0027 A1          LD 1    ; REVERSE INPUT CHARACTER
0028 F6          RAR
0029 F7          TCC
002A B0          XCH 0
002B F6          RAR
002C F7          TCC
002D B3          XCH 3
002E A1          LD 1
002F F5          RAL
0030 F7          TCC
0031 F5          RAL
0032 83          ADD 3
0033 B3          XCH 3
0034 A1          LD 1
0035 F5          RAL
0036 F5          RAL
0037 F7          TCC
0038 F5          RAL
0039 F5          RAL
003A 83          ADD 3
003B B3          XCH 3

```

003C	A1		LD 1
003D	F6		RAR
003E	F6		RAR
003F	D0		LDM 0
0040	F6		RAR
0041	83		ADD 3
0042	B1		XCH 1
0043	A0		LD 0
0044	F6		RAR
0045	1267		JCN 2, TEST AND 0FFH ; IF MSB OF INPUT A ONE
0047	AF	HANSEN:	LD 15
0048	F5		RAL
0049	B0		XCH 0
004A	30		FIN 0 ; DO TABLE LOOKUP FOR ASCII CHAR
004B	507E		JMS SRO ; OUTPUT ASCII CHAR ROUTINE
004D	E1		WMP
004E	D8		LDM 8
004F	B4		XCH 4
0050	5082	TO1:	JMS SBR2
0052	F1		CLC
0053	B0		XCH 0
0054	F6		RAR
0055	B0		XCH 0
0056	B1		XCH 1
0057	F6		RAR
0058	B1		XCH 1
0059	F7		TCC
005A	E1		WMP
005B	7450		ISZ 4, TO1 AND 0FFH
005D	5082		JMS SBR2
005F	D1		LDM 1
0060	E1		WMP
0061	5082		JMS SBR2
0063	5082		JMS SBR2
0065	4002		JUN MARK
0067	A1	TEST:	LD 1 ; CHECK FOR FIGS OR LTRS
0068	F1		CLC
0069	F2		IAC
006A	1474		JCN 4, LTRS AND 0FFH
006C	F1		CLC
006D	D5		LDM 5
006E	81		ADD 1
006F	1479		JCN 4, FIGS AND 0FFH
0071	FA		STC
0072	4047		JUN HANSEN ; IF NOT LTRS OR FIGS PRINT
0074	FA	LTRS:	STC ; SET TO LOOKUP A0-BF
0075	D5		LDM 5
0076	BF		XCH 15
0077	4002		JUN MARK
0079	FA	FIGS:	STC ; SET TO LOOKUP C0-DF
007A	D6		LDM 6
007B	BF		XCH 15
007C	4002		JUN MARK
007E	2800	SRO:	FIM 8, 0
0080	29		SRC 8
0081	C0		BBL 0
0082	283C	SBR2:	FIM 8, 60
0084	7984	L2:	ISZ 9, L2 AND 0FFH

0086	7884		ISZ 8, L2 AND 0FFH
0088	283C	SBR1:	FIM 8, 60
008A	798A	L1:	ISZ 9, L1 AND 0FFH
008C	788A		ISZ 8, L1 AND 0FFH
008E	C0		BBL 0
00A0		ORG 0A0H	
00A0	00	DB 00H ;	BLANK
00A1	45	DB 45H ;	E
00A2	0A	DB 0AH ;	LF
00A3	41	DB 41H ;	A
00A4	20	DB 20H ;	SPACE
00A5	53	DB 53H ;	S
00A6	49	DB 49H ;	I
00A7	55	DB 55H ;	U
00A8	0D	DB 0DH ;	CR
00A9	44	DB 44H ;	D
00AA	52	DB 52H ;	R
00AB	4A	DB 4AH ;	J
00AC	4E	DB 4EH ;	N
00AD	46	DB 46H ;	F
00AE	43	DB 43H ;	C
00AF	4B	DB 4BH ;	K
00B0	54	DB 54H ;	T
00B1	5A	DB 5AH ;	Z
00B2	4C	DB 4CH ;	L
00B3	57	DB 57H ;	W
00B4	48	DB 48H ;	H
00B5	59	DB 59H ;	Y
00B6	50	DB 50H ;	P
00B7	51	DB 51H ;	Q
00B8	4F	DB 4FH ;	O
00B9	42	DB 42H ;	B
00BA	47	DB 47H ;	G
00BB	00	DB 00H ;	FIGS
00BC	4D	DB 4DH ;	M
00BD	58	DB 58H ;	X
00BE	56	DB 56H ;	V
00BF	00	DB 00H ;	LTRS
		;	FIGS CONVERSION TABLE
00C0	00	DB 00H ;	BLANK
00C1	33	DB 33H ;	3
00C2	0A	DB 0AH ;	LF
00C3	2D	DB 2DH ;	-
00C4	20	DB 20H ;	SPACE
00C5	07	DB 07H ;	BELL
00C6	38	DB 38H ;	8
00C7	37	DB 37H ;	7
00C8	0D	DB 0DH ;	CR
00C9	24	DB 24H ;	\$
00CA	34	DB 34H ;	4
00CB	27	DB 27H ;	'
00CC	2C	DB 2CH ;	,
00CD	21	DB 21H ;	!
00CE	3A	DB 3AH ;	:
00CF	28	DB 28H ;	(
00D0	35	DB 35H ;	5
00D1	22	DB 22H ;	"
00D2	29	DB 29H ;)
00D3	32	DB 32H ;	2
00D4	00	DB 00H ;	STOP

00D5	36	DB 36H ; 6
00D6	30	DB 30H ; 0
00D7	31	DB 31H ; 1
00D8	39	DB 39H ; 9
00D9	3F	DB 3FH ; ?
00DA	26	DB 26H ; &
00DB	00	DB 00H ; FIGS
00DC	2E	DB 2EH ; .
00DD	2F	DB 2FH ; /
00DE	3B	DB 3BH ; SEMICOLON
00DF	00	DB 00H ; LTRS

END

☒ 4004 ☐ 8008 ☐ 8080 ☒ 4040

(use additional sheets if necessary)

Program Title John Conway's Solitaire Game of "LIFE" 32x32 grid

Function Computes and displays successive generations of game area status. Accepts coordinates of points for the definition of initial conditions, or for altering patterns between generation cycles.

Required Hardware Break Key Status sensor; ASCII keyboard input; ASCII alphanumeric display output, minimum area: 34 lines of 66 characters.
Serial TTY may be used, input on ROM input port 0, bit 0; output on RAM output port 0, bit 0; 750 KHz clock for bit timing.

Required Software I/O drivers for Keyboard input and Display output.
Sample drivers for serial TTY included.

Input Parameters 3 Commands: / clears grid
 ; displays current grid status
 : begins generation cycle simulation
Grid points entered (or deleted) by input of two-character coordinates: RC,RC,... . Each coordinate consists of either a digit (0-9) or a letter (A-V), designating one of 32 rows or one of 32 columns in the grid.
All other input is ignored.

Output Results In response to ":" or ";" command, program outputs a line of column headings (= column coordinates) along the top, a column of row headings along the left margin, and asterisks (*) at the grid points which represent live cells. During generation cycle simulation, each new cycle begins automatically after completion of previous. Output display may be aborted by activating Break Status. Brief instructions display initially, at program startup.

Registers Modified: 0,1,2,8,9,10,11,12,13,14,15	Maximum Subroutine Nesting Level: MAX(2,0+IO)
RAM Required: 4 4002s, in one bank	Assembler/Compiler Used: "Hardware Assembler"
ROM Required: 3 Pages (646 bytes+ I/O)	Programmer: Tom Pittman
	Box 23189 San Jose, CA 95153

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only):
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer and company
2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps should **not** be included.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

JOHN CONWAY'S SOLITAIRE GAME OF "LIFE"

This is a stand-alone program designed to run on an MCS-4 or MCS-40 with three pages of program memory and one bank of 4002 RAM (320 4-bit digits), with some means of keyboard entry and visual display output (such as a printer), for the presentation of the state of the "universe".

The program accepts key inputs to set or clear any or all of the cells in the array, to display the current state of the array, and to start the simulation. The "universe" for this program is an array, 32 cells by 32 cells. These cells are addressed by row and column designations, where each row or column designator is either a decimal digit, or a letter between A and V, inclusive. The digit 0 (zero) designates the top row, or the leftmost column, 1 the second row or column; the letter A designates the eleventh row or column, and V is the rightmost column, or the bottom row. A cell is addressed by specifying its coordinates as a two character string, where the first character designates the row, and the second the column. Examples of valid coordinate pairs:

00 (left top corner) VV (bottom right corner)
A99A99AA (a "block")

Isolated letters and digits are not valid coordinates, and are ignored by the program. Each time the coordinates of a cell are keyed into the program, the state of that cell is reversed, so that a second entry of the same cell coordinates will delete an erroneous entry, and a third time will replace it again.

Three commands are recognized by this program:

- / This character clears the entire matrix in preparation for the entry of a new pattern.
- ; A semicolon initiates a display cycle without processing the data in the array. This permits the visual verification of the patterns keyed in, before beginning actual simulation.
- : A colon begins actual simulation. The array is scanned, and the next generation is computed during the display cycle. The array is updated at the same time as the new data printout, so that at any time, the most recent display output of any cell represents the current contents of that cell, unless altered by explicit input.

Any display output operation may be interrupted by an operator-controlled Break status. If this occurs during a heading typeout, the line is aborted, but the display continues. If the Break occurs during the array typeout, the display (and the update, if during simulation) is aborted, and the program returns to await new command input.

This program requires I/O drivers to interface to the keyboard input and the display output, as well as the sensing of the Break status. These drivers are called as subroutines from the main routine of this program, and may be nested to any depth appropriate to the processor. All of the drivers may use or alter the contents of any of the processor registers, with the following exceptions:

Registers 2, 8, 9, 10 should be preserved, or saved and restored.

Registers 14, 15 are used to hold input or output characters, with register 14 holding the most significant four bits. On input, ASCII characters keyed in should be stripped of their parity bit, so that bit 3 of register 14 is set to 0. Most (but not all) of the output data is encoded even parity by the program.

The following subroutines are required by this program:

TBRK This subroutine samples the Break Key status. On return, the Carry bit is set to one if the Break status is true, and zero otherwise.

KEYIN This routine accepts a one-character ASCII keyin, and returns with the 7-bit code right-justified in the register pair 14,15. If there is no keyin pending at the time of the call, the subroutine waits until there is.

PRINT This subroutine displays the character in the register pair 14,15 on the output device. Except for the initial instructions typeout which contains some carriage returns and linefeeds, all characters sent to this subroutine are either spaces or printable graphics.

SPACE This a special case entry to PRINT, which forces a space code (X'AO') into registers 14,15 first, so that a single space is printed by this call.

NL This subroutine causes the display to start a new line. It may be implemented by calls to the Print routine with a Carriage Return and a Linefeed, or by whatever special controls are appropriate to the display device being used.

Included in the program listing are sample I/O drivers for a bit-serial TTY on ROM Input and RAM Output Ports 0. These drivers were tested for operation on the SIM4 Prototyping system, and produced the sample output shown.

The Sample Run printout included in this packet shows some of the operational features of the program.

The first six lines of text are typed by the program, as a brief summary of the input syntax. All characters input which are not recognized as coordinates or commands are ignored by the program. This includes spaces and control characters, which may be inserted at will (but not between the row and column designators of a cell coordinates!) to improve readability.

The virgule cleared the playing area, then started a new line. The long string of co-ordinates places the initial pattern of the "Cheshire Cat". The second row of coordinates entered was an attempt to place a Glider; some errors were made in this string: Cell 2A was entered at first, then thought better of, and deleted by entering it again, and cell 1B is misplaced, as the subsequent listing shows. The semicolon initiated the display of the current status, showing the correctly entered Cheshire Cat, and the erroneous Glider.

When the display ended (it could have been terminated early by the Break key), the error is corrected in the Glider, by deleting cell 1B, and adding cell 1C, and the scene is displayed again. This time it is correct, so the simulation is started by typing a colon.

In this example the program proceeds through seven cycles of update/display without intervention. The eighth cycle is aborted during row M. At this time new data may be entered, or the the cycles continued, at the will of the operator.

Since this program was designed to run with a Teletype, which is a fairly slow device, several measures have been taken to minimize unnecessary printing time. On lines which have no cells to display, or which are blank to the right of some point in the line, printing is suppressed for the remainder of the line.

The operator may also abort the heading line by hitting the Break key after this line starts to print, then releasing the key for the printing of the matrix. When the pattern is wholly contained in the top part of the matrix, printing time may be further reduced by hitting the Break key after the pattern is printed, then typing a colon to continue with the next cycle from the top.

```

01:/ JOHN CONWAY'S GAME OF "LIFE" (32X32 MATRIX)
02:/ REVISED 1 MAY 1975
03:/ AUTHOR: T. PITTMAN, MICROCOMPUTER CONSULTANT
04:/
05:/ REQUIRES MCS-4/40, 4 4002 RAMS, I/O DRIVERS.
06:/
07:/ INITIALIZATION & PATTERN INPUT
08:/
09:/ TYPE OUT DIRECTIONS
10:START NOP
11:    FIM 0,TEXT        POINT TO TEXT STRING
12:    FIN 14            GET A BYTE
13:    LD 14             IS THIS END?
14:    IAC               =X'FF'
15:    JAZ CLEAR         YES.
16:    JMS PRINT         NO, PRINT IT.
17:    ISZ 1,START+3     GO BACK FOR MORE
18:    ISZ 0,START+3
19:/ CLEAR FIELD
20:CLEAR FIM 8,0
21:    CLB
22:    SRC 8
23:    WRM               STORE ZEROS AT ALL POINTS
24:    ISZ 8,*-2         LOOP THRU 256 DIGITS
25:    ISZ 9,*-4
26:    JMS NL            THEN TYPE NEW LINE
27:/ ACCEPT NEW POINTS: RC,RC,...
28:NEW    JMS KEYIN      GET TYPEIN
29:    FIM 0,58          IS THIS COLON?
30:    JMS CCH
31:    JAZ GO            YES; GO START.
32:    FIM 0,47          IS THIS SLASH?
33:    JMS CCH
34:    JAZ CLEAR         YES; GO CLEAR THE FIELD.
35:    FIM 0,59          SEMICOLON?
36:    JMS CCH
37:    JAZ SHOW          YES. DISPLAY CURRENT SCENE.
38:    JMS CNVRT         CONVERT CODE TO ADDRESS
39:    JNZ NEW           OH, NOT LETTER OR DIGIT
40:    LD 14             PICK OUT TOP 4 BITS
41:    RAR               FOR MEMORY COLUMN
42:    LD 15             (= DISPLAY ROW)
43:    RAR
44:    XCH 9
45:    TCC               HOLD REMAINING BIT FOR NOW
46:    XCH 8
47:    JMS KEYIN         GET SECOND CHARACTER
48:    JMS CNVRT         CONVERT TO 5-BIT BINARY
49:    JNZ NEW           NO GOOD. START THIS ONE OVER.
50:    LD 14             NOW SHIFT IN HELD ROW BIT
51:    ADD 8             FROM LEFT
52:    ADD 8

```

```

65:      RAR              (UPPER 2 BITS)
66:      XCH 15
67:      RAR              (LOWER 4)
68:      XCH 15
69:      RAR              (LOW BIT)
70:      XCH 15
71:      RAR              UPPER 4 BITS
72:      XCH 8            BECOME MEMORY ROW;
73:      LD 15
74:      RAL              LOW 2 BITS
75:      RAL
76:      FIM 10,12        BECOME COUNTERS
77:      XCH 11            TO POSITION THE BIT:
78:      ADD 11            ADD 12
79:      XCH 11
80:      LD 11
81:      XCH 10
82:      TURN BIT ON (OR OFF IF ALREADY ON) AT SELECTED POINT
83:      SRC 8            SELECT THE DIGIT
84:      RDM
85:      RAL              SHIFT BIT INTO CARRY
86:      ISZ 10,*-1
87:      CMC              COMPLEMENT IT
88:      RAR              SHIFT IT BACK
89:      ISZ 11,*-1
90:      WRM
91:      JUN NEW          GO GET ANOTHER.
92:      95:/
93:      INITIAL INSTRUCTIONS TEXT
94:      95:/
95:TEXT  141             (RETURN)
96:      10              (LINEFEED)
97:      160             (SPACE)
98:      160
99:      160
100:     160
101:     51              3
102:     178              2
103:     216              X
104:     51              3
105:     178              2
106:     160
107:     34              "
108:     204              L
109:     201              I
110:     198              F
111:     197              E
112:     34              "
113:     141
114:     10
115:     10
116:     175              /

```

117:	160	
118:	195	C
119:	204	L
120:	197	E
121:	65	A
122:	210	R
123:	160	
124:	85	U
125:	78	N
126:	201	I
127:	86	V
128:	197	E
129:	210	R
130:	83	S
131:	197	E
132:	141	
133:	10	
134:	187	,
135:	160	
136:	160	
137:	68	D
138:	201	I
139:	83	S
140:	80	P
141:	204	L
142:	65	A
143:	89	Y
144:	160	
145:	83	S
146:	212	T
147:	65	A
148:	212	T
149:	85	U
150:	83	S
151:	141	
152:	10	
153:	58	:
154:	160	
155:	160	
156:	66	B
157:	197	E
158:	71	G
159:	201	I
160:	78	N
161:	160	
162:	195	C
163:	89	Y
164:	195	C
165:	204	L
166:	197	E
167:	83	S
168:	141	

169:	10	
170:	60	<
171:	210	R
172:	195	C
173:	190	>
174:	160	
175:	160	
176:	65	A
177:	68	D
178:	68	D
179:	160	
180:	207	O
181:	210	R
182:	160	
183:	68	D
184:	197	E
185:	204	L
186:	197	E
187:	212	T
188:	197	E
189:	160	
190:	195	C
191:	197	E
192:	204	L
193:	204	L
194:	141	
195:	10	
196:	160	
197:	160	
198:	160	
199:	210	R
200:	172	,
201:	195	C
202:	160	
203:	189	=
204:	160	
205:	48	O
206:	45	-
207:	57	9
208:	172	,
209:	65	A
210:	45	-
211:	86	V
212:	141	
213:	10	
214:	255	(END OF TEXT)
215:/	OUTPUT STARTUP	
215:SHOW	XCH 2	SET NON-PROCESS FLAG
216:	ISZ 2,GO+1	THEN GO DISPLAY IT
218:GO	XCH 2	SET FLAG FOR PROCESS
219:	JMS NL	START WITH CLEAN LINE
221:	JMS SPACE	


```

223:      FIM 8,6          FIRST DECIMAL DIGITS,
225:      JUN CHD          GO PRINT COLUMN HEADERS
227:/
227:/  CHARACTER HANDLING SUBROUTINES
227:/
227:/  COMPARE X0-1 TO X14-15 (LEVEL 3)
227:CCH  CLC
228:      LD 14
229:      SUB 0
230:      JNZ BBC          NOT EQUAL IN LEFT 4 BITS.
232:      CLC
233:      LD 15
234:      SUB 1
235:      JNZ BBC          NOT EQUAL IN RIGHT 4 BITS
237:      BBL 0            EQUAL. CARRY=1
238:BBC  BBL 12           NOT EQUAL. CARRY=1 IF X14-15>X0-1
239:/  CONVERT ALPHA CHARACTER TO 5-BIT BINARY (LEVEL 2)
239:CNVRT FIM 0,87       IS THIS "W" OR GREATER?
241:      JMS CCH
243:      JOC BBC          YES. NOT ALLOWED.
245:      FIM 0,48        IS IT LESS THAN ZERO?
247:      JMS CCH
249:      JCZ BBC          ALSO NOT ALLOWED.
251:      FIM 0,58        IS IT A DIGIT?
253:      JMS CCH
255:      CMC              (IF SO, THIS IS ALL)
256:      JOC CND          YES.
258:      FIM 0,65        BE SURE IT IS A LETTER.
260:      JMS CCH
262:      JOC *+3
264:      BBL 12          NO.
265:      LDM 8            YES. BIAS TO 5-BIT NUMBER
266:      ADD 15           (+CARRY)
267:      XCH 15
268:CND  LDM 12
269:      ADD 14
270:      XCH 14
271:      CLC
272:      BBL 0
273:/
273:/  PATTERN PROCESS & DISPLAY
273:/
273:/  PRINT COLUMN HEADERS
273:CHD  JMS SPACE        SEPARATED BY SPACES.
275:      JMS TBRK        IF BREAK, ABORT TITLE
277:      JOC CHB
279:      FIM 14,48       ASCII ZERO,
281:      LD 8
282:      XCH 15           WITH DIGIT ADDED
283:      JMS PRINT
285:      INC 8            INCREMENT DIGIT.
286:      ISZ 9,CHD        COUNT TO TEN.

```

288:	FIM 8,65	NOW ALPHABET, A-V
290:CHA	JMS SPACE	
292:	JMS TBRK	IF BREAK, ABORT TITLE
294:	JOC CHB	
296:	LD 8	COPY CHARACTER
297:	XCH 14	
298:	LD 9	
299:	XCH 15	
300:	FIM 0,87	IS THIS END ("W")?
302:	JMS CCH	
304:	JOC CHX	YES.
306:	JMS PRINT	NO.
308:	ISZ 9,CHA	INCREMENT TO NEXT LETTER
310:	ISZ 8,CHA	(NEVER FALLS THRU)
312:CHB	JMS TBRK	WAIT FOR BREAK KEY RELEASE
314:	JOC CHB	
316:CHX	JMS NL	END OF LINE.
318:/	INITIALIZE SCAN COUNTER	
318:	FIM 8,0	256 DIGIT POSITIONS
320:	FIM 10,192	4 BITS IN EACH DIGIT
322:	JUN ROW	
324:/	PRINT ROW HEADER	
324:ROW	LD 8	GET HIGH 5 BITS
325:	RAL	
326:	LD 9	
327:	RAL	
328:	FIM 14,48	TRY FOR DECIMAL DIGIT
330:	DAA	(ADD 6 & TEST FOR >16)
331:	XCH 15	
332:	JCZ PRH	9 OR LESS.
334:	LD 9	NO, MUST BE LETTER.
335:	RAL	CHECK FOR <P
336:	JCZ **7	WHICH IS COUNT <26
338:	CLC	
339:	DAA	
340:	JCZ **3	
342:	INC 14	IF NOT, INCREMENT LETTER, 16.
343:	INC 14	ADD PROPER BIAS
344:	ISZ 15,**3	
346:	INC 14	
347:PRH	JMS PRINT	
349:	JMS SPACE	
351:	LD 2	IF THIS IS GENERATE CYCLE,
352:	JAZ GEN	GO DO IT.
354:	FIM 0,0	OTHERWISE SCAN THIS ROW,
356:	JMS DC8	BACKWARDS,
358:CSCAN	CLC	
359:	RDM	LOOKING FOR NON-TRIVIAL DATA
360:	ADD 1	(PROPAGATE "ZERO SUPPRESS")
361:	ADD 0	
362:	WR2	SET FLAG (0=END)
363:	XCH 1	(NOT 0 IF MORE TO RIGHT)

```

364:      JMS DC8
366:      JCZ CSCAN
368:      JUN TLINE-2      NOW GO PRINT IT
370:/    COMPUTE ONE ROW OF NEXT GENERATION
370:GEN   SRC 8
371:      FIM 0,0          FIRST CLEAR COUNTER.
373:      CLC              POINT TO NEXT ROW DOWN
374:      LDM 8
375:      ADD 8
376:      XCH 12           IN X12-13
377:      LDM 0
378:      ADD 9
379:      XCH 13
380:      RDM              COPY OLD VALUE HERE,
381:      JCZ **+3
383:      LDM 0            OR ZERO, IF BOTTOM ROW,
384:      SRC 12           TO STATUS DIGIT 0,
385:      WRO              OF THAT LINE.
386:      JOC BOTOM       IF NOT BOTTOM LINE,
388:      RDM              READ NEIGHBORS
389:      JMS BIT          (DIRECTLY BELOW)
391:      RDM              (DOWN & RIGHT)
392:      JMS RIGHT
394:      JCZ **+6
396:      JMS RBIT
398:      JMS NRT
400:      SRC 12
401:      RDM
402:      JMS LEFT         (DOWN & LEFT)
404:      JCZ **+4
406:      JMS LBIT
408:BOTOM JMS T8C         OOPS, WANT ADJACENT DIGIT
410:      RDO              NOW DO CURRENT ROW
411:      JMS BIT          AND PREVIOUS ROW.
413:      RDO              (CENTER UP)
414:      JMS RIGHT        (UP, RIGHT)
416:      JMS T8C          (RESTORE POINTER)
418:      RDO
419:      JMS LEFT         (UP, LEFT)
421:      JMS T8C          (RESTORE POINTER AGAIN)
423:      RDM
424:      JMS RIGHT        CELL TO THE RIGHT
426:      JCZ **+7
428:      JMS RBIT
430:      RDO              OH, NEXT DIGIT, LOW BIT...
431:      JMS RBIT+1        DO TOP ROW WHILE AT IT
433:      JMS T8C          RESTORE POINTER.
435:      RDM              NOW THE CELL TO THE LEFT.
436:      JMS LEFT
438:      JCZ **+7         LIKEWISE SYMMETRICALLY.
440:      JMS LBIT
442:      RDO

```

```

443:      JMS LBIT+1
445:      SRC 8
446:      CLC                NOW EVALUATE NEIGHBOR COUNT
447:      LDM 13
448:      ADD 0
449:      JAZ SET             3 MAKES BIRTH (CARRY SET)
451:      CMC
452:      JCZ SET            4 OR MORE MAKES DEATH
454:      IAC
455:      JCZ SET            1 OR LESS MAKES DEATH
457:      RDM                OTHERWISE STAY SAME
458:      JMS BIT
460:SET   RD1                SHIFT INTO STATUS 1
461:      RAR
462:      WRI
463:      ISZ 10,GEN          GO TO NEXT CELL
465:      LDM 12
466:      XCH 10
467:      JMS IN8
469:      JCZ GEN
471:      FIM 0,0             NOW SCAN THE ROW,
473:      JMS DC8             BACKWARDS,
475:BSCAN CLC
476:      RD1                LOOKING FOR NON-TRIVIAL DATA
477:      ADD 1                (PROPAGATE "ZERO SUPPRESS")
478:      ADD 0
479:      WR2                SET FLAG (0=END)
480:      XCH 1                (NOT 0 IF MORE TO RIGHT)
481:      JMS DC8
483:      JCZ BSCAN
485:      JUN TLINE-2          NOW GO TYPE IT
487:/
487:/  SUBROUTINES FOR MANIPULATING CELL POINTER (X8-9)
487:/
487:/  INCREMENT R8 CIRCULARLY (MOD 8) (LEVEL 3)
487:IN8   LD 8                SHIFT TOP BIT INTO CARRY.
488:      STC                SHIFT IN 1
489:      RAL
490:      XCH 8
491:      ISZ 8,++3            INCREMENT IT
493:      INC 8                OVERFLOW - INCREMENT CARRY OUT
494:      XCH 8
495:      RAR
496:      XCH 8
497:      SRC 8                SET NEW SELECT
498:      BBL 0                CARRY=1 IF WRAPAROUND
499:/  DECREMENT R8 CIRCULARLY (MOD 8) (LEVEL 3)
499:DC8   LD 8                CHECK FOR NEAR WRAPAROUND
500:      CLC
501:      RAL
502:      JNZ ++3              NAW.
504:      CMC                YES. PRECOMPLEMENT HIGH BIT

```

```

505:      RAR
506:      DAC
507:      XCH 8
508:      CLC          NOW, IF THIS IS WRAPAROUND,
509:      RAL
510:      DAC          LEAVE CARRY
511:      CMC          SET TO 1.
512:      SRC 8
513:      BBL 0
514:/ COPY X8-9 TO X12-13 (LEVEL 3)
514:T8C LD 8
515:      XCH 12
516:      LD 9
517:      XCH 13
518:      SRC 8      ALSO SELECT THAT ADDRESS
519:      BBL 0
520:/
520:/ MATRIX TYPEOUT
520:/
520:/ TYPE A LINE, UPDATING MATRIX.
520:      JMS IN8
522:TLINE JMS TBRK    CHECK FOR BREAK
524:      JCZ TBIT    NO. CONTINUE.
526:      JMS NL      YES. STOP UPDATE/DISPLAY
528:      JMS TBRK    WAIT FOR BREAK RELEASE
530:      JOC *-2
532:      JUN NEW     THEN GO ACCEPT NEW INPUT
534:TBIT SRC 8        GET NEW VALUE
535:      LD 2
536:      JAZ **4      IF GENERATING NEW CYCLE,
538:      RDM          OR OLD VALUE,
539:      16          (=SKP) IF NOT.
540:      RDI          (4 BITS)
541:      WRM          COPY TO MATRIX
542:      JMS BIT      SELECT A BIT
544:      FIM 14,160   TYPE SPACE IF ZERO,
546:      JCZ **4
548:      FIM 14 170   ASTERISK IF ONE.
550:      RD2          ANY THING LEFT?
551:      JAZ **6      IF NOT, DONT PRINT.
553:      JMS PRINT
555:      JMS SPACE
557:      ISZ 10,TBIT   DO 4 BITS.
559:      LDM 12
560:      XCH 10
561:      JMS IN8      ADVANCE TO NEXT DIGIT
563:      JCZ TLINE
565:      JMS NL      NOW PROCEED TO NEXT ROW
567:      JMS DC8
569:      ISZ 8,JROW
571:      ISZ 9,JROW
573:      LD 2      IF JUST DISPLAY,

```

```

574:      JNZ TLINE+4      GO ACCEPT ANOTHER COMMAND;
576:      JUN GO           ELSE DO ANOTHER ITERATION
578:JROW  JUN ROW
580:/
580:/  SUBROUTINES FOR BIT SELECTION
580:/
580:/  PICK BIT FROM CELL TO RIGHT (LEVEL 3)
580:RIGHT XCH 11          (SAVE A)
581:      LD 10            GET COUNT
582:      IAC             BUMP IT
583:      JOC 112          OOPS, WE ARE ON BOUNDARY.
585:RLB  XCH 11           OK, LIKE BIT...
586:      RAL
587:      ISZ 11,*-1
589:      JCZ BBO
591:      INC 0
592:      CLC              ... BUT LEAVE CARRY 0.
593:BBO  BBL 0
594:112  INC 12           INCREMENT DIGIT ADDRESS,
595:      LD 12           AND CHECK FOR RIGHT EDGE.
596:      CLC
597:      RAL
598:      JAZ NRT          YES.
600:LRS  SRC 12          OK. SET SELECT LOGIC.
601:      STC
602:      BBL 0
603:NRT  LD 12           BACK UP FROM ADVANCE
604:      DAC
605:      XCH 12
606:      CLC
607:      BBL 0
608:/  PICK A BIT FROM CELL TO LEFT (LEVEL 3)
608:LEFT XCH 11          (LIKE RIGHT, SYMMETRICALLY)
609:      LDM 4
610:      CLC
611:      ADD 10
612:      JAZ D12
614:      LD 10
615:      DAC
616:      JUN RLB
618:D12  LD 12           DECREMENT DIGIT ADDRESS.
619:      DAC
620:      XCH 12
621:      CLC
622:      RAL
623:      JNZ LRS          GO RETURN
625:      INC 12          IF AT MARGN. RESTORE,
626:      CLC             AND QUIT
627:      BBL 0
628:/  COUNT LEFT-MOST OR RIGHT-MOST BIT (LEVEL 3)
628:LBIT RDM
629:      RAL

```

```
630:      JUN CBIT
632:RBIT  RDM
633:      RAR
634:      JUN CBIT
636:/    PICK CURRENT BIT (LEVEL 3)
636:BIT   XCH 11          (SAVE A)
637:      LD  10          GET BIT POSITION COUNT
638:      XCH 11          USE X11 AS COUNTER
639:      RAL             SHIFT BIT INTO CARRY
640:      ISZ 11,*-1      (BIT 0 GOES 4 PLACES MAX)
642:CBIT  JCZ **+3
644:      INC 0
645:      BBL 0
646:/
```

```

646:/ SAMPLE I/O ROUTINES FOR TTY ON PORTS 0 (BIT SERIAL)
646:/
646:NL      FIM 14,141      (CARRIAGE RETURN)
648:        JMS PRINT
650:        FIM 14,10      (LINEFEED)
652:        JUN PRINT
654:SPACE FIM 14,160
656:PRINT LDM 6            SET COUNTER FOR TEN BITS
657:        XCH 11
658:        FIM 12,128     WAIT 1.5 STOP BIT TIMES
660:        JMS WAIT
662:        JMS WAIT
664:        JMS SRO        SELECT RAM PORT 0
666:WTT     TCC            PULL DATUM OUT OF CARRY,
667:        WMP            OUTPUT IT
668:        JMS WAIT       WAIT ONE BIT TIME
670:        STC            SHIFT IN ONES FOR STOP BITS,
671:        JMS S2R        GO SHIFT RIGHT 1 BIT,
673:        ISZ 11,WTT     RECYCLE,
675:        FIM 14,0       CLEAR DATA BYTE TO 0,
677:/ SELECT BANK 0 RAM 0 (LEVEL 3)
677:SRO     FIM 12,0
679:        CLB
680:        DCL
681:        SRC 12
682:        BBL 0          CARRY ZERO
683:KEYIN JMS SRO
685:        FIM 14,127     INITIALIZE DATA BYTE,
687:TSB     RDR            GET TTY LINE INPUT,
688:        RAR            MOVE TO CARRY FOR TEST,
689:        JCZ TSB        CARRY IS START BIT
691:        FIM 12,128     PRESET COUNTER TO 4 MS,
693:GNB     JMS WAIT       WAIT FOR MIDDLE OF BIT,
695:        RDR            READ TTY BIT,
696:        CMA            (IT IS INVERTED)
697:        WMP            ECHO IT BACK,
698:        RAR            DATUM TO CARRY AND SHIFT IT,
699:        JMS S2R        CARRY IS NOW ZERO ON LAST BIT,
701:        JOC GNB        AND ONE IF MORE,
703:        JMS S2R        POSITION THE BYTE
705:        JMS WAIT       WAIT FOR PARITY BIT
707:        LDM 1          NOW SET STOP (OR IDLE) BITS,
708:        WMP            THEN DELAY TO MIDDLE OF STOP BIT.
709:/ 9 MS DELAY SUBROUTINE, ASSUMING 0 IN 12,13.
709:WAIT    NOP            32.4 US INNER LOOP CYCLE.
710:        ISZ 13,WAIT     INNER LOOP TOTAL 518.4US.
712:        NOP            EXTEND OUTER LOOP CYCLE
713:        NOP            TO 562US,
714:        ISZ 12,WAIT     ASSUMING 10.8 CYCLE TIME.
716:        BBL 12          TOTAL 9.08 MS,
717:/ SHIFT INDEXES 14,15 RIGHT ONE BIT (LEVEL 3)

```


717:S2R	LD 14	
718:	RAR	
719:	XCH 14	
720:	LD 15	
721:	RAR	
722:	XCH 15	
723:	BBL 0	CARRY OUT OF THE SHIFT.
724:/	TEST FOR TTY BREAK	KEY (LEVEL 2)
724:TBRK	JMS SRO	TEST FOR TTY BREAK KEY
726:	RDR	WHICH IS SPACE CONDITION
727:	RAR	
728:	BBL 0	CARRY = 1 IF BREAK
729:\$		

32X32 "LIFE"

```

/ CLEAR UNIVERSE
; DISPLAY STATUS
; BEGIN CYCLES
<RC> ADD OR DELETE CELL
  R,C = 0-9,A-V

```

```

/
  55 65 66 67 68 58 79 74 84 89 86 87 99 94 A5 A6 A7 A8
  2A 3B 3C 3D 2D 1B 2A
;
0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V

```

1B 1C

SECTION 7
UNCLASSIFIED PROGRAMS

REFERENCE NUMBER	PROGRAM
4-17	Random Number Generator
4-18	General Purpose ROM
4-19	Translate HEX
4-20	4x8 Keyboard Scanner

intel[®] 4-BIT USERS PROGRAM LIBRARY

JUNE 1975

TABLE OF CONTENTS

SECTION 1	INTRODUCTION
SECTION 2	STANDARD SOFTWARE
SECTION 3	MEMBERSHIP SUBMITTAL INFORMATION
SECTION 4	PROGRAMS: OPERATING, TESTING, DEBUGGING
SECTION 5	PROGRAMS: MATH AND NUMERICAL MANIPULATION
SECTION 6	CROSS PRODUCT SOFTWARE
SECTION 7	PROGRAMS: UNCLASSIFIED

SUBROUTINE KBD Scans a TOUCHTONE^R Keyboard Switch Matrix

```

KBD:  LDM      3      ;
      DCL      ;designate 4211 GPI/O on CM-RAM 3
      FIM      0,0COH ;port 0 for SRC 0
      FIM      2,0EOH ;port 2 for SRC 2
X1:   CLB      ;clear ACC and carry
      LDA      1      ;load ACC with 0001B - start of column scan
      SRC      0      ;select output port 0
      JUN      X3      ;jump to write first strobe in port 0
X2:   SRC      0      ;select output port 0
      RDR      ;read port content into ACC
      RAL      ;shift 1 space left
      JC       X1      ;if carry set, begin new scan
X3:   WRR      ;write ACC to output port 0 for column strobe
      SRC      2      ;select input port 2
      RDR      ;port content to ACC
      JZ       X2      ;try next strobe (falls thru on detection)
      KBP      ;process to obtain row
      CMA      ;complement ACC
      JZ       X6      ;error (multiple keyhit?) return thru X6
      CMA      ;restore ACC (content 1,2,3 or 4)
      DAC      ;decrement to 0,1,2 or 3
      XCH      15      ;place in REG 15
      LD       15      ;back in ACC
      RAL      ;times 2
      ADD      15      ;plus 1 (effectively times 3)
      XCH      15      ;REG 15 content is 0,3,6 or 9 (switch row dependent)
      SRC      0      ;select output port 0
      RDR      ;to ACC (column scan indicator)
      KBP      ;column value (1,2 or 3) in ACC
      ADD      15      ;final value (almost) in ACC
      XCH      15      ;in register
      LDM      15      ;load ACC with 1111B
      WRR      ;all 1's to output port 0
      SRC      2      ;select input port 2
X4:   RDR      ;port 2 to ACC
      JNZ      X4      ;keep checking until switch is released
      LDM      5      ;load 0101B in ACC
      ADD      15      ;ACC=0 if key 0 pushed (key 0 puts 'B' in REG 15)
      JZ       X5      ;REG 15 changes from 'B' to "0" at X5
      BBL      0      ;return with ACC=0
X5:   XCH      15      ;place '0' in REG 15
      BBL      0      ;return with ACC=0
X6:   CMA      ;ACC=1111B
      XCH      15      ;'F' placed in REG 15
      BBL      15      ;error return with 'F' in ACC and REG 15

```

Dr. P. J. Ferrell
6021 S. 119th St.
Seattle, WA 98178



MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

Ref. No. 4-12☒ 4004 ☒ 4040 ☐ 8008 ☐ 8080

(use additional sheets if necessary)

Program
TitleSubroutine to scan a TOUCHTONE^R keyboard. "KBD"

Function

When called, this subroutine awaits a key action by scanning a 3 x 4 switch array. The key value is calculated and placed in Register 15. Return is delayed until the key is released. The subroutine returns with Register 15 and the Accumulator given by the following table:

Pushbutton Symbol	Register 15 (HEX)	Accumulator (HEX)
0	0	0
1	1	0
2	2	0
3	3	0
4	4	0
5	5	0
6	6	0
7	7	0
8	8	0
9	9	0
*	A	0
#	C	0
Error:	F	F

This subroutine requires 53 bytes of program memory as coded, and may be reduced to 47 bytes if delayed return (wait til key release) is deleted.

Required
Hardware

An input and an output port are required. For the purpose of illustration, a type 4211 GP I/O chip is assumed connected to CM-RAM3. Port 0 is used as output and scans switch "columns", while port 2 is used as input and reads switch "rows".

Required
Software

A matrix keyboard (ITT, Chromerics, etc.) is required. No other software routines are required for this subroutine.

Registers Modified: Register 15	Assembler/Compiler Used:
RAM Required None	Programmer: Dr. P. J. Ferrell
ROM Required: 53 Bytes Maximum	Company: Boeing Aerospace Co.
Maximum Subroutine Nesting Level:	Address:

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
 - a. Processor (check appropriate box)
 - b. Program title: Name or brief description of program function
 - c. Function: Detailed description of operations performed by the program
 - d. Required hardware:
For example: TTY on port 0 and 1
Interrupt circuitry
I/O Interface
Machine line and configuration for cross products
 - e. Required software:
For example: TTY routine
Floating point package
Support software required for cross products
 - f. Input parameters: Description of register values, memory areas or values accepted from input ports
 - g. Output results: Values to be expected in registers, memory areas or on output ports
 - h. Program details (for resident products only)
 1. Registers modified
 2. RAM required (bytes)
 3. ROM required (bytes)
 4. Maximum subroutine nesting level
 - i. Assembler/Compiler Used:
For example: PL/M
Intellec 8 Macro Assembler
IBM 370 Fortran IV
 - j. Programmer, company and address
2. A source listing of the program must be included. This should be the output listing of a compile or assembly, Extra information such as symbol table or code dumps is not necessary.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

Your program will be photo-copied for publication in the User's Library. Please send an original, clear, un-marked copy.

Send completed documentation to.

Intel Corporation
User's Library
Microcomputer Systems
3065 Bowers Avenue
Santa Clara, California 95051