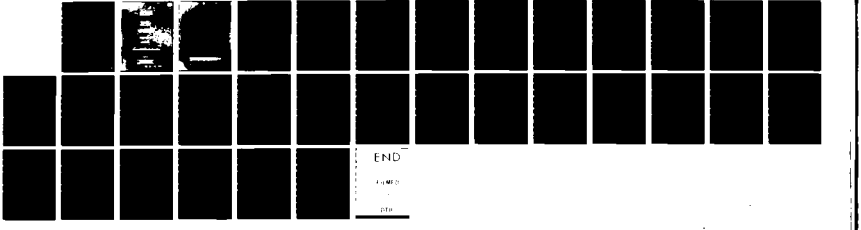
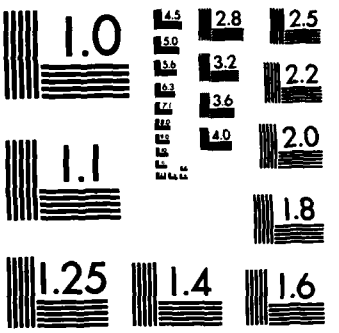
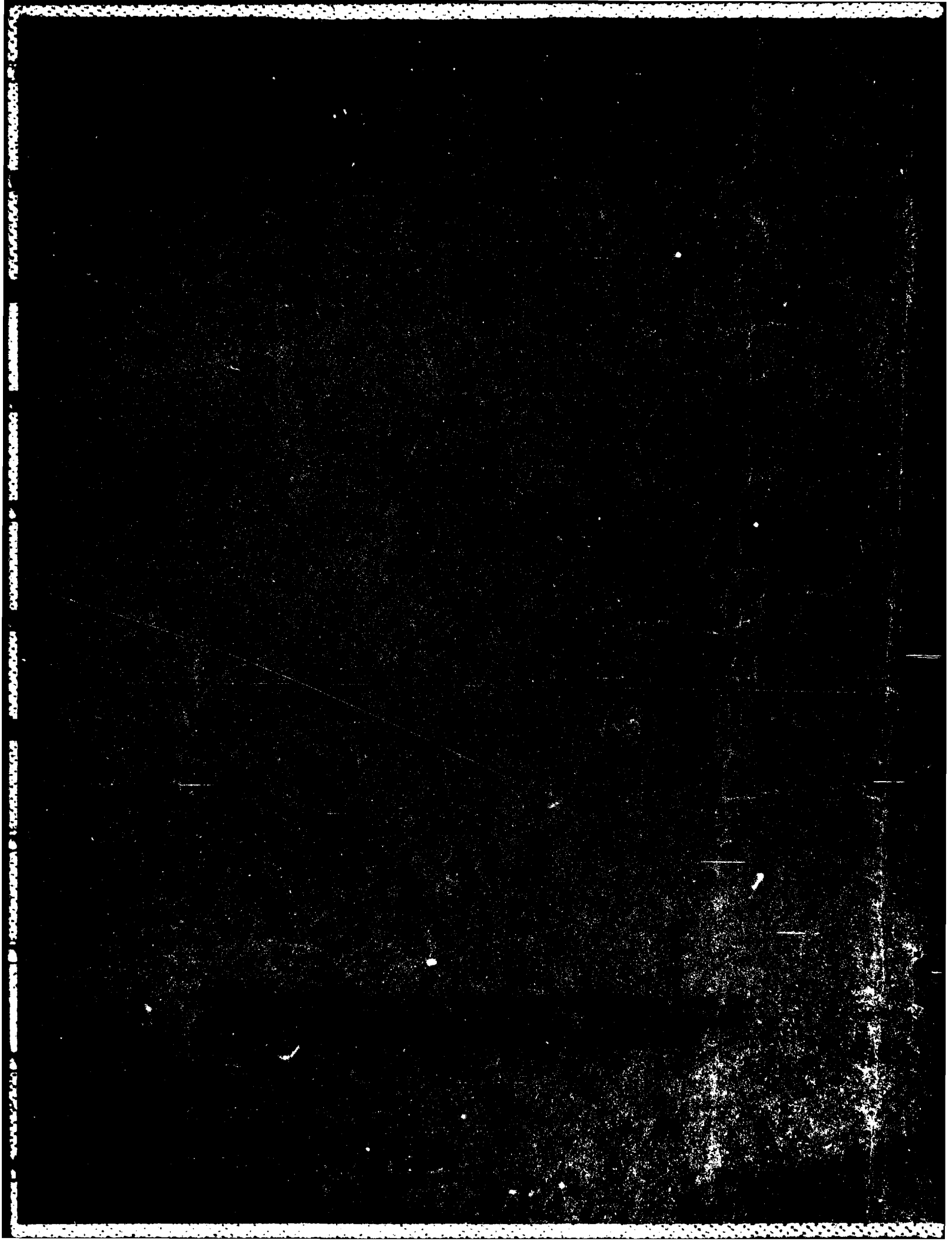
AD-A123 378 A VAX ASSEMBLER FOR THE INTEL 8748 MICROCOMPUTER(U) 1/1
NORTHEASTERN UNIV BOSTON MASS ELECTRONICS RESEARCH LAB
J R MANLEY ET AL. 14 OCT 82 SCIENTIFIC-1
UNCLASSIFIED AFGL-TR-82-0306 F19628-80-C-0050 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



MIL-STD-847A
31 January 1973

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFGL-TR-82-0306	2. GOVT ACCESSION NO. AD-A123378	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A VAX Assembler for the Intel 8748 Microcomputer		5. TYPE OF REPORT & PERIOD COVERED Scientific ~~Report No.~~ 1 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James R. Manley, Jr. J. Spencer Rochefort Thomas P. Wheeler		8. CONTRACT OR GRANT NUMBER(s) F19628-80-C-0050
9. PERFORMING ORGANIZATION NAME AND ADDRESS Northeastern University Electronics Research Laboratory Boston, MA 02115		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62101F 765904AS
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Geophysics Laboratory Hanscom AFB, MA 01731 Monitor/Raymond E. Wilton/LCR		12. REPORT DATE October 14, 1982 13. NUMBER OF PAGES 37
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Intel 8748 Assembler, BIME Encoder, VAX MACRO Assembler, Rocket Encoder		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report details the program developed to use the VAX 11/780 Operating System to assemble programs written for the Intel 8748 microcomputer. Input programs are written in Intel MCS-48 mnemonics and the output listing gives the instructions and memory addresses in hexidecimal code. The assembly language itself is written in MACRO		

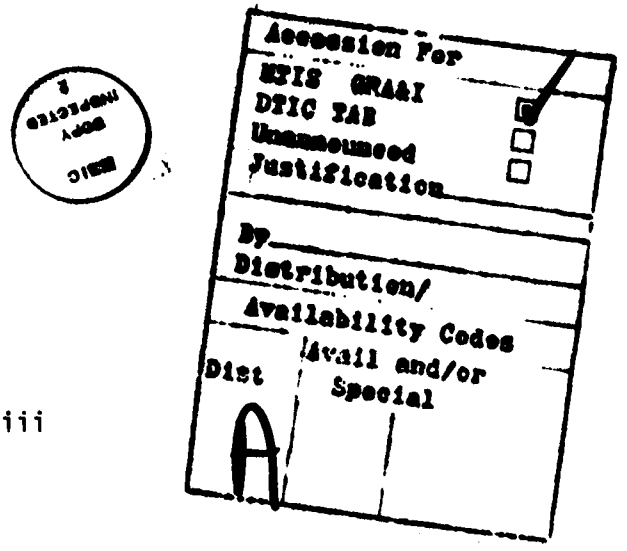
DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Page

TABLE OF CONTENTS



1. BACKGROUND

1.1 Software Programmable Airborne Encoder

As part of the research and development effort carried out under this contract the unique characteristics of microprocessors have been combined with PCM encoders to produce a versatile data acquisition system which can be programmed to perform the tasks required for a specific mission. These encoders employ the Intel 8748 microcomputer as a controlling unit. The software control is provided by means of a program which is placed in the EPROM memory of the 8748. The program is used to set the sampling rate at input ports, the amplitude resolution via bits per word, the number of words per frame, the use of parity, the synchronization code, the transmission rate and other suitable parameters. A basic version of these programmable encoders was developed for the BIME rocket program, and two were successfully flown on rockets A20.123-1 and A20.123-2 from the Natal Rocket Range, Brazil in September 1982.

1.2 The Microcomputer

The Intel 8748 was chosen for this application. It is classified as a single component, 8 bit microcomputer and has a user programmable/erasable EPROM. A device to write the program into the EPROM memory was available but it required as its input the hexidecimal codes for both the memory addresses and also the program instructions which were to be written into memory. Consequently any user was confronted with the task of first writing the program using Intel's MCS-48 mnemonics and keeping track of memory locations, and then translating the mnemonics and the memory locations into hexidecimal codes. This process was very

time consuming and error prone and an assembler for the 8748 appeared to be the answer.

2. THE 8748 ASSEMBLER

2.1 VAX Usage

The Northeastern University VAX 11/780 Operating System has an extensive conditional macro assembler. This macro assembler and parts of the VAX assembler were used to construct the 8748 assembler. Use of this VAX facility enabled the new assembler to employ such utilities as linking, extensive macro processing, debugging and the like. Furthermore, the powerful Digital Command Language, DCL, could be employed to minimize the detailed knowledge of the VAX system required on the part of the user.

The design of the assembler was undertaken as a senior project by James Manley under the sponsorship of this contract and under the supervision of Professor J.S. Rochefort. Its first test was to assemble the 8748 program which was written by Mr. Thomas Wheeler for the BIME rocket program.

2.2 Procedure for Assembler Usage

The occasional user of the 8748 assembler should be able to use one of the VAX editors and be able to program using the mnemonics listed in Intel's MCS-48 user's manual.

The file should be named

"File Name".MAR

and should contain the program which is to be assembled in Intel's format.

The last entry into the file is

.END

and is placed in the op code field.

Do not use 5 periods (. . . . .) or the dollar sign ($) in any symbol names as they are reserved for the VAX and will cause confusion.

All indirect accesses using registers RØ and R1 should be written in the form

_RØ instead of @RØ

_R1 instead of @R1

since @ is used by VAX for a shifting function.

All constants should be equated to symbols and then these symbols should be used in the operand field. If constants are used in the operand field then they must be expressed as a base ten number, otherwise an error will be made in assembly.

Example: 1 Both of following will assemble correctly

```
ADD A, 15     CONST = 15
              ADD A, CONST
```

Example: 2 The left column will assemble incorrectly

```
ADD A, ^X 15     CONST = ^X 15
                 ADD A, CONST
```

The step-by-step procedure to use the assembler becomes

1. Edit a file "File Name".MAR using Intel mnemonics with the exception noted above. Close the file with .END .
2. Enter the DCL command

   @ASM8748

   This DCL program will do all the proper file manipulations

and executions. The resulting file, named "File Name".LIS, will be created containing the object code and errors.

3. If errors exist use this new file to re-edit the original program in "File Name".MAR and then go to step 2.

4. When errors no longer exist a hard copy can be obtained using the command

PRINT "File Name".LIS;n .

2.3 The Assembly Program

The DCL program is shown in Appendix A and is designated as ASM8748.COM;1. A programmer familiar with DCL can break this file down into smaller modules to save time under some circumstances or adapt the program to other systems.

The assembler program is shown in Appendix B and is designated by SPAREASM.MAR;1. As written the assembler handles the instructions currently used by the 8748. If new instructions are developed for future modifications of the 8748 by Intel then a programmer with macro experience can write an update.

3. APPLICATION TO BIME

3.1 The Program for Assembly

The first three pages of the program written for the 8748 microcomputer contained in the BIME encoder are shown in Appendix C and designated as BIME.MAR;1.

The bulk of the first page of the printout is devoted to defining constants by the symbols used in the program. The program instructions begin with the line DIS I. Comments are found to the right of the semicolons.

3.2 The Assembled Program

The first three pages of the assembled program for the BIME 8748 microcomputer are shown in Appendix D and designated as BIME.LIS;1. This printout should be compared against that of BIME.MAR;1.

The majority of the first page of this appendix is concerned with the constants. The lines from the input program (BIME.MAR;1) can be easily recognized to the right of the column of zeroes. The 8 column printing to the extreme left contains the hexidecimal equivalent for each decimal constant.

Once the program itself is encountered it should be noted that 2 or 3 lines are allotted to each of the original instructions. The first line associated with each instruction repeats the instruction from BIME.MAR and shows the address of the assigned EPROM cell as a four digit hexidecimal number. The second line (and third line where needed) give the hexidecimal code for the instruction as a 2-digit number followed by the 4-digit address. The following examples serve to illustrate this format.

Example A

Instruction SEL RB0 is assigned to memory address 0030 (from first line), and C5 (the op code for SEL RB0) is to be put into this cell.

Example B

Instruction MOV A, W1 is assigned memory address 0006 and 0007. Into the first address 23 is to be put, and into address 0007 is to be put 82 (it should be noted that 82 is the hexidecimal value of the constant W1).

Example C

The program line S1: JNT1 S1 is assembled so that 46 is put into memory address 0039 and 39 is put into address 003A (the 39' is to be considered the same as 39).

APPENDIX A

ASM8748.COM;1

```
 10  $WRITE SYS$OUTPUT "Enter file to be assembled,"
 20  $WRITE SYS$OUTPUT "   do not put in .MAR;n part of filename."
 30  $WRITE SYS$OUTPUT "System will assume .MAR;n at end of filename."
 40  $INQUIRE P1 ENTER FILE SPEC
 50  $WRITE SYS$OUTPUT "YOU HAVE BEGUN THE ASSEMBLY OF ''P1'.MAR"
 70  $ON WARNING THEN CONTINUE
 80  $APPEND/NEW SPAREASM.MAR.'P1'.MAR A.MAR
 85  $ON WARNING THEN CONTINUE
 90  $MAC/LIS='P1'.LIS/NOOBJ/NOSHOW=(CND,MD,ME)/SHOW=(MC,MEB)/DISABLE=
     GLOBAL A.MAR
100  $DEL A.MAR;*
```

APPENDIX B

SPARASH.MAR;1

```
        .MACRO CONSTANT
R0....=1
R1....=2
R2....=3
R3....=4
R4....=5
R5....=6
R6....=7
R7....=8
_R0....=9
_R1....=10
A....=11
P1....=12
P2....=13
BUS....=14
_A....=15
C....=16
F0....=17
F1....=18
PSW....=19
T....=20
CNT....=21
TCNT....=22
TCNTI....=23
RB0....=24
RB1....=25
MB0....=26
MB1....=27
CLK....=28
I....=29
P4....=30
P5....=31
P6....=32
P7....=33
        .ENDM   CONSTANT
        CONSTANT
        .MACRO  MAC1    MARG1,MARG2
        .IF NOT_BLANK.MARG1
                                .BYTE   <MARG2>
                                .BYTE   <MARG1>
        .MEXIT
        .ENDC
        .WARN   ;MISSING JUMP ADDRESS
        .ENDM   MAC1
        .MACRO  MAC2    MARG1,MARG2,MARG3
        .IF DEFINED MARG1'....
        .IF DEFINED MARG2'....
        .IF EQUAL MARG1'.... - A....
        .IF EQUAL MARG2'.... - _A....
                                .BYTE   <MARG3>
        .MEXIT
        .ENDC
        .WARN   ;INVALID SECOND ARGUMENT
        .MEXIT
```

```
          .ENDC
          .WARN    ;INVALID FIRST ARGUEMENT
          .MEXIT
          .ENDC
          .ENDC
          .WARN; INVALID PARAMETER
          .ENDM    MAC2
          .MACRO   MAC3     MARG1,MARG2
          .IF DEFINED MARG1'....
          .IF EQUAL MARG1'.... - A....
                                    .BYTE    <MARG2>
          .MEXIT
          .ENDC
          .ENDC
          .WARN    ;INVALID ARGUEMENT
          .ENDC
          .ENDM    MAC3
          .MACRO   MAC4     MARG1,MARG2
         ..IF NOT_DEFINED MARG1'....
                                    .BYTE    <<<<MARG1@-3>& ^X0E0>!MARG2>>
                                    .BYTE    <MARG1>
          .MEXIT
          .ENDC
          .WARN    ;INVALID OR MISSING ADDRESS
          .ENDM    MAC4
          .MACRO   MAC5     MARG1,MARG2
          .IF DEFINED MARG1'....
          .IF EQUAL MARG1'.... - P4....
                                    .BYTE <MARG2>
            .MEXIT
          .ENDC
          .IF EQUAL MARG1'.... - P5....
                                    .BYTE <MARG2 ! 01>
            .MEXIT
          .ENDC
          .IF EQUAL MARG1'.... - P6....
                                    .BYTE <MARG2 ! 02>
            .MEXIT
          .ENDC
          .IF EQUAL MARG1'.... - P7....
                                    .BYTE <MARG2 ! 03>
            .MEXIT
          .ENDC
          .ENDC
          .WARN;INVALID PORT NAME
          .ENDM
          .MACRO   MAC6 MARG1,MARG2
          .IF DEFINED MARG1'....
ERROR=1
          .IF EQUAL MARG1'.... - _R0....
                                    .BYTE    <MARG2>
            .MEXIT
          .ENDC
          .IF EQUAL MARG1'.... - _R1....
```

```
                                        .BYTE <MARG2 ! 01>
          .MEXIT
        .ENDC
        .ENDC
ERROR=0
        .ENDM
        .MACRO  MAC7    MARG1,MARG2,MARG3,MARG4,MARG5
        .IF NOT_DEFINED MARG1'....
          .WARN;MISSING PARAMETER
          .MEXIT
        .ENDC
        .IF EQUAL MARG1'.... - A....
                                        .BYTE <MARG2>
          .MEXIT
        .ENDC
        .IF EQUAL MARG1'.... - C....
                                        .BYTE <MARG3>
          .MEXIT
        .ENDC
        .IF EQUAL MARG1'.... - F1....
                                        .BYTE <MARG4>
          .MEXIT
        .ENDC
        .IF EQUAL MARG1'.... - F0....
                                        .BYTE <MARG5>
          .MEXIT
        .ENDC
        .WARN;INVALID PARAMETER
        .ENDM
        .MACRO  MAC8 MARG1
        .IF DEFINED MARG1'....
        .IF EQUAL MARG1'.... - R0....
REG=0
          .MEXIT
        .ENDC
        .IF EQUAL MARG1'.... - R1....
REG=1
          .MEXIT
        .ENDC
        .IF EQUAL MARG1'.... - R2....
REG=2
          .MEXIT
        .ENDC
        .IF EQUAL MARG1'.... - R3....
REG=3
          .MEXIT
        .ENDC
        .IF EQUAL MARG1'.... - R4....
REG=4
          .MEXIT
        .ENDC
        .IF EQUAL MARG1'.... - R5....
REG=5
          .MEXIT
        .ENDC
```

```
          .IF EQUAL MARG1'.... - R6....
REG=6
            .MEXIT
          .ENDC
          .IF EQUAL MARG1'.... - R7....
REG=7
            .MEXIT
          .ENDC
          .ENDC
REG=9
          .ENDM
          .MACRO MAC9 MARG1,MARG2,MARG3,MARG4,MARG5
          .IF NOT_DEFINED MARG1'....
            .WARN;MISSING FIRST PARAMETER
            .MEXIT
          .ENDC
          .IF EQUAL MARG1'.... - A....
            MAC6 MARG2,MARG3
            .IF EQUAL ERROR
              MAC8 MARG2
              .IF EQUAL REG - 9
                                    .BYTE MARG4
                                    .BYTE MARG2
                .MEXIT
              .ENDC
                                    .BYTE <MARG5 ! REG>
              .MEXIT
            .ENDC
            .MEXIT
          .ENDC
          .WARN;FIRST ARGUEMENT MUST BE AN A
          .ENDM   MAC9
          .MACRO  MAC10 MARG1,MARG2,MARG3,MARG4,MARG5,MARG6
          .IF EQUAL MARG1'.... - A....
            MAC8 MARG2
            .IF EQUAL REG - 9
              MAC6 MARG2,MARG4
              .IF EQUAL ERROR
                                    .BYTE MARG5
                                    .BYTE MARG2
                .MEXIT
              .ENDC
              .MEXIT
            .ENDC
                                    .BYTE <<MARG3> ! REG>
            .MEXIT
          .ENDC
          .IF EQUAL MARG1'.... - BUS....
                                    .BYTE MARG6
                                    .BYTE MARG2
            .MEXIT
          .ENDC
          .IF EQUAL MARG1'.... - P1....
                                    .BYTE <MARG6 ! 01>
```

```
                                        .BYTE MARG2
            .MEXIT
          .ENDC
          .IF EQUAL MARG1'.... - P2....
                                        .BYTE <MARG6 ! 02>
                                        .BYTE MARG2
            .MEXIT
          .ENDC
          .WARN;INVALID PARAMETER
          .ENDM   MAC10
          .MACRO  MAC11   MARG1,MARG2,MARG3,MARG4,MARG5
          .IF DEFINED MARG1'....
ERROR2=1
          .IF EQUAL MARG1'.... - PSW....
                                        .BYTE MARG5
            .MEXIT
          .ENDC
          .IF EQUAL MARG1'.... - T....
                                        .BYTE MARG4
            .MEXIT
          .ENDC
          MAC6 MARG1,MARG3
          .IF EQUAL ERROR
            MAC8 MARG1
            .IF EQUAL REG - 9
ERROR2=0
              .MEXIT
            .ENDC
                                        .BYTE <MARG2 ! REG>
          .ENDC
          .MEXIT
          .ENDC
ERROR2=0
          .ENDM   MAC11
          .MACRO  DA      ARG1=A
          MAC3    ARG1,87
          .ENDM   DA
          .MACRO  SWAP    ARG1=A
          MAC3    ARG1,71
          .ENDM   SWAP
          .MACRO  RL      ARG1=A
          MAC3    ARG1,231
          .ENDM   RL
          .MACRO  RLC     ARG1=A
          MAC3    ARG1,247
          .ENDM   RLC
          .MACRO  RR      ARG1=A
          MAC3    ARG1,119
          .ENDM   RR
          .MACRO  RRC     ARG1=A
          MAC3    ARG1,103
          .ENDM   RRC
          .MACRO  INS     ARG1=A,ARG2=BUS
          .IF EQUAL ARG1'.... - A....
          .IF_TRUE
```

```
.IF EQUAL ARG2'.... - BUS....
.IF_TRUE
                              .BYTE   ^X08
.IF_FALSE
.WARN   ;INVALID SECOND ARGUEMENT
.MEXIT
.ENDC
.IF_FALSE
.WARN   ;INVALID FIRST ARGUEMENT
.MEXIT
.ENDC
.ENDM   INS
.MACRO  JMP     ARG1
MAC4    ARG1,04
.ENDM   JMP
.MACRO  JMPP    ARG1=_A
.IF EQUAL ARG1'.... - _A....
.IF_TRUE
                              .BYTE   ^XB3
.IF_FALSE
.WARN   ;INVALID ARGUEMENT
.ENDC
.ENDM   JMPP
.MACRO  JC      ARG1
MAC1    ARG1,246
.ENDM   JC
.MACRO  JNC     ARG1
MAC1    ARG1,230
.ENDM   JNC
.MACRO  JNZ     ARG1
MAC1    ARG1,150
.ENDM   JNZ
.MACRO  JZ      ARG1
MAC1    ARG1,198
.ENDM
.MACRO  JT0     ARG1
MAC1    ARG1,54
.ENDM   JT0
.MACRO  JNT0    ARG1
MAC1    ARG1,38
.ENDM   JNT0
.MACRO  JT1     ARG1
MAC1    ARG1,86
.ENDM   JT1
.MACRO  JNT1    ARG1
MAC1    ARG1,70
.ENDM   JNT1
.MACRO  JF0     ARG1
MAC1    ARG1,182
.ENDM   JF0
.MACRO  JF1     ARG1
MAC1    ARG1,118
.ENDM   JF1
.MACRO  JTF     ARG1
```

```
MAC1    ARG1,22
.ENDM   JTF
.MACRO  JNI     ARG1
MAC1    ARG1,134
.ENDM   JNI
.MACRO  CALL    ARG1
MAC4    ARG1,20
.ENDM   CALL
.MACRO  RET
                        .BYTE   ^X83
.ENDM   RET
.MACRO  RETR
                        .BYTE   ^X93
.ENDM   RETR
.MACRO  MOVP    ARG1=A,ARG2=_A
MAC2    ARG1,ARG2,163
.ENDM   MOVP
.MACRO  MOVP3   ARG1=A,ARG2=_A
MAC2    ARG1,ARG2,227
.ENDM   MOVP3
.MACRO  STOP    ARG1=TCNT
.IF EQUAL ARG1'.... - TCNT....
.IF_TRUE
                        .BYTE   ^X65
.IF_FALSE
.WARN   ;INVALID ARGUEMENT
.ENDC
.ENDM   STOP
.MACRO  ENTO    ARG1=CLK
.IF EQUAL ARG1'.... - CLK....
.IF_TRUE
                        .BYTE   ^X75
.IF_FALSE
.WARN   ;INVALID ARGUEMENT
.ENDC
.ENDM   ENTO
.MACRO  NOP
                        .BYTE   ^X00
.ENDM   NOP
.MACRO  IN      ARG1=A,ARG2
.IF EQUAL ARG1'.... - A....
  .IF EQUAL ARG2'.... - P1....
                        .BYTE 09
    .MEXIT
  .ENDC
  .IF EQUAL ARG2'.... - P2....
                        .BYTE 10
    .MEXIT
  .ENDC
  .WARN;ILLEGAL PORT NUMBER
  .MEXIT
.ENDC
.WARN;INPUT CAN ONLY BE A
.ENDM   IN
.MACRO  OUTL    ARG1,ARG2=2
```

```
.IF EQUAL ARG2'.... - A....
  .IF EQUAL ARG1'.... - P1....
                        .BYTE ^X39
    .MEXIT
  .ENDC
  .IF EQUAL ARG1'.... - P2....
                        .BYTE ^X3A
    .MEXIT
  .ENDC
  .IF EQUAL ARG1'.... - BUS....
                        .BYTE ^X02
    .MEXIT
  .ENDC
  .WARN;INVALID FIRST ARGUEMENT
  .MEXIT
.ENDC
.WARN;SECOND ARGUEMENT MUST BE AN A
.ENDM   OUTL
.MACRO  MOVD    ARG1=A,ARG2=A
.IF EQUAL ARG1'.... - A....
  MAC5 ARG2,12
  .MEXIT
.ENDC
.IF EQUAL ARG2'.... - A....
  MAC5 ARG1,60
.ENDC
.ENDM   MOVD
.MACRO  ANLD    ARG1,ARG2=A
.IF EQUAL ARG2'.... - A....
  MAC5 ARG1,156
  .MEXIT
.ENDC
.WARN;SECOND ARGUEMENT MUST BE AN A
.ENDM   ANLD
.MACRO  ORLD    ARG1,ARG2=A
.IF EQUAL ARG2'.... - A....
  MAC5 ARG1,140
  .MEXIT
.ENDC
.WARN; SECOND ARGUEMENT NOT AN A
.ENDM   ORLD
.MACRO  XCHD    ARG1=A,ARG2
.IF EQUAL ARG1'.... - A....
  MAC6  ARG2,48
  .IF EQUAL     ERROR
    .WARN; INVALID INDIRECT REGISTER NUMBER
  .ENDC
.MEXIT
.ENDC
.WARN; FIRST ARGUEMENT MUST BE AN A
.ENDM   XCHD
.MACRO  MOVX    ARG1=A,ARG2=A
  .IF DEFINED ARG1'....
```

```
  .IF DEFINED ARG2'....
.IF EQUAL ARG1'.... - A....
MAC6    ARG2,128
.IF EQUAL ERROR
  .WARN; INVALID SECOND ARGUEMENT
.ENDC
  .MEXIT
.ENDC
.IF EQUAL ARG2'.... - A....
  MAC6  ARG1,144
  .IF EQUAL ERROR
    .WARN; INVALID FIRST ARGUEMENT
  .ENDC
  .MEXIT
.ENDC
.ENDC
.ENDC
.WARN; ONE OF THE ARGUEMENTS MUST BE AN A
.ENDM   MOVX
.MACRO  STRT    ARG1
.IF BLANK       ARG1
  .WARN;MISSING ARGUEMENT
  .MEXIT
.ENDC
.IF EQUAL       ARG1'.... - CNT....
                        .BYTE  ^X45
  .MEXIT
.ENDC
.IF EQUAL       ARG1'.... - T....
                        .BYTE  ^X55
  .MEXIT
.ENDC
.WARN; INVALID PARAMETER
.ENDM   STRT
.MACRO  EN      ARG1
.IF BLANK ARG1
  .WARN; MISSING ARGUEMENT
  .MEXIT
.ENDC
.IF EQUAL ARG1'.... - I....
                        .BYTE ^X05
  .MEXIT
.ENDC
.IF EQUAL ARG1'.... - TCNTI....
                        .BYTE ^X25
  .MEXIT
.ENDC
.WARN; INVALID ARGUEMENT
.ENDM   EN
.MACRO  DIS     ARG1
.IF BLANK ARG1
  .WARN; MISSING ARGUEMENT
  .MEXIT
.ENDC
.IF EQUAL ARG1'.... - I....
```

```
                              .BYTE ^X15
  .MEXIT
.ENDC
.IF EQUAL ARG1'.... - TCNTI....
                              .BYTE ^X35
  .MEXIT
.ENDC
.WARN; INVALID ARGUEMENT
.ENDM    DIS
.MACRO   SEL      ARG1
.IF BLANK ARG1
  .WARN; MISSING ARGUEMENT
  .MEXIT
.ENDC
.IF EQUAL ARG1'.... - MB0....
                              .BYTE ^X0E5
  .MEXIT
.ENDC
.IF EQUAL ARG1'.... - MB1....
                              .BYTE ^X0F5
  .MEXIT
.ENDC
.IF EQUAL ARG1'.... - RB0....
                              .BYTE ^X0C5
  .MEXIT
.ENDC
.IF EQUAL ARG1'.... - RB1....
                              .BYTE ^X0D5
  .MEXIT
.ENDC
.WARN; INVALID PARAMETER
.ENDM    SEL
.MACRO   CLR      ARG1
MAC7     ARG1,39,151,165,133
.ENDM    CLR
.MACRO   CPL      ARG1
MAC7     ARG1,55,167,181,149
.ENDM    CPL
.MACRO   DJNZ     ARG1,ARG2
.IF BLANK ARG1
  .WARN; MISSING REGISTER NUMBER
  .MEXIT
.ENDC
.IF BLANK ARG2
  .WARN; MISSING ADDRESS
  .MEXIT
.ENDC
MAC8 ARG1
.IF EQUAL REG - 9
  .WARN; INVALID REGISTER
  .MEXIT
.ENDC
                              .BYTE <^X0E8 ! REG>
                              .BYTE <ARG2>
```

```
.ENDM   DJNZ
.MACRO  JB      ARG1,ARG2
.IF BLANK ARG1
  .WARN; MISSING BIT NUMBER
  .MEXIT
.ENDC
.IF BLANK ARG2
  .WARN; MISSING JUMP ADDRESS
  .MEXIT
.ENDC
.IF GREATER ARG1 + 1
  .IF LESS_THAN ARG1 - 8
                        .BYTE <ARG1@5 ! 18>
                        .BYTE <ARG2>
    .MEXIT
  .ENDC
  .WARN;BIT NUMBER NEGATIVE
  .MEXIT
.ENDC
.WARN; BIT NUMBER GREATER THAN 7
.ENDM   JB
.MACRO  ADD     ARG1=A,ARG2
MAC9    ARG1,ARG2,96,03,104
.ENDM   ADD
.MACRO  ADDC    ARG1=A,ARG2
MAC9    ARG1,ARG2,112,19,120
.ENDM   ADDC
.MACRO  XRL     ARG1,ARG2
MAC9    ARG1,ARG2,208,211,216
.ENDM   XRL
.MACRO  XCH     ARG1=A,ARG2
.IF BLANK       ARG2
  .WARN; MISSING SECOND ARGUEMENT
  .MEXIT
.ENDC
.IF EQUAL ARG1'.... - A....
  MAC6 ARG2,32
  .IF EQUAL ERROR
    MAC8 ARG2
    .IF EQUAL REG - 9
      .WARN; INVALID SECOND ARGUEMENT
      .MEXIT
    .ENDC
                        .BYTE <^X20 ! REG>
    .MEXIT
  .ENDC
.MEXIT
.ENDC
.WARN; FIRST ARGUEMENT MUST BE AN A
.ENDM   XCH
.MACRO  DEC     ARG1=A
.IF EQUAL ARG1'.... - A....
                        .BYTE ^X07
  .MEXIT
.ENDC
```

```
MAC8 ARG1
.IF NOT_EQUAL REG - 9
                              .BYTE <^X0C8 ! REG>
  .MEXIT
.ENDC
.WARN; INVALID ARGUEMENT
.ENDM   DEC
.MACRO  INC     ARG1
.IF EQUAL ARG1'.... - A....
                              .BYTE ^X17
  .MEXIT
.ENDC
MAC6 ARG1,16
.IF EQUAL ERROR
  MAC8 ARG1
  .IF EQUAL REG - 9
    .WARN; INVALID ARGUEMENT
    .MEXIT
  .ENDC
                              .BYTE <^X18 ! REG>
.ENDC
.ENDM   INC
.MACRO  ANL     ARG1=A,ARG2
MAC10   ARG1,ARG2,88,80,83,152
.ENDM   ANL
.MACRO  ORL     ARG1=A,ARG2
MAC10   ARG1,ARG2,72,64,67,136
.ENDM   ORL
.MACRO  MOV     ARG1,ARG2
.IF DEFINED ARG1'....
.IF EQUAL ARG1'.... - A....
  MAC11 ARG2,248,240,66,199
  .IF EQUAL ERROR2
                              .BYTE ^X23
                              .BYTE ARG2
    .MEXIT
  .ENDC
  .MEXIT
.ENDC
.ENDC
.IF DEFINED ARG2'....
.IF EQUAL ARG2'.... - A....
  MAC11 ARG1,168,160,98,215
  .IF EQUAL ERROR2
    .WARN;INVALID PARAMETER
  .ENDC
  .MEXIT
.ENDC
.ENDC
  .IF NOT_DEFINED ARG1'....
    .WARN; INVALID PARAMETER
    .MEXIT
  .ENDC
MAC8 ARG1
```

```
.IF EQUAL REG - 9
  .IF EQUAL ARG1'.... - _R0....
                        .BYTE ^X0B0
                        .BYTE ARG2
    .MEXIT
  .ENDC
  .IF EQUAL ARG1'.... - _R1....
                        .BYTE ^X0B1
                        .BYTE ARG2
    .MEXIT
  .ENDC
  .WARN; INVALID FIRST ARGUEMENT
  .MEXIT
.ENDC
                        .BYTE <^X0B8 ! REG>
                        .BYTE ARG2
.ENDM   MOV
```

```
A1=00
A2=01
A3=02
A4=03
A5=04
B1=08
B2=09
B3=10
B4=11
B5=12
B6=13
B7=14
C2=18
C3=19
C4=20
C5=21
C6=22
C7=16
C8=17
C9=64
C70=112
SC=00    ;SPARE CHANNEL
KI=17
KP=16    ;COUNTER POINTER=16
P1=02    ;POINTER 1=02
P2=08    ;POINTER 2=08
P3=09    ;POINTER 3=09
P4=00    ;POINTER 4=00
P5=01    ;POINTER 5=01
W1=130   ;SET WORD LENGTH=(8+1)=9BITS AND SET OUTPUT CODE=Bi0-L
W2=00    ;SET PARITY=NO PARITY
W3=40    ;SET BIT RATE=11.9439*10E6/2(1+1)(1+3)=221.184KHz
W4=15    ;SET OUTPUT FILTER=XXXKHz
W5=16    ;SET PGA=2
W6=01    ;SYNC PATTERN
W7=230
W8=01
W9=64
W10=254  ;SYNC PATTERN/BAR
W11=25
W12=254
W13=191
         DIS      I
         ENTO     CLK      ;ENABLE CLOCK
         SEL      MB0
         SEL      RB0
         MOV      R0,P1    ;PORT 2
         MOV      A,W1     ;SELECT WORD LENGTH = (8+1) = 9 BITS AND
         MOVX     _R0,A    ;SELECT OUTPUT CODE = Bi0-L
         INC      R0       ;PORT 3
         MOV      A,W2     ;SELECT PARITY = NO PARITY
         MOVX     _R0,A
```

```
        INC     R0      ;PORT 4
        MOV     A,W3    ;SELECT BIT RATE = 222.222KBITS
        MOVX    _R0,A
        INC     R0      ;PORT 5
        MOV     A,W4    ;SELECT OUTPUT FILTER = XXXKHz
        MOVX    _R0,A
        MOV     A,W5    ;SELECT PGA = 2
        OUTL    P2,A
        MOV     R0,P2   ;SET POINTERS FOR SYNC PORTS
        MOV     R1,P3
        MOV     R2,W6   ;SYNC PATTERN
        MOV     R3,W7
        MOV     R4,W8
        MOV     R5,W9
        SEL     RB1
        MOV     R0,P4   ;SET POINTERS FOR SYNC PORTS
        MOV     R1,P5
        MOV     R2,W10  ;SYNC PATTERN/BAR
        MOV     R3,W11
        MOV     R4,W12
        MOV     R5,W13
        SEL     RB0
        MOV     R6,KI   ;SET INITIAL CONDITIONS ON POINTER-COUNTER
        CLR     A       ;AND FLAG
        MOV     _R0,A
        CLR     F0
        CPL     F0
LOOP:   MOVX    A,_R0
S1:     JNT1    S1
        MOV     A,R4
        MOVX    _R1,A
        MOV     A,R5
        MOVX    _R0,A
        MOV     A,B1    ;B1
        OUTL    P1,A
        MOVX    A,_R0
S2:     JNT1    S2
        SEL     RB0
        DJNZ    R6,PTA
        MOV     R6,KP
        CLR     F0
PTA:    MOV     A,B2    ;B2
        OUTL    P1,A
        MOVX    A,_R0
S3:     JNT1    S3
        JF0     PTB
        CLR     A
        MOV     _R0,A
PTB:    MOV     A,B3    ;B3
        OUTL    P1,A
        MOVX    A,_R0
S4:     JNT1    S4
        MOV     A,B4    ;B4
        OUTL    P1,A
```

```
        MOVX    A,_R0
S5:     JNT1    S5
        MOV     A,A1        ;A1
        OUTL    P1,A
        MOVX    A,_R0
S6:     JNT1    S6
        MOV     A,A2        ;A2
        OUTL    P1,A
        MOVX    A,_R0
S7:     JNT1    S7
        MOV     A,B1        ;B1
        OUTL    P1,A
        MOVX    A,_R0
S8:     JNT1    S8
        MOV     A,B2        ;B2
        OUTL    P1,A
        MOVX    A,_R0
S9:     JNT1    S9
        MOV     A,B3        ;B3
        OUTL    P1,A
        MOVX    A,_R0
S10:    JNT1    S10
        MOV     A,B4        ;B4
        OUTL    P1,A
        MOVX    A,_R0
S11:    JNT1    S11
        MOV     A,A3        ;A3
        OUTL    P1,A
        MOVX    A,_R0
S12:    JNT1    S12
        MOV     A,C70       ;ID
        OUTL    P1,A
        MOVX    A,_R0
S13:    JNT1    S13
        CLR     A
        MOVX    _R1,A
        MOV     A,_R0
        MOVX    _R0,A
        MOV     A,B1        ;B1
        OUTL    P1,A
        MOVX    A,_R0
S14:    JNT1    S14
        INC     _R0
        MOV     A,B2        ;B2
        OUTL    P1,A
        MOVX    A,_R0
S15:    JNT1    S15
        MOV     A,B3        ;B3
        OUTL    P1,A
        MOVX    A,_R0
S16:    JNT1    S16
        MOV     A,B4        ;B4
        OUTL    P1,A
```

APPENDIX D

BIME.LIS:1

```
                0000    177
00000000        0000      0 A1=00
00000001        0000      0 A2=01
00000002        0000      0 A3=02
00000003        0000      0 A4=03
00000004        0000      0 A5=04
00000008        0000      0 B1=08
00000009        0000      0 B2=09
0000000A        0000      0 B3=10
0000000B        0000      0 B4=11
0000000C        0000      0 B5=12
0000000D        0000      0 B6=13
0000000E        0000      0 B7=14
00000012        0000      0 C2=18
00000013        0000      0 C3=19
00000014        0000      0 C4=20
00000015        0000      0 C5=21
00000016        0000      0 C6=22
00000010        0000      0 C7=16
00000011        0000      0 C8=17
00000040        0000      0 C9=64
00000070        0000      0 C70=112
00000000        0000      0 SC=00
00000011        0000      0 KI=17
00000010        0000      0 KP=16
00000002        0000      0 P1=02
00000008        0000      0 P2=08
00000009        0000      0 P3=09
00000000        0000      0 P4=00
00000001        0000      0 P5=01
00000082        0000      0 W1=130
00000000        0000      0 W2=00
00000028        0000      0 W3=40
0000000F        0000      0 W4=15
00000010        0000      0 W5=16
00000001        0000      0 W6=01
000000E6        0000      0 W7=230
00000001        0000      0 W8=01
00000040        0000      0 W9=64
000000FE        0000      0 W10=254
00000019        0000      0 W11=25
000000FE        0000      0 W12=254
000000BF        0000      0 W13=191
                0000      0           DIS     I
             15 0000
                0001      0           ENTO    CLK
             75 0001
                0002      0           SEL     MBO
             E5 0002
```

```
     0003     0        SEL     RB0      } EXAMPLE A
C5   0003
     0004     0        MOV     R0,P1
B8   0004
02   0005
     0006     0        MOV     A,W1     } EXAMPLE B
23   0006
82   0007
```

```
     0008     0        MOVX    _R0,A
90   0008
     0009     0        INC     R0
18   0009
     000A     0        MOV     A,W2
23   000A
00   000B
     000C     0        MOVX    _R0,A
90   000C
     000D     0        INC     R0
18   000D
     000E     0        MOV     A,W3
23   000E
28   000F
     0010     0        MOVX    _R0,A
90   0010
     0011     0        INC     R0
18   0011
     0012     0        MOV     A,W4
23   0012
0F   0013
     0014     0        MOVX    _R0,A
90   0014
     0015     0        MOV     A,W5
23   0015
10   0016
     0017     0        OUTL    P2,A
3A   0017
     0018     0        MOV     R0,P2
B8   0018
08   0019
     001A     0        MOV     R1,P3
B9   001A
09   001B
     001C     0        MOV     R2,W6
BA   001C
01   001D
     001E     0        MOV     R3,W7
BB   001E
E6   001F
     0020     0        MOV     R4,W8
```

```
BC   0020
01   0021
     0022      0          MOV      R5,W9
BD   0022
40   0023
     0024      0          SEL      RB1
D5   0024
     0025      0          MOV      R0,P4
B8   0025
00   0026
     0027      0          MOV      R1,P5
B9   0027
01   0028
     0029      0          MOV      R2,W10
BA   0029
FE   002A


     002B      0          MOV      R3,W11
BB   002B
19   002C
     002D      0          MOV      R4,W12
BC   002D
FE   002E
     002F      0          MOV      R5,W13
BD   002F
BF   0030
     0031      0          SEL      RB0
C5   0031
     0032      0          MOV      R6,KI
BE   0032
11   0033
     0034      0          CLR      A
27   0034
     0035      0          MOV      _R0,A
A0   0035
     0036      0          CLR      F0
85   0036
     0037      0          CPL      F0
95   0037
     0038      0 LOOP:    MOVX     A,_R0
80   0038
     0039      0 S1:      JNT1     S1
46   0039
39'  003A
     003B      0          MOV      A,R4
FC   003B
     003C      0          MOVX     _R1,A
91   003C
     003D      0          MOV      A,R5
FD   003D
```

EXAMPLE C

REFERENCES

1. Intel "MCS-48 User's Manual"

2. "VAX/VMS Command Language Users Guide," Digital Equipment Corporation.

3. "VAX-II Macro User's Guide," Digital Equipment Corporation

RELATED CONTRACTS AND PUBLICATIONS

F19628-76-C-0111	12 January 1976 through 30 November 1979
F19628-78-C-0218	15 September 1978 through 14 September 1981
F19628-80-C-0050	1 December 1979 through present.

J. Spencer Rochefort, Lawrence J. O'Connor and Norman C. Poirier, "Communication and Instrumentation Systems for Space Vehicles" Final Report for Contract F19628-76-C-0111, (January 1980, AFGL-TR-80-0189.

R. Sukys and J. S. Rochefort, "Control and Data Transmission System for a Balloon-Borne Ion Mass Spectrometer", Proceedings of the International Telemetry Conference, October 1980, vol. XVI, pp. 335-341. Also issued as Scientific Report No. 3, under contract AF19628-78-C-0218, October 1980, AFGL-TR-81-0202.