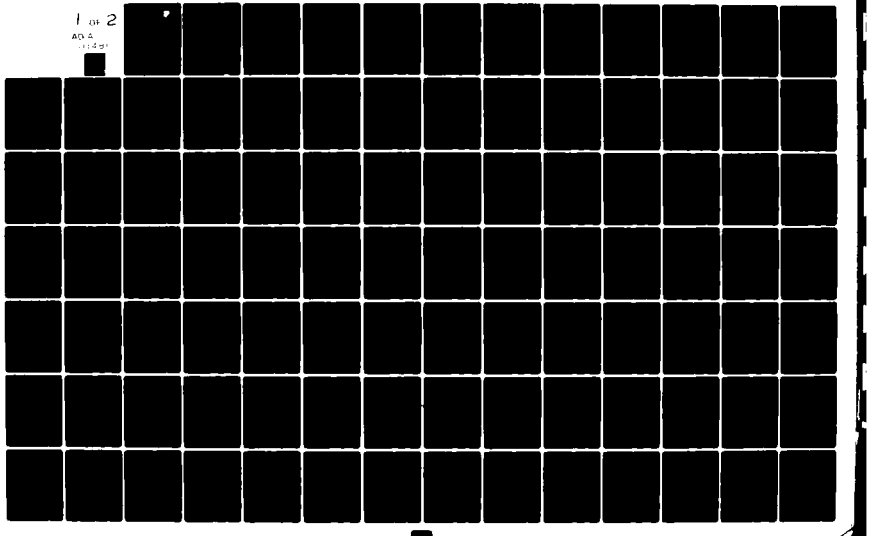
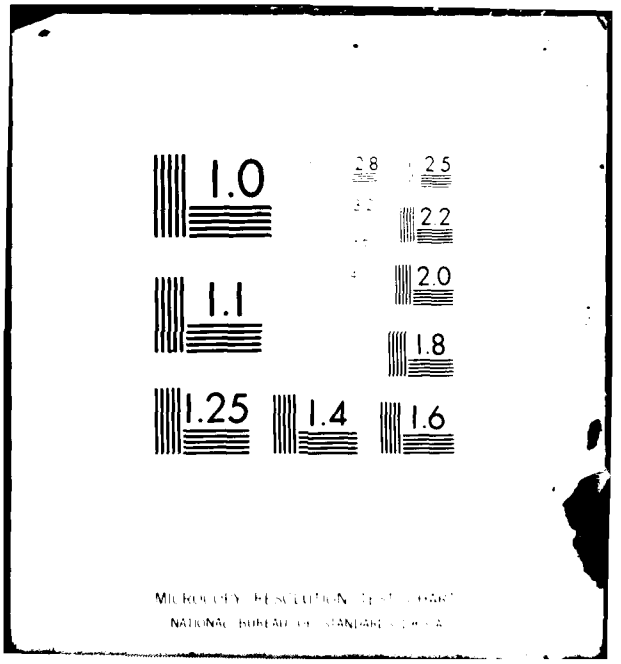
AD-A111 481 PARKE MATHEMATICAL LABS INC CARLISLE MASS F/G 9/5
TOPICS IN OPTICAL MATERIALS AND DEVICE RESEARCH - II. VOLUME II--ETC(U)
JAN 82 T B BARRETT, H HASKEL, C E RYAN F19628-78-C-0089
UNCLASSIFIED RADC-TR-81-372-VOL-2 NL

1 of 2
AD A
111481




1.0
1.1
1.25
1.4
1.6
2.0
1.8
2.2
2.5

RADC-TR-81-372, Vol II (of two)
Final Technical Report
January 1982

TOPICS IN OPTICAL MATERIALS AND DEVICE RESEARCH - II

Parke Mathematical Laboratories

T. B. Barrett
H. Haskel
C. E. Ryan
R. V. Wood
S. P. Yukon

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-81-372, Vol II (of two) has been reviewed and is approved for publication.

APPROVED: Carl H. Pitha

CARL PITHA
Project Engineer

APPROVED: Freeman D. Shepherd

FREEMAN D. SHEPHERD, JR.
Acting Director
Solid State Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ESO) Hanscom AFB MA 01731. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: RADC-TR-81-372, Vol II (of two) 2. GOVT ACCESSION NO.: AD-A111 481	3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): TOPICS IN OPTICAL MATERIALS AND DEVICE RESEARCH - II	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report 1 Apr 78 - 30 Sep 80
	6. PERFORMING ORG. REPORT NUMBER: N/A
7. AUTHOR(s): T. B. Barrett, H. Haskel, C. E. Ryan, R. V. Wood, S. P. Yukon	8. CONTRACT OR GRANT NUMBER(s): F19628-78-C-0089
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Parke Mathematical Laboratories, 1 River Road, Carlisle MA 01741	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62702F, 46001927
11. CONTROLLING OFFICE NAME AND ADDRESS: Deputy for Electronic Technology (RADC/ESO), Hanscom AFB MA 01731	12. REPORT DATE: January 1982
	13. NUMBER OF PAGES: 192
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): Same	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

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)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Carl A. Pitha (ESO)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Fiber-Optics
Optical Waveguide
Wavelength Multiplexing
CCD
Crystal-Growth Mechanism
Optical Time-Domain Reflectometry
Computer
Silicon
Silicon Nitride
Silicon Dioxide
Nicolet
Optic[illegible]se Generator
IEEE-488
Z-80

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

1. A new analysis of fiber-optic wavelength multiplexing was developed.
2. A study of the limitations imposed by materials quality on CCD's was carried out.
3. The SiO_2-Si interface and its problems were investigated.
4. Epitaxial growth mechanism for silicon, silicon nitride, and silicon dioxide were studied.
5. A theory was developed to describe time-domain reflectometry return [illegible]

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

signals.

Foreword

This report is the second part of a two-part Final Report for Contract F19628-78-C-0089. Part I is entitled "Topics in Optical Materials and Device Research - II".

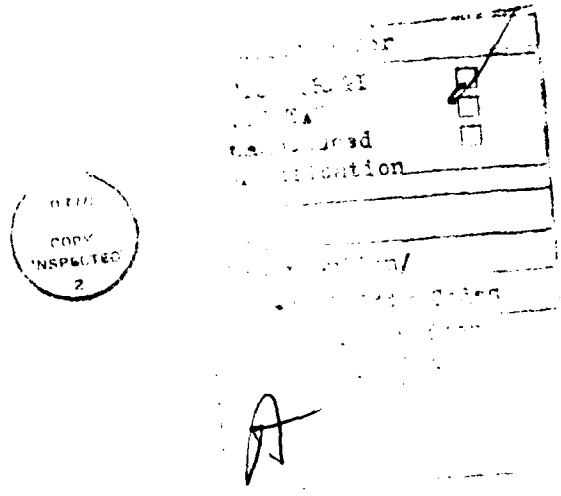


TABLE OF CONTENTS

Preface (to part II)

The design, construction and software/firmware "implementation" of the IEEE-488 interface device described in this report is the result of labors of several people.

The basic design and layout were done by Dan Terpstra, Florida State University, Tallahassee, Florida and Dave Wright, University of Illinois, Urbana, Illinois. The current PROM program was also provided by Dan Terpstra. Construction of the device was done by Larry Armour of A.J. Lincoln & Co., Inc.. Some design modifications and most of the initial debugging was done by George Bartley of AJL&Co.

We gratefully acknowledge the help of these people in this project. We also wish to acknowledge the help of Mark Ahles, Nicolet Instrument Corp., Madison, Wisconsin who came to our rescue when the Nicolet computer failed and who provided the initial contact with Dan Terpstra and Dave Wright.

INTRODUCTION

The NIC-488/CTL (referred to simply as CTL or the CTL) bridges the rather large gap between the Nicolet 1080 computer (and its relatives referred to simply as NIC) and the IEEE-488 (GPIB) bus. The system consisting of NIC-CTL-GPIB is a system of 4 computers: the Nicolet itself, the CTL which actually consists of two microcomputers and the computer which is assumed to be attached to the other end of the GPIB cable. This is an interesting and rather complex system which can, and should, be looked at from several angles in order to fully understand the operation of the overall system. This report provides the following views:

1) The hardware - a brief look at NIC as it pertains to attachment to CTL and the CTL hardware itself. Note that the GPIB is partly a set of definitions and state diagrams and partly some hardware (electrical and mechanical) specifications. From a hardware standpoint, we only include a diagram of the bus-socket pins (which, of course, is standard). Most of this information is contained in the Appendices.

2) System logic - i.e., a description of how the system works. This is done in steps and with varying detail by looking at the NIC/CTL interface and the CTL/GPIB interface separately since they are well isolated logically.

3) System software - there are basically 2 levels of software to be described: the "operating system" software and the "application software".

The above plus various appendices, diagrams and tables should provide the reader of this report with sufficient information to successfully operate and modify the system for applications other than for which it was designed. Additional details can be found in the references.

SECTION 1 - SYSTEM LOGIC

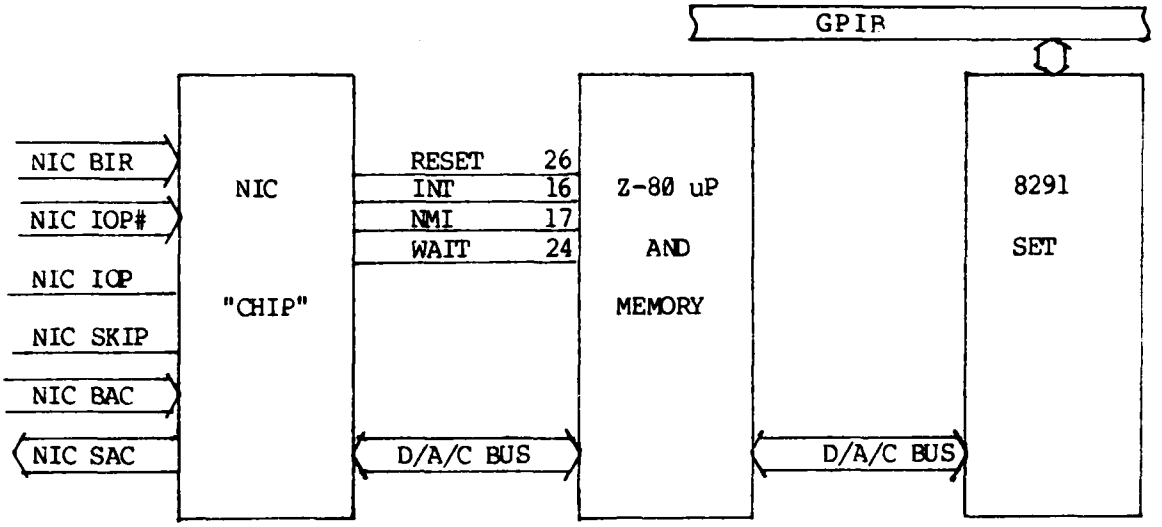
Introductory Comments

The purpose of the CTL is, of course, to allow the NIC to talk to other computers and/or "intelligent" instruments using the GPIB standards. Thus through CTL, NIC can become a controller/talker/listener and, potentially at least, can partake in and control all of the interface functions which collectively define the GPIB. The word "potentially" is used since the current operating system is designed to support only some portions of some of the interface functions. In particular, it is complete in so far as its ability to talk to and control the Hamamatsu C1000 GPIB interface M999-04.

The purpose of this section is to describe, what can be best called, the system logic. As has been mentioned elsewhere, CTL itself actually consists of two microcomputers: a Z-80 with associated memory and firmware (or software) and the Intel chips set consisting of one 8291, one 8292 and two 8293 LSI chips (referred to as the 8291-set). The 8292 is actually an 8800 uP with onboard RAM and ROM which has been programmed (by Intel) to perform a specific set of functions. The 8291 is a special purpose LSI chip which acts like a small computer while the 8293 provides for the electrical interface between the bus and the other chips, along with a small amount of hardwired logic.

Thus data and control flow is from NIC to Z-80 to 8291-set to the bus and whatever is on the other end of the bus. Loosely speaking, NIC initiates all actions in the rest of the system; is the ultimate source of all device dependent data used for control purposes (specifically here, this means camera control data); and is the ultimate sink of all device dependent data from other talkers on the bus. The Z-80 is used mainly to monitor the activities of the 8291-set, to pass data between NIC and the 8291-set. The 8291-set provides for all of the GPIB interface functions and, in particular, passes device dependent data between the Z-80 and the bus.

In what follows, the CTL is described in terms of the NIC "chip", the Z-80 set and the 8291-set where, loosely speaking, the NIC "chip" consists of the onboard discrete logic which performs the digital function transformations from the instruction lines (BIR, IOP, etc.) to the onboard operations, and the latches and I/O ports associated with data transfer between NIC and CTL. The Z-80 set consists of the Z-80 uP and associated memory while the 8291-set contains the 8291, 8292, (2) 8293 plus a port for observing the task complete, TCI, interrupt (so it can be used as a flag rather than an interrupt). The following simplified logic diagram (Figure 1.) pertains to this system as it is described below.



Note:

BAC = buffered accumulator (8 bit data out - from NIC).
SAC = set accumulator (9 bit data in - into NIC).
BIR = buffered instruction register (5 bit data from NIC instruction register).
IOP# = IOP1, IOP2, IOP3 - input/output pulse (approximately 1 usec apart).
IOP = input/output - (Ø signifies an i/o instruction).
SKIP = skip next instruction (used to indicate external device is not ready).

Figure 1. - CTL System Block Diagram

Since the Z-80 set is well documented elsewhere (see ref. 1), it is not described here except in so far as it interacts with the other two main blocks of the system.

The NIC "chip" as seen from NIC

Looking at the NIC "chip" from the NIC computer, there is the following:

1) An 8-bit input port (address 40H) which allows data to be fed from NIC to CTL via the 8 lowest order BAC lines. (Thus, only the 8 lowest order bits in ACC may be transferred to CTL.)

2) A 9-bit output port (address 40H) which allows data to be fed from CTL to NIC via the 9 lowest order SAC lines. The 8 lowest order bits represents data coming from the Z-80 uP while the 9th bit can be set (or reset) by the Z-80 through a control register described below. This bit is called the "service" (SRVC) bit.

3) A 9-bit "command" port through which i/o instructions from the NIC may be decoded. The commands are fed from the NIC instruction register to CTL via the BIR(3-7) lines, IOP1, IOP2, IOP3, and IOP. See Table 1 for the list of instructions, the accepted mnemonics and a summary of their use.

   These lines are used as follows:

   BIR4-7 and IOP - device select (select NIC "chip").
   BIR3 and IOP1,2,3 - provide 8 commands to the NIC "chip". IOP1, 2 and 3 act as read or write pulses. They are sent 1 micro-second apart during the time NIC is ready to send or receive data.

4) A SKIP output. When SKIP is set, instructions which have the skip bit set (e.g., CTLSK) will skip the next instruction. Thus a typical instruction sequence is:

```
CTLSK      / TEST SKIP
JMP $-1    / TEST AGAIN IF NOT SET
CTLRDC     / READ THE DATA
```

				BIR													
	CLA			1	SKIP	0	IOP	DS	DS	DS	DF	DS	IOP 1	IOP 2	IOP 3		
inst. mnem.																inst. (octal)	use
CTLRD	1			1	0	0	0	0	0	1	1	0	1			44064	READ CTL,SRVC
CTLCF				1	0	0	0	0	0	1	1	0		1		4062	CLEAR BUSY/DONE
CTLRDC	1			1	0	0	0	0	0	1	1	0	1	1		44066	READ, CTL, SRVC, CLEAR
CTLWR				1	0	0	0	0	0	1	1	1			1	4071	WRITE CTL
CTLSK				1	1	0	0	0	0	1	1	0			1	6061	SKIP ON DONE
CTLRS				1	0	0	0	0	0	1	1	1		1		4072	RESET

CTLRD
Clears ACC and reads 9 bit data from the "chip" into ACC. (IOP1 is the RD pulse. The 9th bit is "service".)

CTLCF
Clears BUSY/DONE flags. (Using the IOP2 pulse.)

CTLRDC
Logical OR of CTLRD and CTLCF. Used for normal read since clearing the flags indicates that the data port is ready to receive more data from the Z-80.

CTLWR
Sends 8 bit data from ACC to the chip, sets BUSY and clears DONE. (Uses the IOP3 pulse.)

CTLSK
Loads the state of the DONE latch (does not change it) onto the SKIP line. If set, the next NIC instruction is skipped.

CTLRS
Reset: clears CTRLO latches (see below), clear BUSY and DONE, reset pulse to 8291-set and Z-80.

Table 1. - NIC - CTL Instruction Set

The NIC "chip" as seen from the Z-80

Looking at the NIC "chip" from the Z-80 uP, there is the following:

1) NICP (address 40H) - the 8 bit data i/o port for transfering data from the NIC chip to and from the Z-80. Input data to this port is put on the 8 lowest order SAC lines to the NIC while data from the BAC lines are fed to this port for output to the Z-80. In addition, a write to NICP sets the DONE latch and resets the BUSY latch.

2) CTRLO (address 80H) - a 4-bit register (bits 0-3) which may be written to in order to set or reset various latches within the NIC "chip". This register is diagramed and described in Table 2 (note that it is shown as an 8-bit register with 4 "don't care" bits denoted by X).

3) CTRLI (address 80H) - an 8 bit status register which may be monitored by the Z-80 to determine the state of the NIC "chip". See Table 3.

4) WAIT "pin" - this "pin" is attached to the Z-80 $\overline{\text{wait}}$ pin and is used to put the Z-80 into a wait state. See DNEWT and DMAWT.

5) $\overline{\text{BUSY}}$ "pin" - this "pin" is attached to the $\overline{\text{int}}$ pin of the Z-80. When BUSY is set, then $\overline{\text{BUSY}}$ is active low causing a Z-80 interrupt if Z-80 interrupt has been software enabled.

6) INT "pin" - this "pin" is attached to the interrupt pin of the 8291 and as such is used merely to monitor the status of the 8291 interrupt output.

7) $\overline{\text{RESET}}$ "pin" - this "pin" is attached to the Z-80 $\overline{\text{reset}}$ input and when active low causes the Z-80 to be reset. It is also attached to $\overline{\text{reset}}$ of the 8291-set (actually 8292 $\overline{\text{reset}}$).

8) RESET "pin" - this "pin" is attached to the reset pin of the 8291-set (actually the reset input of the 8291).

9) SYC "pin" - this "pin" is attached to the 8291-set system control (SYC) input. It actually goes to the first switch in the DIP set which includes the address switches. When this switch (switch 1 in the DIP set) is off, the CTL is the GPIB controller in charge of the bus.

10) I/O control input "pins" - $\overline{\text{IORQ}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$ input attached to the corresponding Z-80 system control output pins.

X	X	X	DMAWT	X	DNECL	SRVC	DNEWT

DMAWT (DMA-WAIT) - When set, the WAIT pin on the NIC "chip" is sent active low unless the DREQ input to the "chip" is active high. Since WAIT is connected to the Z-80 wait input and DREQ is connected to the 8291 DREQ output, setting DMAWT enables the transfer of data between the 8291 and the Z-80 memory in a "DMA mode". This is described in more detail in the section on the 8291-set.

DMAWT is reset by outputting bit$_4$=0 to the CTRLO register or by the CTLRS command from NIC.

DNECL (DONE CLEAR) - This bit is used to reset (clear) the DONE latch (see below). This reset occurs whenever a write to CTRLO is done with DNECL=1.

SRVC (SERVICE) - The state of the "service" line (the 9th bit) to NIC is determined by this bit. It is latched to 1 as long as SRVC is 1 and to 0 otherwise.

DNEWT (DONE WAIT) - When set, the WAIT pin on the NIC "chip" is sent active low as long as DONE is set. Since WAIT is connected to the Z-80 wait input, this means that the Z-80 is in a wait state as long as DONE is latched on (set). It is taken out of this wait state (DONE is reset) by issuing a CTLCF, CTLRDC, CTLWR or CTLRS.

DNEWT can be reset by writing to CTRLO with DNEWT=0 or by issuing a CTLRS from NIC.

Table 2. - CTRLO Register

INT	BUSY	DONE	AD4	AD3	AD2	AD1	AD0

INT - Status of the 8291 interrupt line. Set to 1 when the 8291 is issuing an interrupt request to the Z-80.

BUSY - Status of the BUSY latch. (1 means BUSY is set.)

DONE - Status of the DONE latch. (1 means DONE is set.)

AD4-AD0 - Status of the 5 address switches in the NIC "chip". These switches can be used to set the CTL talker-listener primary address if desired. Switch on is a logic 0.

Table 3. - The CTRLI Register

Protocol

The main function of the NIC "chip" is to provide for the asynchronous flow of data in bit parallel, byte serial form between NIC and the Z-80.
This is accomplished mainly through the use of the BUSY and DONE latches. Table 4 summarizes how these latches (and others) are influenced by actions of NIC and of the Z-80 uP. Below are some examples of typical software segments which can be used to implement this asynchronous communications. First, however, we note how NIC and Z-80, respectively, observe and control these two latches:

	DONE CTRLI bit 5	BUSY CTRLI bit 6	8291 CTRLI bit 7	DNEWT CTRLO bit 0	SRVC CTRLO bit 1	DNECL CTRLO bit 2	DMAWT CTRLO bit 4	NICP	Z80 INT
CTLRD					READ			READ	
CTLCF	CLEAR	CLEAR							CLEAR
CTLRDC	CLEAR	CLEAR			READ			READ	CLEAR
CTLWR	CLEAR	SET						WRITE	SET
CTLSK	DONE TO SKIP								
CTLRS	CLEAR	CLEAR		OFF	OFF	OFF	OFF		CLEAR
OUT (40),A	SET	CLEAR						WRITE	CLEAR
OUT (80),b2=1	CLEAR								
OUT (80),b1=1					SET				
OUT (80),b1=0					CLEAR				
OUT (80),b4=1							Z80 WAIT LATCH TO 8291 DREQ		
OUT (80),b4=0							DISCON-NECT		
OUT (80),b0=1				Z80 WAIT LATCH TO DONE					
OUT (80),b0=0				DISCON-NECT					
IN A,(80)	READ	READ	READ						
IN A,(40)								READ	

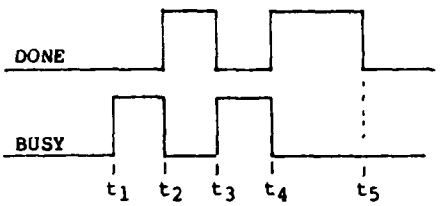
Table 4. - The action of NIC and Z-80 instructions on various "control" bits.

BUSY - set by NIC with CTLWR
reset by NIC with CTLCF, CTLRDC, CTLRS
reset by Z-80 with OUT (40),A
read by Z-80 with IN A,(80)

DONE - set by Z-80 with OUT (40),A
reset by Z-80 with OUT (80),A (bit 2=1)
reset by NIC with CTLWR, CTLRDC, CTLCF, CTLRST
read by NIC with CTLSK (causes an instruction skip if DONE is set)

NIC —> Z-80

Consider the transfer of two data bytes from NIC to the Z-80. Assume that initially both DONE and BUSY are reset and that the Z-80 is waiting in a loop for data from NIC.



t_1 BUSY set with CTLWR (first data byte from ACC ready)

t_2 DONE set and BUSY reset with OUT (40),A (first data byte in the Z-80)

t_3 BUSY set and DONE cleared with CTLWR (second data byte from ACC ready)

t_4 DONE set and BUSY reset with OUT (40),A (second byte in the Z-80)

t_5 DONE reset with CTLCF.

The following code implements the above sequence.

NIC

```
        MTOM  COUNT  /set counter
NEXT,   MEMA @POINT  /to -2
        CTLWR
        CTLSK
        JMP  #-1
        MPOM POINT
        JMP NEXT
        CTLCF
```

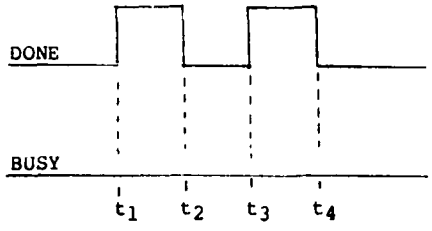
CTL

```
        LD    C,40H
        LD    B,2
        ;LD  A,DNEWT
        ;OUT (80H),A
CT1     IN A,(80H)
        BIT BUSY,A
        JR Z,CT1
CT2     INI
        OUT (40H),A
        DJNZ CT1     ;(or CT2 if DNEWT set
        ;XOR A
        ;OUT (80H),A
```

Another way of implementing the transfer of data from NIC to CTL, particularly if large blocks of data are to be sent, is to set the DNEWT bit which in turn ties the Z-80 WAIT pin to the DONE latch. Since DONE is cleared at the same time as BUSY is set by the CTLWR instruction, it is possible to jump to CT2 without checking for BUSY (except for the first byte). This variant is indicated by commented instructions in the CTL program.

Z-80 ⟶ NIC

Next consider the transfer of two data bytes from the Z-80 to the NIC. Again it is assumed that initially both DONE and BUSY are reset. Now, however, NIC is waiting in a loop for the Z-80 to start sending the data.



t_1 DONE set with OUT (40),A (first data byte from Z-80 is ready).

t_2 DONE reset with CTLRDC. (first data byte into ACC).

t_3 DONE set with OUT (40),A (second data byte from Z-80 is ready).

t_4 DONE reset with CTLRDC (second byte into ACC).

As in the previous case, it is possible to simplify the Z-80 code by using DNEWT. The code below shows this without and with DNEWT set.

NIC

```
        MTOM COUNT
NEXT,   CTLSK     /wait for next
        JMP  #-1 /byte
        CTLRDC    /read & reset DONE
        ACCM @POINT
        MPOM POINT
        MPOMZ COUNT
        JMP NEXT
```

CTL (without DNEWT)

```
        LD  C, 40H
        LD  B,2
CT1     IN  A,(80H)  ;wait for last
        BIT  DONE    ;byte to "clear"
        JR  NZ,CT1
        OUTI
        DJNZ  CT1
```

CTL (with DNEWT)*

```
        LD  C,40H
        LD  B,2
        LD  A,DNEWT
        OUTI         ;send first data byte
        OUT  (80H),A ;set DNEWT
        OTIR         ;send next data byte
        XOR A        ;reset DNEWT
        OUT  (80H),A
```

*Note that DNEWT cannot be set until the first byte is sent out since otherwise nothing will get transferred.

DONE is normally used (set) to indicate that the Z-80 is ready (a byte has been sent or received). It can be used otherwise however, since the Z-80 can both set and reset DONE. (Reset is accomplished with OUT (80H),A with bit 2 set to 1.)

The SRVC status bit (bit 1 of CTRLO) can be used to indicate a special condition within the CTL (caused, for example, by an abnormal condition of the GPIB). This status bit is read by NIC as the 9th bit in ACC after reading the NIC I/O port (40H).

The use of the other latch bit, DMAWT, is discussed further in a later section after the 8291-set has been described.

8291-set

Introductory Comments

The 8291-set consists of the following:

Intel 8291	GPIB Talker/Listener
Intel 8292	GPIB Controller
2 Intel 8293	GPIB Transceivers

A Status Port
Associated circuitry and minor hardware to tie everything together.

The intra-set configuration is shown in Figure 2.

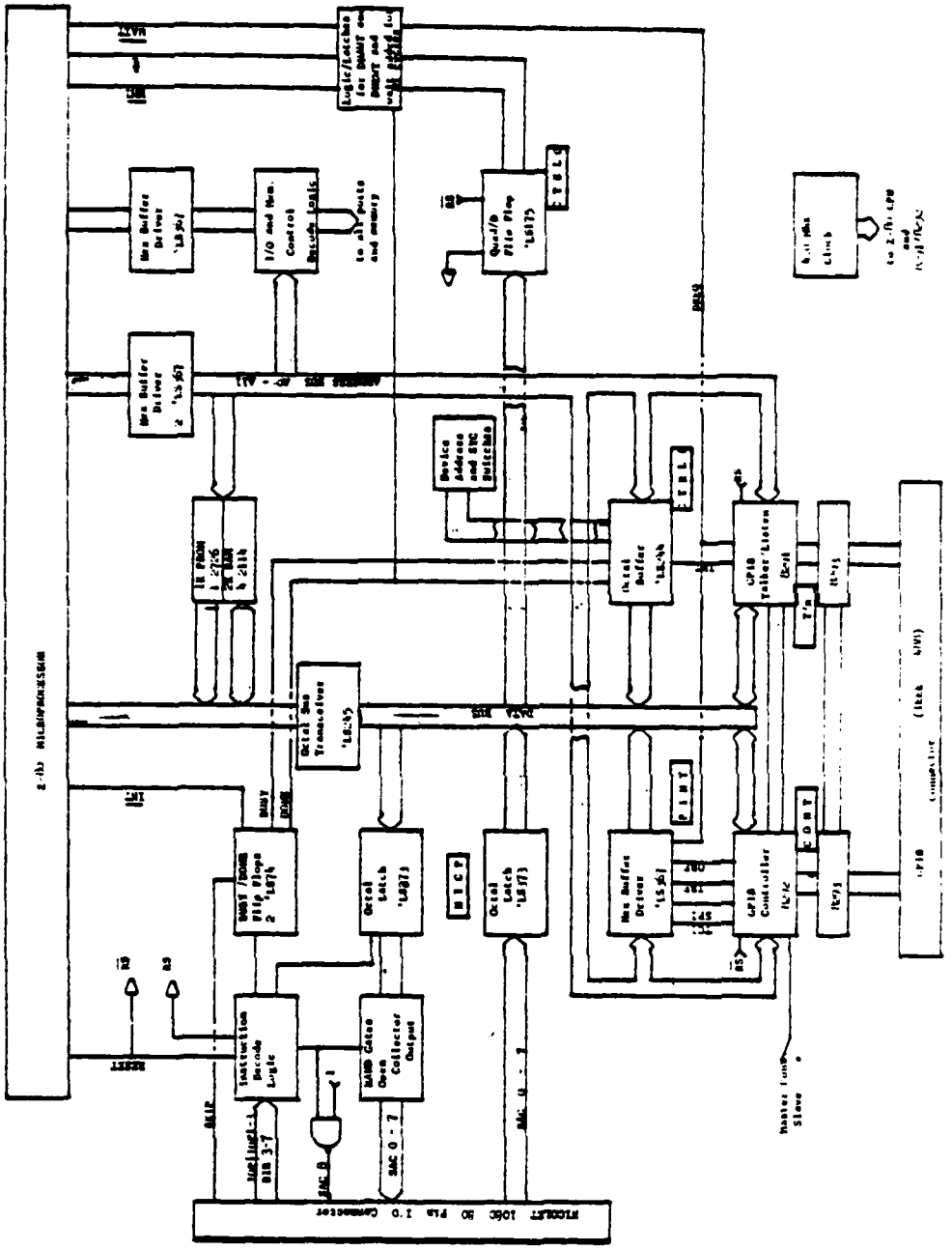


Figure 2. - Intra-Set Configuration

This set provides for the complex transfer of data between the Z-80 uP and the GPIB. It essentially implements all of the interface functions as specified in the IEEE 488-GPIB standards. The 8291-set is best comprehended by "observing" its performance, in conjunction with the controlling uP, while carrying out a typical GPIB task. By extrapolation it should then be a relatively simple procedure to use the 8291-set for any legitimate GPIB task. This performance observation (or task walk-through) is set forth rather exhaustively in the sub-section on "request for service" walk-through. There are several sub-sections leading up to this walk-through which serve both for reference and for description of the 8291-set. The reader is advised to merely glance through the sections on system description and interaction with the uP and to concentrate more on the operation sections.

System Description

In the following system description the various ports and registers of the 8291-set are described. In addition, some detail is given on how these ports, and registers may be used and certain precautions which must be made in accessing them (timing considerations, etc.). Before doing so, several points should be made.

(1) The 8291 may be used alone as a talker/listener. Here, however, since the CTL has GPIB controller capabilities with the help of the 8292, we prefer to look at the chip set as a single entity although parenthetically note is made of which port or register, etc. is associated with which chip. The 8293's are passive devices with little onboard "intelligence", i.e., they act mainly as an electrical interface as opposed to a "logical" interface.

(2) Both the 8291 and 8292 have "hidden" or "indirect" registers which may not be interfaced directly (with a simple IN (port) or OUT (port)). We choose to label them by the bit patterns used to access them, as well as by the names chosen for them by Intel.

(3) Although Intel uses the term "register" for all of the "doors and windows" used for data transfer and status observation, we choose to classify them as ports, registers and memory according to primary usage. The term "register" is used below with a capital R when we wish to talk of the ports, registers and memory collectively. Thus ports are used primarily to transfer random data or specific command data between the Z-80 uP and the 8291-set; registers are primarily used to indicate the status (or state) of the 8291-set or the GPIB while memory is used primarily to hold configuration bytes within the 8291-set, i.e., to configure

the 8291-set with various multi bit pieces of information. Note that some Registers are multi-purpose but are listed under only one category.

Below, all Registers are listed with the following information:

name	- usually as assigned by Intel.
function	- a brief description of the function of the Register.
abbreviation symbol	- for use with Table 5 which describes the purpose of all single bits.
(n)	- where n=1 indicates 8291 Register and n=2 indicates 8292 Register.
protocol symbol	- see the section on Register protocol.
contents with bit labels	- the single bit contents are described in Table 5.
multi-bit description	- (for example, data byte)

ports

Data-in - moves data from the GPIB to the Z-80.

DI (1) TIM1

DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0

DI0 -> DI7 is datum.

Data-out - moves data from the Z-80 to the GPIB.

DO (1) TIM2

DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

DO0 -> DO7 is datum.

Command pass-through - passes "undefined" multi-line interface messages and secondary addresses from the GPIB to the Z-80 set.

CP (1) EV1 (CPT)

CPT7	CPT6	CPT5	CPT4	CPT3	CPT2	CPT1	CPT0

CPT0 -> CPT7 is "command" datum.

Auxiliary mode - passes "mode data" and "commands" from the Z-80 set to the 8291-set (specifically to the 8291). This is a multi-purpose register.

AM (1) DIR

CNT2	CNT1	CNT0	COM4	COM3	COM2	COM1	COM0

COM0 -> COM4 is internal command datum.
CNT0 -> CNT2 is register control datum.

Command field - passes commands from the Z-80 set to the 8291-set (specifically to the 8292).

CF (2) TIM3

1	1	1	OP	C3	C2	C1	C0

C0 -> C3 is internal or external command datum.

Address 0/1 - passes addresses and address "function" bits to the 8291-set. (This data is stored in the memory "registers" Address 0 or Address 1. Bit RS,DT,DL are state SET BITS.)

A01 (1) DIR

ARS	DT	DL	AD5	AD4	AD3	AD2	AD1

AD1 -> AD5 is address datum.

registers (state set)

Interrupt enable 1 - enable interrupts.

IE1 (1) DIR

CPT	APT	GET	END	DEC	ERR	BO	BI

Interrupt enable 2 - enable interrupts.

IE2 (1) DIR

0	0	DMAO	DMAI	SPASC	LLOC	REMC	ADSC

Serial poll mode - enable request service (also holds serial poll status bits).

SP (1) SPEC1

S8	rsv	S6	S5	S4	S3	S2	S1

S1 -> S6 plus S8 is serial poll status datum.

Address mode - selects addressing mode.

AM (1) DIR

TO	LO	0	0	0	0	ADM1	ADM0

ADM0 -> ADM1 is address mode datum.

Auxiliary register A - selects handshake and EOS modes.

AA (1) IDIR1 (Auxiliary mode)

X	X	X	A_4	A_3	A_2	A_1	A_0

A_0 -> A_4 is state-set datum.

Auxiliary register B - select special features.

AB (1) IDIR1 (Auxiliary mode)

X	X	X	X	B_3	B_2	B_1	B_0

B_0 -> B_3 is state-set datum.

Auxiliary register P - set parallel protocol and action.

AP (1) IDIR1 (Auxiliary mode)

X	X	X	U	S	P_3	P_2	P_1

P_1 -> P_3 are parallel poll "address" datum.

Interrupt mask - enable interrupts.

IM (2) DIR

1	SPI	TCI	SYC	OBFI	IBFI	0	SRQ

Error mask - enable error interrupts.

EM (2) DIR

0	0	USER	0	0	TOUT3	TOUT2	TOUT1

Interrupt acknowledge - reset certain interrupt status bits.

IA (2) IDIR2

SYC	ERR	SRQ	EV	1	IFCR	1	1

registers (state observe)

Interrupt status 1 - (corresponds to Interrupt enable 1. Note *that status bits are set regardless of which* interrupts are masked.)

IS1 (1) DIR

CPT	APT	GET	END	DEC	ERR	BO	BI

Interrupt status 2 - (bits 0 to 3 correspond to Interrupt enable 2.)

IS2 (1) DIR

INT	SPAS	LLO	REM	SPASC	LLOC	REMC	ADSC

Serial poil status - (corresponds to Serial poll mode.)

SP (1) EV2

S8	SRQS	S6	S5	S4	S3	S2	S1

Address status - monitors address state and EOI.

AS (1) DIR

ton	lon	EOI	LPAS	TPAS	LA	TA	MJMN

Interrupt status - controller, error, SRQ, etc. status.

IS (2) DIR (SPI reset by IDIR2)

SYC	ERR	SRQ	EV	X	IFCR	IBF	OBF

SPI

Error flag (E4) - (corresponds to Error mask.)

EF (2) IDIR4

X	X	USER	X	X	$TOUT_3$	$TOUT_2$	$TOUT_1$

Controller status (E6) - controller function state.

CS (2) IDIR4

CSBS	CA	X	X	SYCS	IFC	REN	SRQ

GPIB status (E7) - bus status.

GS (2) IDIR4

REN	DAV	EOI	X	SYC	IFC	ATNI	SRQ

Event counter status (E3) - contains current value of the event counter.

ES (2) IDIR4

D7	D6	D5	D4	D3	D2	D1	D0

D0 -> D7 is datum.

Time out status (E9) - contains current value of the time out counter.

TS (2) IDIR4

D7	D6	D5	D4	D3	D2	D1	D0

D0 -> D7 is datum.

TCI status - more interrupt status flags. Used mainly to observe TCI.

TCS (2) DIR

0	0	0	DREQ	IBFI	OBFI	SPI	TCI

(Note that this is an "external" register not part of the 8292 and is provided mainly to provide TCI status.)

memory

EOS - hold an EOS byte.

EO (1) DIR

EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0

EC0 -> EC7 is datum.

Address 0 - holds device primary address (mode 2) or device "major" address (mode 1). (Note that this memory register also holds the disable talker/listener bits passed by the Address 0/1 port.)

A0 (1) IDIR1 (Address mode)

X	DT0	DL0	AD5-0	AD4-0	AD3-0	AD2-0	AD1-0

AD1-0 -> AD5-0 is address datum.

Address 1 - holds device secondary address (mode 2) or device "minor" address (mode 1).

A1 (1) IDIR1 (Address mode)

X	DT1	DL1	AD5-1	AD4-1	AD3-1	AD2-1	AD1-1

AD1-1 -> AD5-1 is address datum.

Event counter (E2) - holds count down from value for the event counter.

EC (2) IDIR3

D7	D6	D5	D4	D3	D2	D1	D0

D0 -> D7 is datum.

Time-out (E1) - holds time to be used (count down from) for the various time out errors (TOUT 1,2,3).

TO (2) IDIR3

D7	D6	D5	D4	D3	D2	D1	D0

D0 -> D7 is datum.

Summarizing the Register "structure" of the 8291-set, there are:

6	ports which may be used to pass commands to the 8291-set or data to and from the 8291-set.
10	state set registers.
11	status (observe) registers (note 1)
5	memory Registers.

Note 1: There is a total of 54 status bits which may be observed to determine the state of the 8291-set and its associated GPIB. It should be noted that some of the status registers are used to hold multi-bit data (e.g., the serial poll register) in addition to status bits; this multi-bit data is not counted as status. It also should be noted that some of this status information is redundant.

Table 5. - 8291-set Status Bit Table

bit (1)	Register (2)	page (3)	Purpose
OP	CF	1-229	1=> bus operation command; 0=> utility command (IDIR3,IDIR4 protocol)
DL	A01	1-208	1=> listener function disabled at address specified by the address datum.
DT	A01	1-208	1=> talker function disabled at address specified by the address datum.
ARS	IE1	1-208	1=> bits 0-6 go to Address-0 register; 0=> bits 0-6 go to Address-1 register.
CPT	IE1	1-207	1=> enable Command Pass Through interrupt.
APT	IE1	1-207	1=> enable "secondary address available" interrupt (if Mode 3 addressing is in effect).
GET	IE1	1-207	1=> enable DTAS interrupt (Device Trigger Active State caused by reception of Group Execute Trigger.
END	IE1	1-207	1=> enable interrupt on END message (a multi-byte transfer has been completed).
DEC	IE1	1-207	1=> enable DCAS interrupt (a Device Clear Active State is in effect - DCL message received).
ERR	IE1	1-207	1=> enable bus error condition interrupt (no active listeners but 8291 is an active talker).
BO	IE2	1-207	1=> enable Buffer Out interrupt (see the section on Register protocol -TIM2).
BI	IE2	1-207	1=> enable Buffer In interrupt (see the section on Register protocol - TIM1).
DMAO	IE2	1-207	1=> enable DMA Out iterrupt (enable direct data transfer from memory to the GPIB).
DMAI	IE2	1-207	1=> enable DMA In interrupt (enable direct data transfer from GPIB and memory).
SPASC	IE2	1-207	1=> enable SPAS Change interrupt (the Talk interface function has just entered or left SPAS).
LLOC	IE2	1-207	1=> enable LWLS/RWLS Change interrupt (the Remote/Local interface function has changed state).
REMC	IE2	1-207	1=> enable LOCS/REMS Change interrupt (the Remote/Local interface function has changed state).
ADSC	IE2	1-207	1=> enable TIDS/LIDS Change interrupt (Talker function/Listener function state change. Also MJMN change).
rsv	SP	1-208	1=> send "request service" message (SRQ line asserted by the 8291).
TO	AM	1-209	1=> enable talk only mode.
LO	AM	1-209	1=> enable listen only mode (if TO and LO are both set, then the 8291 may talk to itself).
U	AP	1-212	0=> enable (1=> disable) parallel poll (U=0 is the local poll enable (lpe) message).
S	AP	1-212	Sense bit involved in response to the ist local message during a parallel poll.
SPI	IM	1-228	1=> enable interrupts on special events (SYC,ERR,SRQ,EV,IFCR: see Register 1A).
TCI	IM	1-228	1=> enable interrupts on task completion.
SYC	IM	1-228	1=> enable interrupt on change of the system controller switch.
OBFI	IM	1-228	1=> enable interrupt on output buffer full.
IBFI	IM	1-228	1=> enable iterrupt on input buffer empty.
SRQ	IM	1-228	1=> enable interrupt on SRQ received.
USER	EM	1-229	1=> enable interrupt on "user" error (8292 not system controller & is requested to send IFC or REN).

Table 5. - 8291-set Status Bit Table (continued)

bit (1)	Register (2)	page (3)	Purpose
TOUT3	EM	1-229	1=> enable interrupt on Time Out Error 3 (handshake stuck on TCSY).
TOUT2	EM	1-229	1=> enable interrupt on Time Out Error 2 (transmission between addressed talker and listener does not start).
TOUT1	EM	1-229	1=> enable interrupt on Time Out Error 1 (current controller does not stop sending ATN after receiving TCT).
SYC	IA	1-231	1=> clear SYC status bit in IS (whenever the IA "command" is sent. See IDIF2, Register protocol).
ERR	IA	1-231	Similarly ERR to IFCR set to 1 will
SRQ	IA	1-231	clear their respective status bits in IS.
EV	IA	1-231	
IFCR	IA	1-231	
INT	IS2	1-206	Shows status of the INT pin (interrupt to uP) on the 8291).
SPAS	IS2	1-206	The local device (actually 8291) has been enabled for serial poll. (Talker function in SPAS.)
LLO	IS2	1-206	The local device is in a "lockout" state (Remote/local function is LWLS or RWLS).
REM	IS2	1-206	The local device is in a "remote" state (Remote/local function is RWLS or REMS).
SRQS	SP	1-208	1=> the Service request function is in the SRQS state.
ton	AS	1-209	1=> the 8291 is in the talk only state.
lon	AS	1-209	1=> the 8291 is in the listen only state.
EOI	AS	1-209	1=> that an END message came with the last data byte.
LPAS	AS	1-209	1=> that the listener primary address has been received. (LE function in LPAS.)
TPAS	AS	1-209	1=> the talker primary address has been received. (TE function in TPAS.)
LA	AS	1-209	1=> Listener function is in the LACS (Listener Active) or LADS (Listener Addressed) state.
TA	AS	1-209	1=> Talker function is in the TACS (Talker Active), TADS (Talker Addressed) or SPAS (Serial Poll Active) state.
MJMN	AS	1-209	1=> The other AS bits refer to the Minor talker/listener.
SYC	IS	1-227	1=> System controller switch has changed state.
ERR	IS	1-227	1=> An error has occurred (USER,TOUT1, 2, or 3).
SRQ	IS	1-227	1=> Service request message has been received.
EV	IS	1-227	1=> The event counter has counted down to 0.
IFCR	IS	1-227	1=> Interface clear message has been received.
IBF	IS	1-227	1=> The 8292 uP input buffer is full. (Do not input another byte until IBF=0.)
OBF	IS	1-227	1=> The 8292 uP output buffer is waiting to be read.
CSBS	CS	1-228	1=> Controller interface function is in the Controller Stand By State (CSBS).
CA	CS	1-228	1=> Controller interface function is in CAWS or CSWS or CACS.

Table 5. - 8291-set Status Bit Table (continued)

bit (1)	Register (2)	page (3)	Purpose
SYCS	CS	1-228	1=> System control interface function is in System Control Active State (SACS). (Determined by SYC switch.)
IFC	CS	1-228	1=> Interface Clear message is being sent.
REN	CS	1-228	1=> The Remote Enable message (REN) is being sent.
SRQ	CS	1-228	1=> The SRQ line is active. (Controller function Controller Service Requested State, CSRS).
REN	GS	1-228	1=> The REN line is active.
DAV	GS	1-228	1=> The DAV line is active.
EOI	GS	1-228	1=> The EOI line is active.
SYC	GS	1-228	1=> The System Controller line is active. (Switch is on.)
IFC	GS	1-228	1=> The Interface Clear line is active.
ATNI	GS	1-228	1=> The ATN line is active.
SRQ	GS	1-228	1=> The SRQ line is active.
DREQ	TCS		1=> Data byte ready for input or output in DMA mode (8291).
IBFI	TCS		1=> Input buffer full (8291 interrupt).
OBFI	TCS		1=> Output buffer full (8291 interrupt).
SPI	TCS		1=> Special event interrupt (8291).
TCI	TCS		1=> Task complete interrupt (8291).
DT0	A0	1-209	1=> Talker function disabled at primary/major address.
DL0	A0	1-209	1=> Listener function disabled at primary/major address.
DT1	A1	1-209	1=> Talker function disabled at secondary/minor address.
DL1	A1	1-209	1=> Listener function disabled at secondary/minor address.

(1) bit title (see 8291-set System Description and reference (2)).

(2) see 8291-set System Description — this is the Register abbreviation.

(3) pages in reference (2) where described.

Interaction with the Z-80

The 8291-set is obviously a complex device and many of its features are not used by the CTL. For example, the 8291-set has several interrupts which can be used but in CTL no interrupts are used, rather, status bits are observed and appropriate action taken. Thus, in what follows, the description of the interaction between the Z-80-set and 8291-set is based on what is actually done within the CTL which in turn is partly based on the particular application fo which the CTL was constructed.

This description is based on information in reference 2 and, in particular, the Data sheet section on the 8291 and 8292 and the Application Notes section "Using the 8292 GPIB Controller".

General observations

1) Many of the Registers are indirect ones, i.e., are not accessible by a single read or write (IN or OUT in Z-80 memories). For example, the majority of the 8292 status registers are read using "utility commands" which in turn require the checking of status bits while being used to get the desired status bits. In other words, a certain protocol is required. The first section describes the different protocols used along with some timing consideration which must be observed.

2) The Z-80 "sees" the GPIB only via the 8291-set. Within this set, the 8291 itself is used to implement all of the Interface Functions (as specified in the IEEE-488 standards) with the exception of the C (Controller) Interface Function (CIF). Moreover, the 8291 is used to handle all multi line messages even for messages which are to be used only by the CIF. Thus, the "logic" of the 8291-set reflects this division of labor and the commands sent from the Z-80 set to the 8291-set must be such that the 8291 is put in the right state relative to the 8292. In other words, there is a subtle interplay between Interface states (as defined by the standard) and the actual states of the 8291 and 8292.

3) While the 8291-set has been designed to implement all of the Interface Functions, the user often has the choice of having the Z-80 set do much of the work. Within CTL the Z-80 set acts only as the source and sink of some of the local messages (and thus controls the overall sequence of events on the bus) and lets the 8291-set do all of the actual interfacing. Of course, the Z-80 set must also take approriate action under various bus error conditions.

Register protocol

As mentioned above, various Register observe different protocols in their accessibility. We classify these protocols as direct (DIR), timed (TIM), indirect (IDIR) and special (SPEC).

Direct

These Registers receive or return information after a single IN or OUT instruction of the Z-80. Most of the 8291 status and state registers can be read or written directly. The 8292 Interrupt Status, Interrupt Mask and Error Mask Registers are also direct.

Timed

Several of the Registers require varying amounts of time to elapse before the data in or out is valid or before a command has been executed. There are 3 sub-categories of time Registers, classified according to the status bits which must be observed for successful completion.

TIM1

These Registers use the BI status bit of the Interrupt status, register. When BI is set, data in a TIM1 register is valid. The BI status bit is reset after a read of the Interrupt status 1 register, a read of the TIM1 register or by a pon local message.

TIM2

These Registers use the BO status bit of the Interrupt status 1 register. When BO is set the corresponding Register is ready to output more data. The BO status bit is reset after a read of the Interrupt status 1 register, a write to the TIM2 register, by the assertion of ATN or by TACS. (Note that in tne CTL configuration the 8291 never sees ATN.)

Note:

The TIM1 and TIM2 Registers are used to receive and send, respectively, multi-line messages on the GPIB. If, for some reason, the devices on the bus do not respond correctly (e.g., are not turned on or are not attached), then the status bits will never be set.

TIM3

These Registers use the IBF bit of the Interrupt status register. When this bit is set, the TIM3 Register is not ready to receive data from the Z-80 set. It is reset whenever the 8292 has accepted data sent to it from the Z-80. It is also reset by either an external reset (e.g., CTLRS), pon local message or by the RST command. The standard procedure for TIM3 registers is to first wait until IBF is reset, write to the Register (usually part of accessing an indirect register) and then wait for IBF to reset.

Note that there is no corresponding Register for output which uses the OBF bit since output from the 8292 is via the indirect registers which require the use of the TCI status bit.

Indirect

The indirect registers require more than a simple read/write or read/write and wait for a status bit. They require the use of an auxiliary register and in some cases a completion signal. There are 4 sub-categories of indirect registers depending on the actual code used to access them.

IDIR1

This is the simplest type of indirect Register. It is only necessary to write to the "intermediate" register. For example, the Address 0 and Address 1 registers are written to via the Address mode register. Auxiliary register A, Auxiliary register B and Auxiliary register P are written to via the Auxiliary mode register. No status bits are checked.

IDIR2

The IDIR2 Registers are those which use IACK (interrupt acknowledge command) to "write" to them (actually reset interrupts) via the Command field register. The status bit used is SPI in the TCI status register to indicate that the SPI status bits have been reset. These (Special Interrupt) bits are SYC, ERR, SRQ, EV and IFCR in the Interrupt status register. It should be noted, however, that if the interrupt conditions (any of them) persist, that SPI will be set again after a short period of time. Also note that the IACK command itself is given via the Command field register which is a TIM3 type register.

IDIR3

These registers are the indirect write-to registers of the 9292 and include the Event counter (E2) and Time out (E1) memory Registers. These are written to by first writing E2 or E4 via the Command field register and immediately writing the desired data to the indirect register. Note that it is not necessary to check IBF between these two writes, only after the last one.

The Command field register has the address-0 line (A_0) set to 1 while all the indirect registers are written to or read with A_0=0. It is somewhat a question of semantics as to whether one should consider that a "Data field" (in/out) register is used to pass data to the indirect registers or whether they are read or written "directly" after the appropriate command has been given. Here the latter interpretation is chosen.

IDIR4

These are the indirect read registers of the 8292 utilizing utility commands E3 to E9. They differ from the IDIR3 registers in two respects. One (obvious) is that data is read from them. The second is that the TCI status bit of the TCI status register must be observed to indicate that the desired register is ready to be read. TCI is also used to indicate completion of various "direct" commands by the 8292. It is very important to observe that TCI does not reset immediately on issuing a command so that usually a wait loop must be inserted to observe the reset of TCI before another loop can be entered to observe the set of TCI.

Event

The event Registers are those which contain valid data only on the occurrence of events which are signaled by setting of various event status bits. These are usually "special" events associated with GPIB interface functions and which usually occur rather infrequently. They are classified according to the actual event (and status bit) associated with them.

EV1

The Command pass-through register contains a valid undefined multi-line interface message (ATN must be set) whenever the CPT status bit of the Interrupt status 1 register is set (Command pass-through must first have been enabled by writing the appropriate command to the Auxiliary B register). This status bit, along with all others in the Interrupt status 1 register, is reset when read.

EV2

The Serial poll status register contains the serial poll status byte (actually bits S1-S6 and S8) as written into the Serial poll mode register. (See SPEC1, below.) The SRQS bit is used to indicate the status of a serial poll by the controller-in-charge. The SRQS bit is set when the 8291 Service request interface function is in the Service Request State (SRQS). Entry to this state is initialized by writing to the Serial poll mode register with the rsv bit set. The SRQS bit is reset when the controller-in-charge does a serial poll and reads the status byte in the serial poll status register.

Special

Special Registers are those which require a special protocol for setting or clearing bits.

SPEC1

The Serial poll mode register may be written directly with the rsv bit set. This is equivalent to sending the local message "request service". This bit should be reset by writing to the serial poll mode register with rsv=0 immediately after the serial poll status byte has been read by the controller-in-charge (as indicated by SRQS - see EV2 above).

8291-set operation

In the previous section the 8291-set was described in terms of the various Registers whhich are accessible to the Z-80 set and which may be used to provide most of the interface functions for the bus. In this section, the 8291-set is described from an operation standpoint, i.e., how it implements various interface functions and its relationship to the Z-80 set while providing this implementation.

Actually we trace through, in some detail, what is involved in answering a request for service from another device (in this case the Hamamatsu camera). In doing so, we get involved in many of the interface functions which make up the complete GPIB system.

GPIB review

1) The GPIB is a set of mechanical, electrical and operational specifications which, when properly implemented, provide for asynchronous transfer of information (messages) among several

devices (with GPIB interfaces) at data rates up to one million bytes/second. Electro-mechanically the bus consists of 16 data and control lines (8 data and 8 control) plus grounds and shield. The control lines provide, among other things, for "handshaking" of messages in such a manner that in a multi-listener configuration (there can be many "listeners" but only one "talker" active at the same time), the data transfer rate is adjusted down to the slowest listener. In addition to data transport, the interface provides for various control functions such as remote/locate switch, device trigger, device polling (serial and parallel), etc. These functions are described in detail in most of the literature on the GPIB and, in particular, in the standard. See reference (3).

2) In order to comprehend the structure of the GPIB it has been functionally partitioned into ten interface functions (perhaps eleven if a distinction is made between the Control Function and the System Control Function). GPIB overall operation is described by several state diagrams wherein each interface function has associated with it one or more (three is the maximum) connected state diagrams. Each such connected state diagram consists of two or more mutually exclusive states (represented by labeled circles) and connections showing how the transition from one state to another is made. (These are illustrated later in much detail for a particular bus operation (service of a request for service)). These state diagrams can, if one so desires, be translated into a set of timing diagrams for bus and "local" signals. However, since in particular, the 8291-set takes care of most of the bus timing, it is much easier to work completely with the state diagrams.

3) Each interface function can be diagramed as a box (see Figure 3), representing the function, with a set of message inputs and outputs (Interface messages) and connections to the other interface functions such that their states may be observed. (Interface functions are interdependent in general.) The input messages always change the state of the interface function. These messages are classifed as <u>local</u>, i.e., between the device containing the interface and the interface itself and <u>remote</u> for messages between the interface and the bus (to other devices). The device itself (as opposed to the interface) is usually described by a set of device functions so local messages are between device functions and interface functions. Roughly speaking, at least from an operational standpoint, the Z-80 set represents the device function(s) and the 8291-set, the interface functions.

Remote messages consist of interface messages (they change the state of an interface function) and device dependent messages (data to be passed among various devices on the bus). The interface messages are also partitioned into six classes which indicate their usage and also into uni-line and multi-line messages. (All device dependent messages are multi-line.)

All remote messages are coded by 3 capital letters in the diagrams (e.g., ATN for attention) - some are input, some output and some both. Local messages, on the other hand, are coded by 3 uncapitalized letters (e.g., pon for power on). There are no local messages from an interface function to a device function, only from device to interface. Thus, in particular, device dependent data (e.g., DAB for data byte) goes "directly" between device functions. Similarly, addresses go "directly" between devices although they are "enabled" by the Controller function.

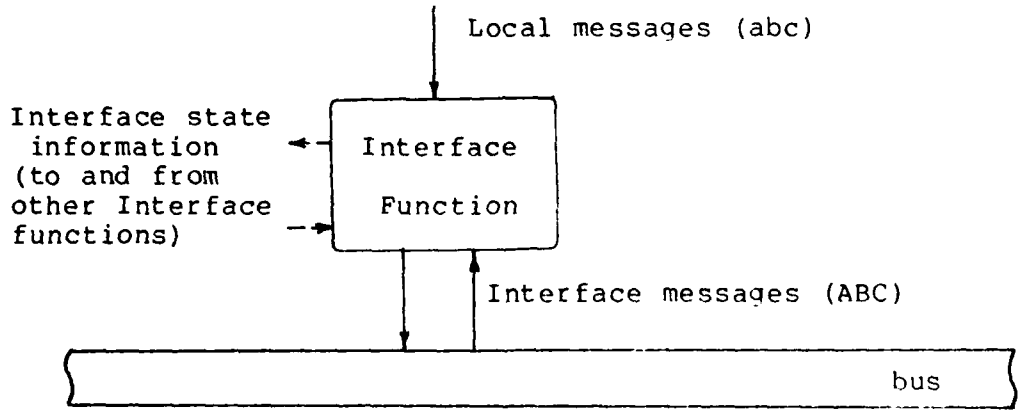


Figure 3. - Interface Function Diagram

There are certain points about messages that are useful to remember.

(1) All messages intercepted by (input to) Interface functions serve to change the state of the function provided a state transition is permitted (by the state of other Interface functions).

(2) Local messages are defined only from device function(s) to Interface functions. However, the Interface function must often "notify" the device function of some event. These "messages" are not defined by the standard.

(3) All remote interface messages received are intercepted by one or more Interface functions and serve to change the state of the intercepting function(s).

(4) All remote, coded (multi-line) interface messages sent from a device originates in a device function not an interface function. However, such remote messages must go via (be enabled by) some interface function - most commonly the Controller function.

Overview

Roughly speaking, the Z-80 set represents the device functions and is the source of Local messages and the sink for device dependent Interface messages. The 8291-set represents all of the interface functions and, among other things, interprets and takes action on all non-device dependent Interface messages. (Exceptions to this blanket statement should become obvious in the following discussion.) Thus, the coding of the Z-80 does not have to include much of the bus "protocol". As noted earlier, the 8291-set consists of the 8291, 8292 and two 8293's and interconnections such that as a unit it provides for all interface functions including proper logical leads on the bus. The 8291 by itself provides for all interface functions except Control (and System Control). More exactly, all Interface states exclusive of those associated with Control and System Control are associated with the 8291 and the remainder with the 8292. It must be noted, however, that the 8291 and 8292 do not operate independently of one another. In particular, all multi-line messages from the device (Z-80 set) go via the 8291 and the 8292.

Request for Service

With this introduction, we describe a response (of the CTL) to a request for service. The strategy is to look at the appropriate Interface function immediately affected by the request, then other Interface functions whose states affect the immediate one, the various actions required by the 8291-set to cause various state transitions and, finally, how these actions are made to happen by appropriate instructions from the Z-80 set.

State diagram viewpoint

[For clarity, all states associated with the camera have subscript r (remote), while those of the CTL have subscript 1 (local). Thus, we refer to local and remote Interface functions referring to the CTL (or 8291-set) and the camera, respectively. Since there is only one controller -- that associated with CTL -- no subscript is used for the control interface function.]

Briefly, a request for service is initiated by a device (the camera in this instance) sending the local message, rsv_r (request service) to its Service request SR_r Interface function causing it to go from the $NPRS_r$ state (Negative Poll Response) to the $SRQS_r$ state (Service Request State). It should go to this state provided its Talker (T_r) Interface function is not in the $SPAS_r$ state (Serial Poll Active). The T_r function should not be in this state but is later put there briefly during the serial poll which is conducted in response to the request for service. While in the $SRQS_r$ state, the camera SR_r function sends the SRQ message.

In response to SRQ, the local Control function makes the transition from the CSNS state (Controller Service Not Requested) to the CSRS state (Controller Service Requested State). While in this state (as long as SRQ is active) it notifies the device function of SRQ (say by sending the srq message).

Upon receiving srq, the local device function initiates a serial poll (1) to find out what device is requesting service and (2) why it is requesting service (by decoding a status message). This is done by sending the talker address, MTA, of the camera "via" the C function (assumed to be in the CACS state). Since MTA is a multi-line message, it must be "handshaked" to the remote devices. This is accomplished through use of the Source Handshake function (SH_1). As will become clear later, the local Talker function T_1 is used in this process although it is not "officially" part of this data transfer. Thus, the local device function makes the local Talker interface function take part in this transfer by putting it into the $TACS_1$ state. (Note that if many devices were connected to the bus, a set of addresses would be sent to determine which one was requesting service. Moreover, it might also be necessary to put other devices in a known state before conducting the serial poll.)

Reception of MTA_r causes the camera Talker Interface function (T_r) to go from the $TIDS_r$ state (Talker Idle) to the $TADS_r$ state (Talker addressed) with the help of the AH_r (Acceptor handshake) function.

The local device function then sends the serial poll enable message (SPE) via C which is still in the CACS state. This causes the T_r function to go from the $SPIS_r$ state (Serial Poll Idle) to the $SPMS_r$ state (Serial Poll Mode).

Now the local device function sends the local message, gts (go to standby) to C causing it to go from the CACS state to the CSBS state as soon as the Source handshake (SH_1) function attains the appropriate stave. (Note that while the active controller is in this state, device dependent messages may be sent - ATN is false.)

The camera T_r function now goes from the $TADS_r$ state to the $SPAS_r$ state (Serial Poll Active) and remains there as long as ATN is false. This, in turn, causes the SR_r function to go from the $SRQS_r$ to the $APRS_r$ state (Affirmative Poll Response). (Note that rsv_r must be sent continuously from the camera device function to SR_r to keep SR_r in the $SRQS_r$ state until T_r goes to the $SPAS_r$ state.) The camera device function should then stop sending SRQ (more specifically, send it passively false). The T_r function sends the RQS message (DIO line 7 = 1) along with a status byte (DIO 1-6 and 8), the STB message, from the camera device function.

The local device function receives the RQS STB message via the local Listen Interface function (L_1) which has previously been put into the $LACS_1$ state (Listen Active). (How this is done will be demonstrated later.) The transfer of datum (the RQS STB message) takes place when the local Acceptor Handshake (AH_1) function goes to the $ACDS_1$ state (Accept Data).

Note that at this point the camera is (or should be) still in the $SPAS_r$, $APRS_r$ state and the local interface functions are in the $TACS_1$ (somewhat artificially), $LASS_1$ states. In order to leave these states and get back to inactive states, the Controller in charge of the bus must send the ATN message. The Controller currently is in the CSBC (Standby) state. The local device function now sends the local message, tcs (take control synchronously) to C. Assuming that the RQS STB message has been successfully transferred, the AH_1 function should be in $ANRS_1$ state and C will go to the CSHS state (Controller Standby Hold) state and then go to the CSWS state (Controller Synchronous Wait) after a time delay of at least 1.5 microseconds. When (and if) C goes to CSWS, it sends the attention message ATN and after another "decay" state falls back to CACS. If the local tcs message is de-asserted while C is in the CSHS state, it falls back to the CSBS state. This may happen, for example, if the AH_1 function cannot go to the $ANRS_1$ state (the handshake is "stuck"). If this happens, the only way short of a pon (resetting everything) is to have the local device issue the message tca (take control asynchronously) with the possible loss of a data byte.

As soon as the ATN message is received, the remote Talker function T_r goes to the $TADS_r$ state (from $SPAS_r$). This, in turn, allows the SR_r function to go from the $APRS_r$ state to the $NPRS_r$ state. (This is the SR "ground" state.)

Finally, the T_r function is put into its "ground" state by sending the UNT (untalk) message. This is accomplished in the same way as MTA was sent as described above.

From the above it is clear that all Interface functions with the exception of Parallel Poll (PP), Remote/Local (RL), Device Clear (DC) and Device Trigger (TR) are involved in a service request and subsequent serial poll. In fact, these latter four Interface functions are not implemented in the CTL since the camera does not implement them. They are, however, easily implementable with the 8291-set; it is mainly a matter of initiating the correct messages within the Z-80 set.

8291 state viewpoint

In the above description of what takes place during a request for service and serial poll, a description was given of the various states the remote Interface function (associated with the camera) must attain during the process along with remote device function messages. From the standpoint of CTL, of course, all that can be ascertained about the remote device (the Interface function states, device function states, etc.) is through observation of various bus messages received and reaction to bus messages sent. Thus, in what follows, the remote states are considered only in so far as they can be "observed" via bus messages (probes). As a matter of fact, the camera does not react according to the standards. For example, its SR_r function does not go into the $SRQS_r$ state (as observed by looking for the SRQ message) until the Controller function C has gone to standby CSBS. Moreover, the camera always has two RSQ messages to give rather than one. Below we assume the ideal condition, i.e., the remote device or devices adhere to the standards.

Having described the service request/serial poll sequence from the state diagram viewpoint, we now describe it by looking at corresponding states of the 8291-set and how these states are intitiated either by the Z-80 instructions or information obtained from the bus. It should be emphasized that there are often alternative ways of using the 8291-set. In particular, as mentioned earlier, the Z-80 can be interrupt driven. Here it is "status bit driven", i.e., interrupts are not used at all.

Before going through the step by step sequence outlined above, it is necessary to put the GPIB (local Interface functions and remote Interface functions) into a known "ground" state. In addition, the 8291-set must be initialized in some manner such that

it too is in a known "ground" state in so far as its "ground" state may include more than the ground state of its associated Interface functions. For this purpose it is assumed that there are no other devices on the bus with the Controller function. (The CTL does have provisions for another device to have control of the bus on power up but, in this condition, does not take part in any bus transactions until its local Control function becomes active. Thus it is possible to control the camera from other devices such as the HP 9825 calculator and to then have this device pass control to the CTL. One of the appendices describes what is involved here.

This "ground" state is established first by pulsing the external reset pin on both the 8291 and 8292 using the CTLRS instruction of NIC. Among other things this has the effect of the local message, pon (power on) putting all Interface functions into their "ground" states. Thus, for the 6 Interface functions we are concerned with:

Source Handshake	(SH)	=>	SIDS	
Acceptor Handshake	(AH)	=>	AIDS	
Talk	(T)	=>	TIDS SPIS	(Note: The 8291-set is not used as an extended talker.)
Listen	(L)	=>	LIDS	(Note: The 8291-set is not used as an extended listener.)
Control	(C)	=>	CIDS CSNS	
System Control	(SC)	=>	SIIS SRIS SNAS	

Assuming that the SYC switch is on, SC goes from SNAS to SACS (System Control Active), i.e., SYC "on" is equivalent to the local message rsc (request system control). This in turn causes the transition SIIS ——> SIAS (the local message sic, send interface clear, is built into the 8292) and the IFC (Interface Clear) message is sent. At the same time (more or less), the C-function ends up in CACS (Controller Active) and the ATN message is sent (continuously). The IFC message is sent via an implied ABORT command which takes 155 microseconds to complete.

External reset (by CTLRS) also causes the Z-80 to start execution at 00. Thus, the initializtion subroutine starts here. The first thing to be done is to wait for the ABORT command to finish execution. This can be done by using a DJNZ loop, i.e.,

```
        LD   B,0
LI      DJNZ L1
```

as the first pair of instructions. (See note 1, below.)

Internally, the following Registers are reset (cleared):

Interrupt Status 1
Interrupt Status 2
Serial Poll Mode
Address Status (EOI bit only)
Auxiliary Register A
Auxiliary Register B
Auxiliary Register P (Parallel poll bit is reset)
Interrupt Status
Interrupt Mask (See note 1)
Error Flag
Error Mask
Time Out
Event Counter (disabled)

Note 1: Because the Interrupt Mask register is cleared, it is not possible to use TCI to observe when the ABORT command has been completed. This is the reason for using the DJNZ loop instead.

After this external reset and ABORT, the following should be done to put the 8291-set into the correct state for succeeding operations:

1) Set the TCI bit in the Interrupt Mask register.

```
INTMR   EQU  1OH     (The Interrupt Mask register is port
INTM    EQU  ØAØH     1OH in the CTL)
 LD     A,INTM
OUT     (INTMR),A
```

2) Preset the internal counter (in the 8291 to match the external clock frequency) used to generate the T_1 time delay for the SH function. The frequency used in the CTL is 4 Mhz; the following instructions are used:

```
CLKRT   EQU  24H
AUXMD   EQU  25H      (Auxiliary Mode register is port 25H
 LD     A,CLKRT        in CTL.)
OUT     (AUXMD),A
```

3) Disable talker and listener at primary and secondary (or major and minor) addresses.

```
ADRØ1   EQU  26H      (Address Ø/1 register is port 26H
DTDL1   EQU  60H       in CTL.)
DTDL2   EQU  ØE0H
 LD     A,DTDL1
OUT     (ADRØ1),A
 LD     A,DTDL2
OUT     (ADR01),A
```

4) Release the "initialization state". This is done with an "immediate execute" pon.

```
XOR  A
OUT  (AUXMD),A
```

The current software in the CTL also includes provisions for looking for "time out" errors for which a time value is inserted in the Time Out register. For clarity this is ignored in the following.

In addition to initializing the 8291-set, the CTLRS instruction initializes the remote devices (in this case just the camera) to their "ground state" through the IFC message. In addition to setting the camera Interface function to ground, the IFC message also causes the camera device function to reset. Specifically, the camera Format and Control functions go to their default values.

At this point the 8291-set is in "ground" state, except that it is the System Controller and Controller in charge sitting in state CACS, and waiting for the Service Request message to come from the camera. (The fact that a previous command to the camera from CTL caused the camera to initiate a video sweep is irrelevant here. We assume that the state of the 8291-set and bus is as given.)

Note that in the CACS state, the Controller function is constantly sending the ATN message. This should not affect the camera's ability to transfer from (Service Request interface function) NPRS to SRQS but, in fact, the camera will not make the transition while ATN is asserted. For simplicity we assume that it adheres to the standard and will make the transition.

When the SRQ message is received, the SRQ status bit in the Interrupt Status register is set. The SRQ status bit on the GPIB Status register is also set. It is looked for on the Interrupt Status register in the followilg loop:

```
        INTST  EQU  11H      (The Interrupt Status register
        SRQ    EQU  5        is port 11H on CTL.)
L2       IN    A,(INTST)
         BIT   SRQ,A
         JR    Z,L2
```

It should be noted that later this bit must be reset using the interrupt acknowledge (IACK) command. However, the SRQ message is sent until the camera's SR function goes to the APRS state as a result of a serial poll initiated by CTL.

Thus, the local Controller in effect sends the local srq message to the device function (the Z-80 set) which starts to initiate the serial poll response. As noted earlier, the first step is to transfer the local Talk function to the TACS state. This is done by sending the local message ton (talk only) which puts the Talker to the TADS (Talker Addressed) state. However, since the Talker (the 8291) does not see ATN (which is still being sent), it goes without further instructions to the TACS state from which it can transmit device messages. The local Source Handshake (SH) function takes part in this message transfer but is transparent to the user except for a status bit which says when the next message can be sent to the 8291-set. The following code is used:

```
INT1    EQU  21H      (The Interrupt Status regis-
BOM     EQU   2        ter is assigned port 21H
TON     EQU  80H       in CTL.)
DOUT    EQU  20H      (The Data Out register on CTL
                       is assigned port 20H.)
TALKP   EQU  42H      (The camera's primary talk
                       address.  It does not use a
                       secondary address for serial
                       poll.)

        LD   A,TON
        OUT  (ADRMD),TON    (Send local ton message.)
        LD   A,TALKP
        OUT  (DOUT),A       (Send camera MTA)
L3      IN   A,(INT1)
        BIT  BOM,A          (Wait for the message to be
        JR   Z,L3            handshaken out of the Data
                             Out register.  This is
                             called WAITO (macro) below.)
```

Note:

The byte output status bit, BO, is set whenever the Talk function is in TACS and the Source Handshake function is in state SGNS (or SWNS). This means that it is ready to transfer a multi-line message to the bus. By waiting here, the Data Out register will be ready the next time a message is to be transmitted. The local message nba (new byte available) is generated whenever a byte is placed in the Data Out register.

With the reception of its talk address, T_r goes to TADS and awaits the SPE message. The following code sends out this message:

```
SPE    EQU   18H
 LD    A,SPE
 OUT   (DOUT),A
WAITO
```

The T_r function goes to SPMS allowing T_r to also go to SPAS as soon as ATN is no longer asserted. The latter happens when C goes to CSBS as follows:

```
        CMD92   EQU   11H          (The Command Field register is
                                     assigned port 11H in CTL.)
        GTSB    EQU   0F6H         (This is the local gts message.)
        PRTF    EQU   08H          (The TCI status register is
                                     assigned port 8H in CTL.)
        TCIF    EQU    0           (The TCI status bit number is 0.)
         LD   A,GTSB               (Send local message gts and wait
        OUT   (CMD92),A
L4       IN   A,(PRTF)              for it to be executed.  See note
        BIT   TCIF,A                below.)
         JR   NZ,L4
L5       IN   A,(PRTF)              Referred to as WAITX below.
        BIT   TCIF,A
         JR   Z,L5
```

Note:

Whenever an "operation command" is sent to the 8292, the TCI (Task Complete Interrupt) is set upon completion of the command (provided kt was masked on in the Interrupt Mask register). It remains set until shortly after a new command is sent, i.e., it takes $7t_{cy}$ to reset. At 4 Mhz this is equivalent to 26.25 microseconds. This is the reason for the first loop.

The camera SR function can now enter the APRS state and the camera device function can send the RQS STB message. The following code is used to receive this message.

```
        LON     EQU   40H          (The lon (listen only) local
                                     message.)
        BIM     EQU    0
        DIN     EQU   20H          (The Data in register is
                                     assigned to port 20H in CTL.)
         LD   A,LON                (This sends the local lon mess-
        OUT   (ADRMD),A             age causing the Listen function
                                    to go to the LADS (Listener
                                    Addressed state).
        XOR   A                    (Sends the local pon (power on)
        OUT   (AUXMD),0            message causing all Interface
                                    functions associated with the
                                    8291 to go to "ground" state.
                                    Note, however, that the lon mess-
                                    age remains active so that the
                                    Listen function goes back to
                                    LADS.  See note below.)
```

```
L6    IN   A,(INT1)
     BIT   BIM,A
      JR   Z,L6              Referred to as WAITI below.
      IN   A,(DIN)
```

Note:

When any message (in this case the RQS STB) has been "handshaken through" the bus, the byte in (BI) status bit of the Interrupt Status 1 register is set. It is reset upon being read.

The camera must now be taken out of the SR_r APRS state and T_r SPAS, SPMS states. Briefly, this is done as follows:

(a) The Controller sends the ATN message causing T_r to go to TADS.
(b) The local device function sends the message (via C), SPD (Serial Poll Disable) which, when handshaken through, causes the SPMS function to go to "ground state" SPIS.
(c) The camera should also go to the SR_r "ground state" NPRS. (Assuming the camera rsv message is no longer being sent.)

The coding is as follows:

```
TCSY  EQU  0FDH          (Take control synchronously
                          local message.)
SPD   EQU  19H           (Serial Poll Disable message.)
 LD   A,TCSY             (Send tcs and wait for state
OUT   (CMD92),A           change.  C goes to CACS via
WAITX                     CSHS, CSWS, CAWS.)

 LD   A,TON              (Since the 8291 was previously
OUT   (ADRMD),A           "configured" to be a listener,
XOR   A                   it must now be configured to
OUT   (AUXMD),A           be a talker.  Note again the
 LD   A,SPD               use of pon to establish the
OUT   (DOUT),A            correct "ground" state.)
WAITO
```

The next to last step is to bring the camera T_r function back to ground state (TIDS) since it is still in the TADS state. This is done by sending another talk address or equivalently the UNT message.

```
UNT   EQU  5FH
 LD   A,UNT
OUT   (DOUT),A
WAITO
```

The last step is to "clean up" the 8291-set interrupt flags. Specifically, the SRQ bit in the Interrupt Status register must be reset via the IACK command.

```
        IACK   EQU  ØBH          (This is the basic IACK command
                                  configuration.)
        SRQB   EQU  20H          (To be added to IACK to clear
                                  SRQ.)
        IBFBT  EQU   1           (Input buffer full bit for the
                                  Interrupt Status register.)
L7       IN   A,(INTST)          (Wait for any pending command
        BIT   IBFBT,A             to the 8292 to clear the input
         JR   NZ,L7               buffer.  This bit is reset
         LD   A,IACK+SRQB         when the buffer has been clear-
        OUT   (CMD92),A           ed by the 8292.  This step can
L8       IN   A,(INTST)           usually be omitted since a
        BIT   IBFBT,A             check for TCI is always done.
         JR   NZ,L8               IBF should clear within 24cy =
                                  90 microseconds at 4 Mhz.)
```

Note:

The last loop is to wait for the IACK command to clear. A TCI is not generated unless the error (ERR) bit is set. The SPI interrupt can also be checked for SRQ clear. See comments below.

Comments

In the previous section we have painstakingly gone through the relatively simple task of responding to a request for service. In doing so some of the relationships between "states" of the 8291-set and those of the various Interface functions have been at least implicitly described. As can be observed from the complete list of Registers available, and the contents thereof, there is much more to learn about the 8291-set. This can be done by a careful reading of the Intel Peripheral Design Handbook. Something should be stated about observation of bus errors however. In the call sequence given above it was implicitly assumed that everything went as planned. Since status polling rather than interrupts are used, if something goes wrong, for example, a message does not get handshaken through the bus, the program will hang up in a loop. Several error conditions may be checked for. These are discussed briefly below.

1) If the 8291 is in the talker active state (TACS) with a message to send and no listener is in the LACS state (the byte can not be handshaken onto the bus), the ERR bit in the Interrupt Status 1 register will be set.

2) There are various errors which are flagged by the 8292. These errors are first of all observed by the error bit, ERR, in the Interrupt Status register. The error type can then be determined by observing the contents of the Error Flag register as follows:

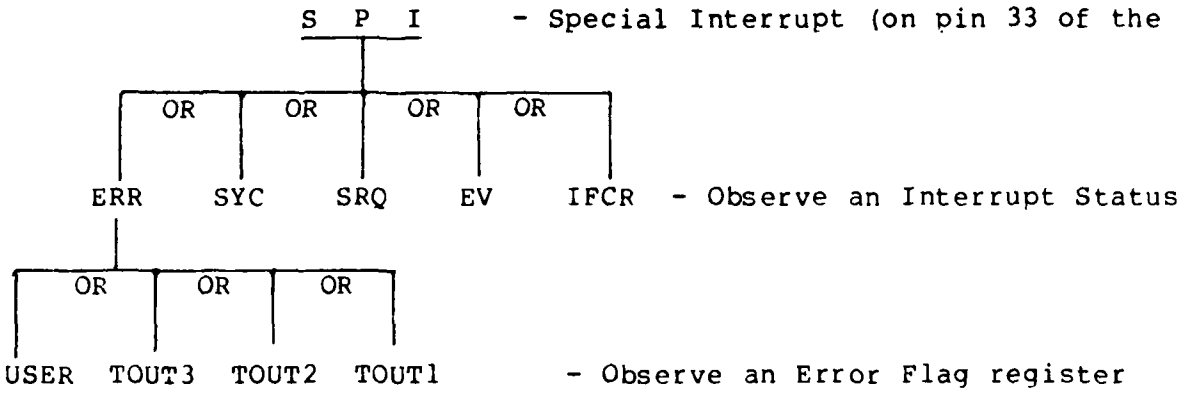
USER - the local message sic (send interface clear) or sre (send remote enable) is passed to the 8292 and it is not the System Controller (in state SIIS, SRAS).

TOUT1 - this "time out" error is generated, after a period of time determined by the contents of the Time Out register, when the local Controller function tries to go from CIDS to CADS when the TCT message is received from the bus and the current Controller in charge has not stopped sending the ATN message.

TOUT2 - a time out error which is generated when the local Controller goes to the CSBS state and the transmission between the addressed talker and listener(s) has not started.

TOUT3 - a time out error which occurs when the local Controller function tries to go from the CSBS state to the CSWS state (via CSHS) and does not do so because AH does not go to the ANRS state (the handshake is "stuck"). If this happens, an alternative is to issue the tca (take control asynchronously) local command which will force the Controller to the CSWS state (with possible loss of a data byte or, worse yet, a device message is interpreted as an Interface message because ATN is sent while a valid data byte is still on the bus.

The ERR "interrupt" in the Interrupt Status register is one member of a set of interrupts which, when ORed together, constitute an actual special interrupt (SPI). The following diagram is pertinent.



The CTL makes a check for TOUT3 and does a tca if such an error occurs. All other bus errors are taken care of by doing an external reset (with CTLRS). Further comments on this are made at the end of the section on Z-80 software.

SECTION II - CTL SOFTWARE

The CTL presently contains some preliminary software which

1) provides for sending data from CTL to one or more listeners (they may have extended addresses).
2) provides for the reception of data from a talker (which may have an extended address).
3) provides for the reception of a service request and subsequent parallel poll. (This function has been specialized to account for the idiosyncrasies of the Hamamatsu camera.)
4) provides for CTL to become controller in charge from another device previously in charge of the bus. Note that CTL cannot take part in data transfers until it is the controller in charge using the current software.
5) provides for the transfer of data between CTL and NIC. With the exception of a few instructions in PROM (which are always executed on external reset), this code is contained within the first 1K-bytes of RAM. It is transferred there from NIC with the help of the resident PROM software. Eventually, this code will be put into PROM (with a few modifications) after it has been thoroughly checked.

In broad outline, this software operates as follows:

a) on reset the PROM software does some preliminary 8291-set initialization, sets the stack pointer to BFF and then waits for a read-register, write-register or "boot" command from NIC. NIC sends this boot command and transfers 1K bytes of data (actually program) into locations 800H to BFFH. The boot program then transfers control to location 800H.
b) more initialization is done including an internal reset. If the SYC switch is on, the CTL becomes the system controller and controller in charge and sends out the IFC message to clear other devices on the bus. Otherwise it waits for TCL from the current Controller in charge of the bus.
c) following initialization, the program goes into the main "command" loop. Within this loop a command table is received from NIC. This is a table, twenty bytes in length, containing parameters for each command to be executed. The first item in the table is the number of the command to be executed.
d) at present five commands are implemented as follows:

COM1	Send data from CTL to the bus.
COM2	Receive data into CTL from the bus.
COM4	Receive service request and do serial poll.

COM5 Receive data into CTL from NIC. (Note that this subroutine is also used to read in each command table.)

COM6 Send data from CTL to NIC.

e) upon completion of each command, the command subroutine returns via way of a pseudo-subroutine which sends a "completion" message to NIC. CTL and NIC are "synchronized" by having access to the same command table which, among other information, contains the total number of bytes to be transferred from CTL to NIC and vice versa - assuming, of course, that nothing goes wrong.

A complete annotated listing of the current CTL software is included in Appendix B. It is largely a modification of software appearing in reference (2).

SECTION III - NIC SOFTWARE

Overview

An extensive library of NIC assembler language subroutines were written to be used in conjunction with CTL. A listing of them appears in Appendix C. Together, in conjunction with the NIC Demon/II operating system, they provide an operational system for use with CTL and, in particular, for obtaining and storing or displaying frames of pictures from the Hamamatsu camera.

Before describing these subroutines, it is instructive to review some of the characteristics of the NIC which have particular significance in the design of its software.

1) Main memory is segmented into 2000 (octal) word sections. References to (direct) addresses in an instruction are always relative to the page boundary. Thus, indirect addressing must be used to reference addresses off of the page. Thus, in particular, a given subroutine should usually not go across page boundaries. Also note that location 1777 (relative) on each page is used by the debugging subroutine NICBUG so should ordinarily not be referenced by a user program.

2) The word size of NIC is 20 bits. Since data from the CTL (and the camera) consists of 8 bit bytes, it is usually desirable to pack them 2 1/2 bytes per word for disk storage.

3) There is no linking loader so once a subroutine is assembled it is unrelocatable. The CTL associated software has been partitioned into 6 segments located (approximately) as follows:

SEG 1	0-1750
SEG 2	2020-2410
SEG 3	2420-2570
SEG 4	2605-3305
SEG 5	3310-3760
SEG 6	4010-4400

Each segment is assembled as a unit and external references (to other segments and NIC Demon/II modules) are given at the end of each segment. These references may have to be updated whenever other segments are changed.

Below, the contents of each segment is briefly described.

Software segments

SEG 1 (MAIN)

This segment contains the CAMERA MAIN program which is a user interactive program with 8 "modes" of operation having to do with interfacing with CTL and gathering data from the camera. It also contains five "command" subroutines, COM1, COM2, COM4, COM5, COM6 corresponding to the five available commands in CTL. In addition, there are a few interspersed local "service" modules used either by CAMERA MAIN or the COM subroutines.

CAMERA MAIN consists of two CTL/camera related tables, a short mode query section and 8 mode sections. The first table, running from 0-23 (all addresses are octal) is the command table referred to earlier in CTL software. The second table, running from 24-70 is a camera table which is used to keep track of the "state" of the camera and to provide camera secondary address values to be associated with mnemonics from camera formatting. See comments in CAMERA MAIN for a description of this table and its use for camera formatting.

Starting at entry point 71 is a short mode query section which asks the user to choose one of eight possible modes (of operation).

Mode 1

This mode provides for initializing CTL and camera. The mode query is FILE NAME?- to which the user should respond with the name of the file containing the program to be loaded into CTL. The user may reply with a non-existent file in which case whatever is in NIC memory starting a location 100000 is loaded into CTL. This feature allows the user to make changes in the Z-80 machine code before it is loaded (see mode 8). Note that the program exists in packed form (5 hex nibbles per NIC word) in NIC. After loading the file (if it exists), the camera table is initialized to the default values (see Table 6). Following this, the CTL reset instructions, CTLRS, is sent followed by the instruction causing the CTL program to be loaded. The program is then passed to CTL using subroutine UNPF which unpacks the contents of 100000 + ... into 8 bit bytes and sends them to CTL.

NAME	REL. ADDR.		COM1		COM2		COM4		COM5		COM6	
	OCT	HEX	NIC	CTL	NIC	CTL	NIC	CTL	NIC	CTL	NIC	CTL
COMN	0	0		X		X		X		X		X
NLIST	1	1		X		X			X		X	
LIS1P	2	2		X		X			(X)		(X)	
LIS1S	3	3		X		X			(X)		(X)	
LIS2P	4	4		X		X			(X)		(X)	
LIS2S	5	5		X		X			(X)		(X)	
LIS3P	6	6		X		X			(X)		(X)	
LIS3S	7	7		X		X			(X)		(X)	
TALKP	10	8				X		X	(X)		(X)	
TALKS	11	9				X		X	(X)		(X)	
EOSC	12	A		X		X			(X)		(X)	
NDAT	13	B		X		X			X	X	X	X
NDATB	14	C		X		X			X	X	X	X
DATAL	15	D		X		X				X		X
DATAH	16	E		X		X				X		X
MESS	17	F										
STAT1	20	10										
STAT2	21	11										
DUM1	22	12							X		X	
DUM2	23	13										

X => always used (X) => sometimes used for "immediate" data

COMN = Command number
NLIST = No. of listeners (or data storage indicator)
LIS1P = First listener primary address (or "immediate" data)
LIS1S = First listener secondary address (or "immediate" data),
etc. for LIS2, LIS3
TALKP = Talker primary address (or "immediate" data)
TALKS = Talker secondary address (or "immediate" data)
EOSC = End of data character
NDAT = No. of data bytes to be transferred
NDATB = No. of 256 byte data blocks to be transferred
DATAL = Low order byte of 2 byte address for start of data store
DATAH = High order byte of 2 byte address for start of data store
MESS = Not used at present
STAT1 = Not used at present
STAT2 = Not used at present
DUM1 = Pack/unpack data indicator
DUM2 = Not used at present

Table 6. - NIC/CTL Command Table

Mode 2

This mode provides for loading special "commands" into CTL and then executing them. The mode queries are COMMAND#?-, to which the user should reply with 7 or 8 and FILE NAME?- to which the user should reply with the name of the NIC file containing the module to be loaded. If the file does not exist, it is not loaded, rather the query is repeated. The following assumptions are made:

(1) the CTL command associated with 7 is to be loaded at C00H in CTL.
(2) the CTL command associated with 8 is to be loaded at E00H.
(3) the NIC commands corresponding to the CTL commands are already loaded in NIC.
(4) Each CTL module is a maximum of 512 (decimal) bytes in length.

At present, the CTL main program contains calls to COM7 and COM8 but the calling addresses are not available. Thus, the following changes should be "patched" into the program which is loaded under mode 1 (use mode 8).

```
location 84E  CD0000 => CD000C
location 851  CD0000 => CD000E
```

Mode 3

This mode provides for the transfer of CTL programs developed on other systems into NIC via way of a floppy disk based intermediary. Specifically, programs developed (and debugged) on a Tektronix 8002 microprocessor development system can be stored in a Sykes Comm-Stor communications storage unit which may then be used to transfer the program to NIC. The main advantage of the Comm-Stor for intermediate storage is its portability (neither the NIC nor Tektronix 8002 is very portable).

The mode query is FILE NAME?-, to which the user should reply with the name of the Comm-Stor file to be transferred to NIC. It is stored under the same name in NIC. The file on Comm-Storr is a direct copy of a Tektronix Hex file which is crated using the WHEX command. (See Appendix D for the format of this file and how it may be created.) It is basically the ASCII equivalent of the Z-80 machine code to be loaded into CTL. Before it is stored in NIC, the header and trailer information on each record is stripped off and the code is packed - 5 nibbles per NIC word. This file may be loaded into CTL using either mode 1 or mode 2.

Mode 4

This mode provides for formatting of the camera using the various secondary listener addresses provided for this purpose (see reference (5)). The mode query is MNEMONIC?-, to which the user should reply with one of 5 acceptable mnemonic codes:

OUT	-	output format	(1 or 2)
INF	-	input format	(1,2 or 3)
XCC	-	starting x-coordinate	(0000 to 1023)
INT	-	interlace number	(1,2 or 4)
MAR	-	marker on/off	(1 or 0)

The current format associated with this code is then printed. The user may reply with CR (carriage return) to accept the current value, or type in the new value desired. If a new value is typed in, it is stored in the camera table. This mode should be repeated for each format value to be sent to the camera.

Mode 5

This is the mode to be used when a "frame" of video data is to be obtained from the camera - either for display or for storage in NIC. The first mode query is FILE NAME?-, to which the user should reply with either the name of the NIC file under which the frame is to be stored or PRINT (or just PRI) if the frame is to be displayed rather than stored (see below for restrictions).

The second mode query is FRAME?-, to which the user should reply with VII, VII or VID for 1 line of video, incremented-video or decremented-video, respectively. In the incremented- and decremented-video camera modes, several video scans are gathered, starting at the current x-coordinate and ending at either x=1023 (incremented) or x=0 (decremented). The maximum number of scans is 1024 depending on values for XCO and INT.

Note that XCO must be set to the desired frame x-coordinate using Mode 4, prior to every use of VII or VID.

Whether the frame is actually stored or printed depends partly on the value of the camera input format (mnemonic INF). If the value is 1 or 3, the frame is never stored because the output under this format is ASCII; each pixel is represented by 4 characters, the last of which is a space (or possibly CR). In addition, if INF=3, a LF character is sent every 16 pixels. The maximum scan

length, if INF=1, is 256 pixels (1024 bytes) since this is the buffer space in CTL for storing a scan line at present. INF=3 will overflow the buffer at present so some data will be lost in this format. (There is no need to use it since in INF=1, the CR/LF is supplied by the display subroutine in NIC.

Thus, regardless of the file name given, if INF does not equal 2, the frame is printed only. If INF=2 and the file name is PRINT, each pixel is displayed as two ASCII characters representing the hexadecimal value of the pixel. There is no space between characters and 32 pixels are displayed on one line (versus 16 if INF=1 or 3).

Each line of video is gathered using the command sequence:

COM1 - send command to camera to make the next video sweep.
COM4 - wait for service request - do serial poll.
COM5 - transfer a scan line from the camera to CTL.
COM6 - transfer the scan line from CTL to NIC.

It has been found, experimentally, that a variable length pause must be inserted between the end of serial poll (COM4) and the transfer of data to CTL (COM5). The length of pause required depends on the length of each scan line. (Thus, it appears that the SRQ message from the camera is asserted before a scan has been completed.)

It also should be noted that the serial poll response (RQS STB) for each line of video is printed regardless of whether the line is to be stored or printed. This can be used to indicate that a frame is indeed being obtained.

Mode 6

This mode is used to transfer part of CTL memory (either PROM or RAM) to NIC (starting at location 100000). The mode query is HEXN-, to which the user should reply with the 4 digit hexadecimal starting address of memory to be transferred followed by a single digit representing the number 256 (decimal) byte locks to be transferred. For example, 08004 will transfer 4 blocks of memory starting at hex 800. The contents of memory may then be displayed by using mode 7. (The number of bytes to be displayed is saved for use by mode 7.)

This mode is useful for examining the contents of CTL memory or for storing it for modification using mode 8. (Note that within NIC the bytes are stored in packed form.)

Mode 7

This mode is used to display in hexadecimal form (5 nibbles/NIC word), the contents of NIC memory starting at location 100000 at 32 characters/line. (This corresponds to one record in the Tektronix Hex file.) This mode is designed to be run after mode 3 or mode 6 in which case the number of bytes (=2 nibbles) to be displayed is provided by these modes. Otherwise, the user should use NICBUG to insert the number of bytes at the location labeled NBY5. (This can be found by looking at the current symbol table for SEG1.)

Mode 8

This mode provides the capability of relatively easily changing the contents of CTL memory. (Note that, at present, all instructions except the "boot" program are in RAM.) The technique is to

(1) bring the program to be modified into NIC by using either mode 3 or mode 6. Alternatively, the program can be loaded directly from NIC disk using the DEMON/II LOAD instruction - assuming the program is stored there.

(2) examine and change using mode 8.

(3) put the modified program into CTL using mode 1 or possibly mode 2 (the contents should be put in a NIC file first if mode 2 is used).

The mode query is OCT-, to which the user should reply with the (up to 7 digit) octal starting address of NIC memory to be examined and possibly changed. (The address given is right justified zero-fill. Thus, 100=0000100. Usually the address will be 100000 plus.)

The contents of memory at this location is then displayed as 5 hexadecimal digits (5 nibbles in each word). The user may respond with

(a) space (actually any sequence of 1 to 4 characters) followed by CR.

(b) 5 characters followed by CR.
(c) CR with no preceding character.

If (a), then the next word is displayed with no change to the current word.

If (b), the current word is changed to the value given and the next word is displayed. (If the value is not hexadecimal, then the changed word is "unpredictable".)

If (c), then the mode is exited with no change to the current word.

The following "escapes" are provided in CAMERA MAIN.

(1) If $\hat{Q}$ (control Q) is typed by the user while the "user monitor" is waiting for input from the user, a jump to NICBUG is made. (Presumably it is loaded when the other modules are loaded.)

(2) If $\hat{G}$ is typed under the same circumstances, the current operation is stopped and CAMERA MAIN is restarted (and MODE?- is typed). This is the usual escape for a mistyped character.

(3) If <u>any</u> character is typed when the "CTL monitor" is waiting for input from the CTL, the program aborts to NICBUG.

(4) In mode 3, $\hat{Z}$ must be typed to escape from the "no find" condition, i.e., if the file name given by the user does not exist on the Comm-Stor diskette.

COM1, COM2, COM4, COM5, COM6

These 5 subroutines are the counterparts to the 5 command subroutines currently in CTL. The flow of control is from NIC COMn to CTL COMn as follows:

(a) In the main program, e.g., CAMERA MAIN, insert appropriate parameters in the command table (see Table 6 for the structure of this table and what parameters are needed for the various commands.
(b) Insert the command number in the first table location.
(c) Call the appropriate NIC command.
(d) Within the NIC command, the first step is usually to pass the command table to CTL using a subroutine called WCTL.
(e) "Close" this operation by calling the NIC "CTL monitor".
(f) Perform whatever operations are required by this command.
(g) "Close" the command by again calling the NIC "CTL monitor".

Miscellaneous Subroutines

The following auxiliary subroutines are imbedded within SEG1.

- COMSTO - read characters from Comm-Stor and look for Z for end-of-file.
- FILEQ - type message "FILE NAME?-".
- TYPE1 - converts a 4 bit (nibble), left justified, to 8 bit NIC ASCII and type it. (E.g., 1010 ..... => 301_8 = typed "A".)
- NIB - packs up to 5 user typed ASCII hex characters into a NIC word as 4 nibbles per character. These are left justified, zero fill. In addition, the number of nibbles (determied by CR) packed is returned.
- ECHO1 - echoes user typed characters (via ECHO) and sets the byte counter in PAKF to 1 on the reception of CR.
- SCTL - sends bytes from NIC to CTL with two possible modes: (1) as it appears in the accumulator (ACC); (2) adding 60 octal to the value in ACC. This changes binary integers to ASCII integers.

SEG2 (ERMON)

This segment contains auxiliary subroutines directly related to the transmission of data between NIC and CTL.

- ERROR - Under certain conditions, CTL can detect a bus error. Whenever this occurs, rather than complete a command, a jump is made to a CTL error routine which is "matched" by ERROR in NIC. An error is signaled to MONITOR (see below) through the 9th bit (the service bit) which is set on error. The CTL error routine sends a status byte and the 2 byte program counter (PC) of the subroutine which incurred the error. (NIC) ERROR prints the status and DC bytes, the number of the command being executed and the NIC program counter associated with the error.
  ERROR jumps to the DEMON/II operating system upon completion.
- MONITOR - Reads data into NIC from CTL. It also looks for the service bit and jumps to ERROR when it occurs. In addition, it monitors the console (TTY) for input and jumps to NICBUG when an input is detected.
- WCTL - Transfers a block of unpacked data from NIC to CTL. This subroutine is used mainly to transfer the command table from NIC to CTL.

RCTL - Used by ERROR to read bytes from CTL to NIC since MONITOR cannot be used here.

CTLTST - This subroutine written by Dave Wright at the University of Illinois, is the mate to his "boot" program. It can be used to read or write data into the various CTL ports. By using CTLTST, it is possible to direct all bus operations (very inefficiently) directly from NIC rather than indirectly through resident CTL modules. It should be pointed out, however, that with this algorithm it is not possible to read the port (08H) which contains the TCI status bit.

ECHO - Input and echo console input. Checks for escape characters G and Q.

VALID - Transforms hex ASCII (NIC) to NIC binary, with error exit for non-valid characters.

HEXT - Right justified 8 bit subword as two hex characters.

SEG3 (IOSUB)

This segment contains the I/O subroutines used for storing information on the disk. Use is made of the DEMON/II modules DIRFUN for directory manipulations and DISK for actually reading and writing the disk. (DEMON/II is the disk operating system supplied by Nicolet.)

OPENW - Opens a file for write by locating the next available space (given the size of the file to be stored) or the first track after the last file stored.

OPENR - Opens a file for read by returning the starting track and file size.

CLOSE - Closes a file just written by adding the file name and other parameters to the directory.

WRITE - Write one or more records from the buffer to the disk (here a record is one track).

READD - Read one or more records from the disk to the buffer (assumed to start at address 100000).

DIRFIN - Swap locations 3000-7600 for directory operations (reads directory into core).

DIROUT - The opposite swap to DIRFIN.

SEG4 (PACK)

This segment, for the most part, contains subroutines which are involved in the packing and unpacking of data.

PRTOCT - Prints the octal value of the contents of ACC.

UNP - Unpacks packed ASCII text and prints it. (Here one word contains 3, 6 bit characters, right justified.) This is used extensively to print messages which are stored using the TEXT pseudo-op in the Disk Editor.

TYPE - Type one character. (This is a 4 line routine used by many other modules.)

CRLF - "Prints" carriage return, line feed.

UNPF - Transfers (and possibly unpacks) data from core to another location via a subroutine, the address of which is passed to UNPF. The packed data can be either 5 nibbles/word or 2.5 bytes/word.

PAKF - This is the inverse of UNPG except nibble packing is not done. (See DEC for this.)

PKR - This is the inverse of UNP except that it packs characters into 2 words only. It is used mainly to store file names from characters typed by the user.

SEG5 (TEKX)

This segment contains the three major subroutines used with mode 3 for transfering "WHEX" files stored on Comm-Stor to packed Z-80 machine code files for use in CTL.

DEC - Transforms (in place) a packed (2.5 bytes/word) ASCII hex string to packed nibbles (5 nibbles/word). This a a highly "subroutine interactive" module wherein, the state of the subroutine can be observed on exit.

NIBBIN - Converts packed BCD or BCH (binary coded hexadecimal) to binary.

TEKHEX - Converts a Tektronix Hex file (stored in 2.5 byte/word packed form in NIC) into the binary file (5 nibbles/word) to be used by CTL. (Note that when the file is read into CTL it is unpacked - 2 nibbles at a time are passed to CTL.) The transformation is mainly that of stripping off the header and trailer information and doing a checksum on the remainder.

SEG6 (MISCL)

This segment contains various "miscellaneous" subroutines.

NICFIL - Creates a file by transfering data from the buffer (starting at 100000) onto the disk and adding the file name to the directory. That is, the OPENW, WRITE and CLOSE operations are done. This subroutine should be used only if the complete file can be stored in the buffer (8192 20 bit words).

SEARCH - Search and replace the contents of the camera table.

ZERTAB - Zeroes the command table.

MULTP - Integer multiply two 20 bit values and place the low order bits in ACC. The high order bits are stored in the MQ (multiplier-quotient) register.

DIVDE - Integer divide either a 40 bit or 20 bit value by a 20 bit value to obtain a 20 bit value plus remainder. Note that DIVDE is designed to follow MULTP but can be used alone provided that MQ is set to zero first.

SENDF - Prints the contents of ACC (assumed to be a decimal or octal integer number) after conversion to ASCII by adding 260 octal, or as two ASCII hexadecimal characters. In addition, carriage return, line-feed is done after a user specified number of bytes have been printed.

GETFIL - Obtains a file and stores it in the buffer area (the inverse of NICFIL).

References

The following is a short, annotated list of references containing information specific to topics discussed in the body of this report or in the appendices.

(1) Z-80

There are numerous references describing the Z-80 uP and its programming.

Mostek Corp., 1979 Microcomputer Data Book, p.75-164.
This reference contains a detailed technical description of the Z-80 (and Z-80A), a listing of OP codes in "Zilog nmemonics", and some programming examples.

Barden, W., The Z-80 Microcomputer Handbook, Howard W. Sams & Co., Inc., 1978.
Discusses Z-80 hardware, software and some Zilog Z-80 Microcomputers.

(2) 8291/8292/8293 (8291-set)

Intel Corp., Peripheral Design Handbook, Aug. 1980.
8291 p.1-199 to 1-224
8291 p.1-225 to 1-238
8293 p.1-239 to 1-251
Using the 8292 GPIB Controller p.2-187 to 2-239
This is the reference used for the "8291-set" during design and implementation of the CTL. All that is "known" about this chip set is contained in this reference.

(3) GPIB (IEEE-488)

IEEE, IEEE Standard Digital Interface for Programmable Instrumentation, 1980. (Available from IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854.)
This is the official American standard for the interface. It is quite technical and difficult to comprehend on first reading.

Philips, N.V., Digital Instrument Course, Part 4 IEC Bus Interface, N.V. Philips Gloeilampenfabriken, Test and Measuring Dept., Eindhoven, The Netherlands. Publication No. 9498.829.00311 ($8.00).
This is a readable discourse on the GPIB. (Note that the European equivalent to the IEEE-488 standard is the IEC 625-1. It differs from the IEEE-488 essentially only in the connector type used.)

Hewlett-Packard, Tutorial Description of the Hewlett-Packard Interface Bus.
This is an elementary tutorial on the GPIB (which HP calls HP-1B). It contains a quite complete and up-to-date bibliography.

(4) Nicolet 1080

Nicolet, Programming the Nicolet 1080 Stored Program Computer, NIC-80/S-7111-M. Nicolet Instrument Corp., 5225 Verona Rd., Madison, WI 53711.
This is the standard source of information on programming the 1080 in assembler language.

Nicolet, DEMON/II Disk Executive Monitor for the Nicolet 294 Disk System, 1973.
Describes the disk storage system and software available for reading and writing the disk as well as some simple utility programs such as STORE file and LOAD file.

Nicolet, Integrated Monitor Package for DEMON/II, 1974.
Describes higher level utility programs such as the Disk Editor, Disk Assembler and Disk Loader.

Nicolet, Programmed Data Transfers, NIC-80/X-7113-D.
This Nicolet document describes how to interface to the Nicolet 1080 via the 80 pin I/O connector.

(5) Hamamatsu GPIB Interface

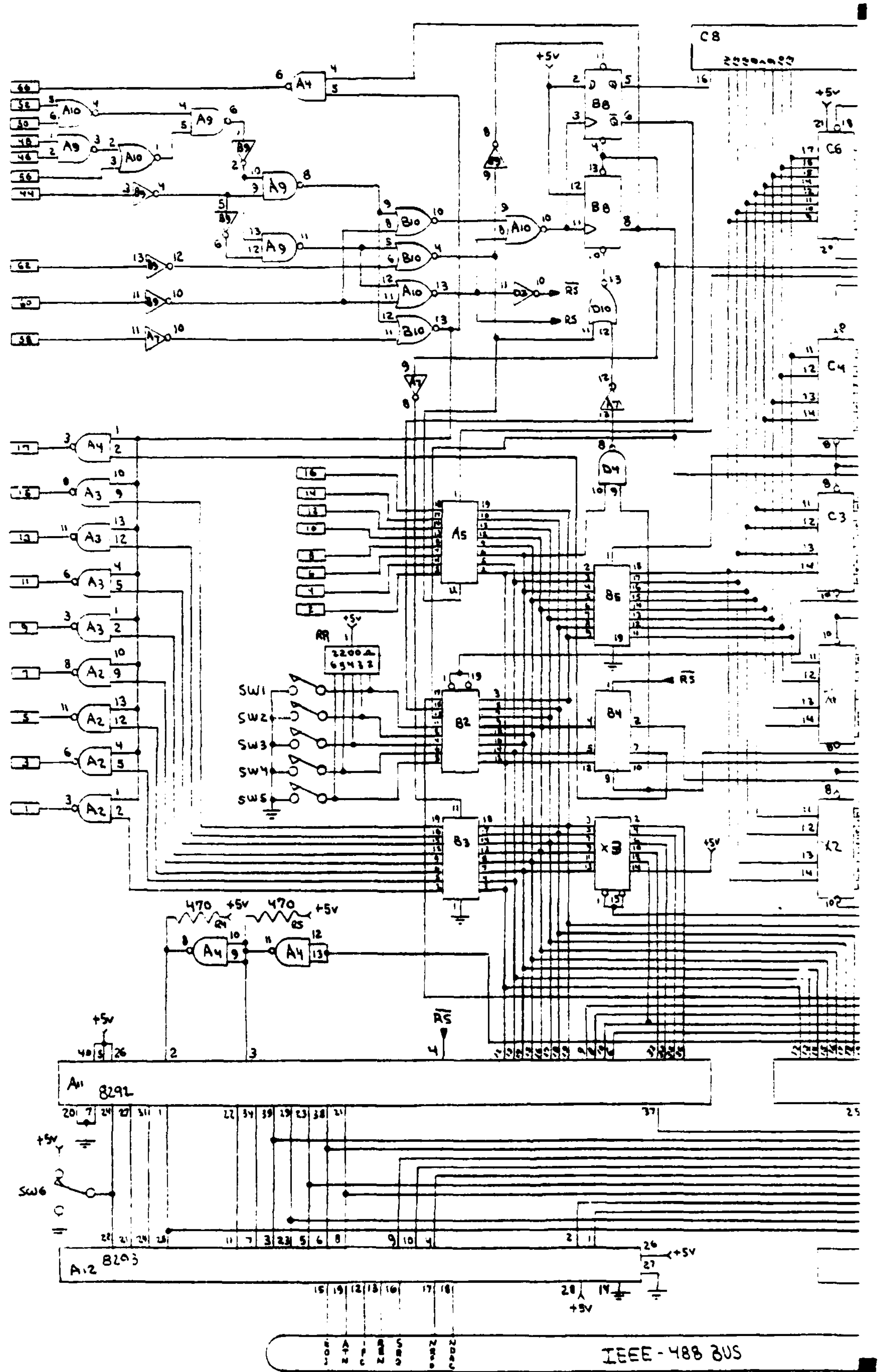
Hamamatsu, M999-04 General Purpose Interface Bus. An IEEE-488 Standard Interface for the C1000 Camera, 1977, Hamamatsu Systems, Inc., 332 Second Ave., Waltham, MA 02154.
This reference provides the GPIB message protocol used by the Hamamatsu interface and describes the Interface functions which have been implemented.

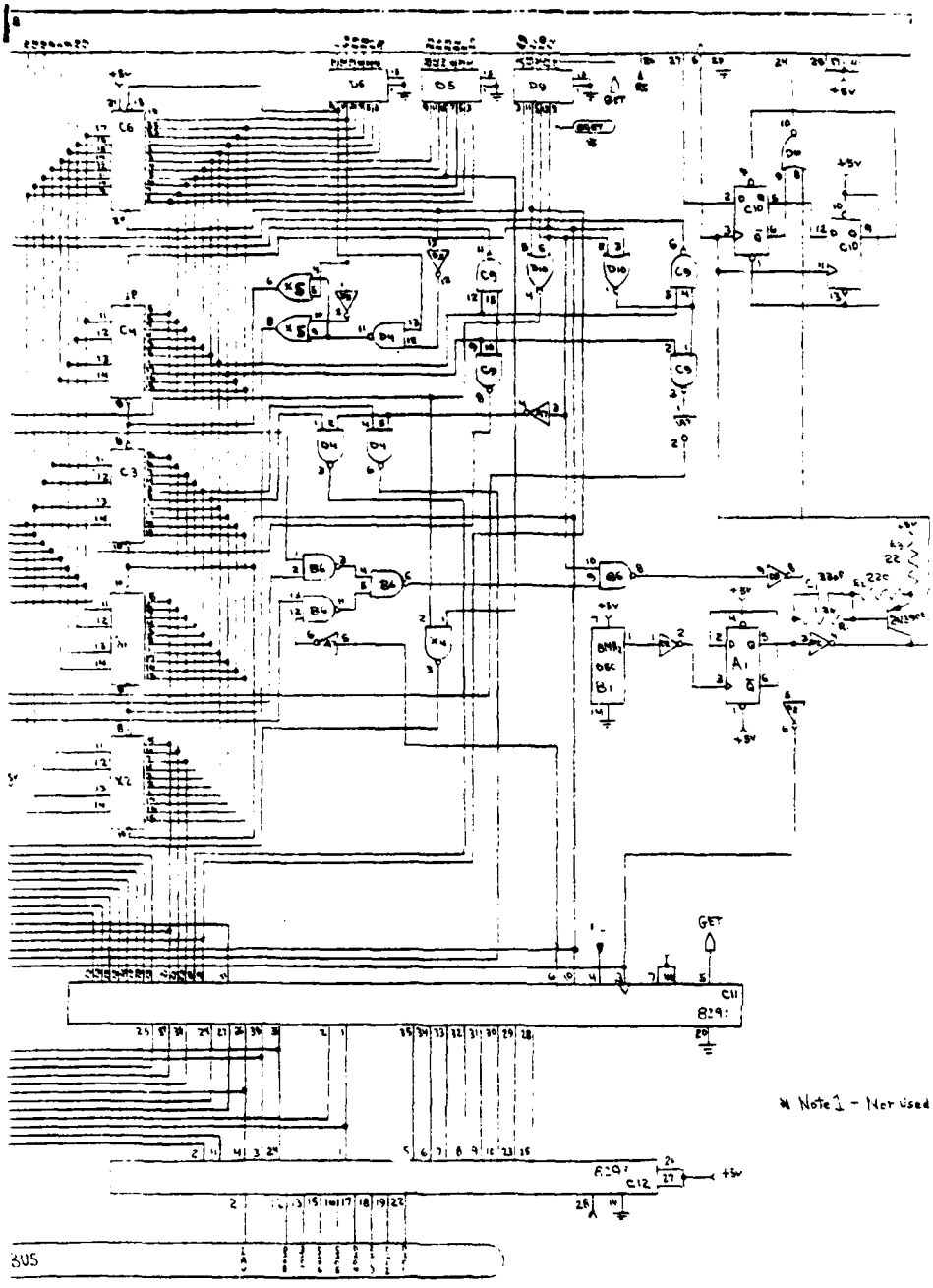
APPENDIX A - CTL Hardware

(a) Schematic

(b) Component Layout

(c) Wire-Run List

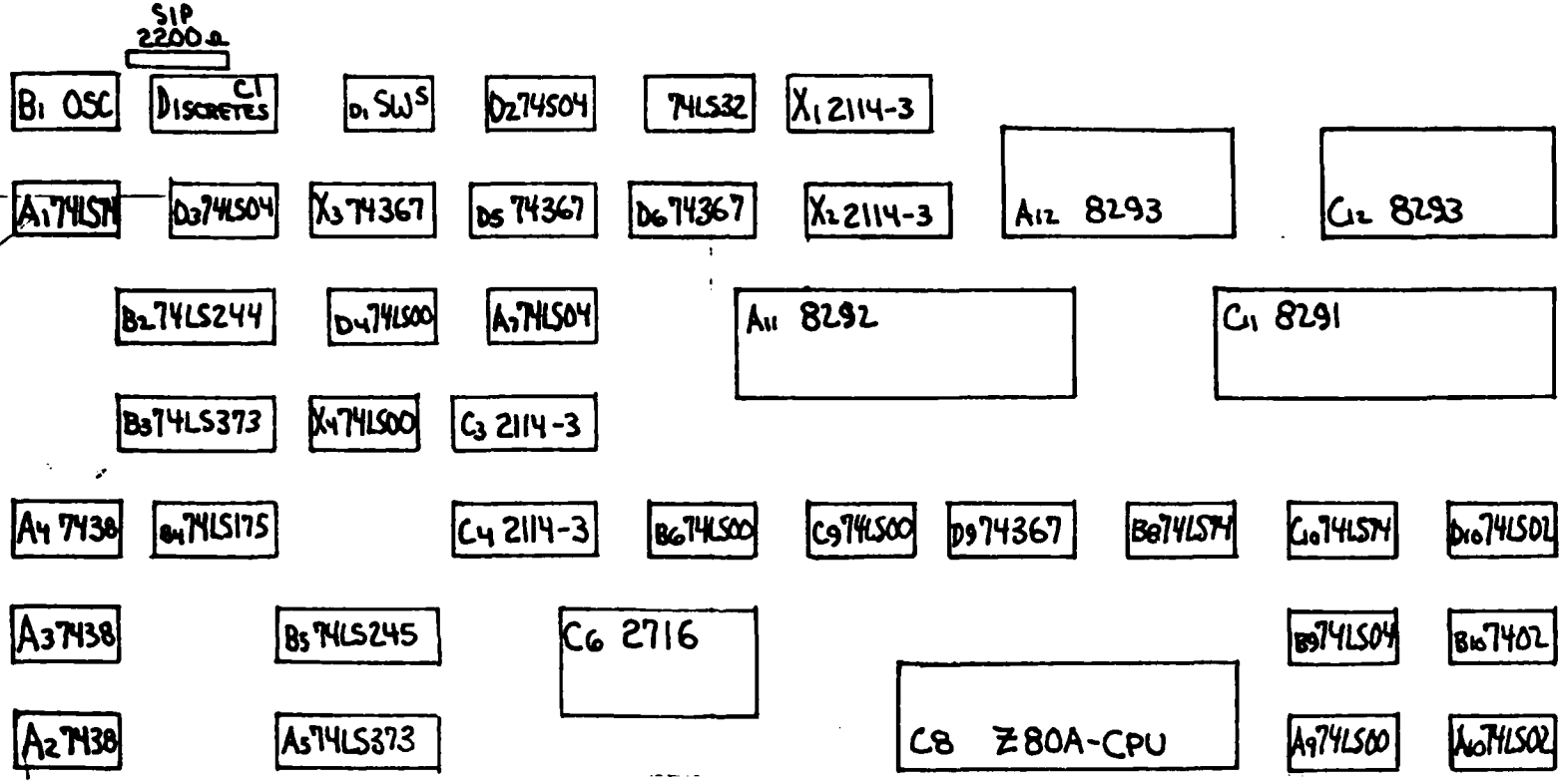

C8
C6
C4
C3
A4
A9
A10
B9
B10
B8
D10
D4
A5
B5
B2
B4
B3
X3
X2
A3
A2
RR
SW1
SW2
SW3
SW4
SW5
SW6
A11 8292
A12 8293
+5V
IEEE-488 BUS


GET
C11
8291
C12
8291
+5V
BUS
* Note 1 - Not Used

ANY DIP PLUGBORD

COMPONENT SIDE LAYOUT PAPER

NOTICE: While this circuitry exists a small percentage of the holes may have solder blockage. This is USUALLY ... typical easily ... holes by component leads. In some cases, a soldering iron may be required.



THIS AREA ... FOR REGULATORS ... AGAINST BOARD ... AT BOTTOM ...

Wire-Run List

A1P1-A1P4-A1P14-Vcc
A1P2-A1P6
A1P3-D2P2
A1P5-D2P3-D2P5
A1P7-Gnd

A2P1-A2P4-A2P10-A2P13-A3P1-A3P4-A3P10-A3P13-A4P1-A4P5-B10P13
A2P2-B3P2
A2P3-NICBUS 1
A2P5-B3P5
A2P6-NICBUS 3
A2P7-Gnd
A2P8-NICBUS 7
A2P9-B3P9
A2P11-NICBUS 5
A2P12-B3P6
A2P14-Vcc

A3P2-B3P12
A3P3-NICBUS 9
A3P5-B3P15
A3P6-NICBUS 11
A3P7-Gnd
A3P8-NICBUS 15
A3P9-B3P19
A3P11-NICBUS 13
A3P12-B3P16
A3P14-Vcc

A4P2-B4P7
A4P3-NICBUS 17
A4P4-B2P13-B6P1-B8P8
A4P6-NICBUS 66
A4P7-Gnd
A4P8-A11P2
A4P9-A4P10-A4P11-A11P3
A4P12-A4P13-C11P3-D2P6
A4P14-Vcc

A5P1-C9P11
A5P2-A11P12-B2P12-B3P3-B4P12-B5P2-C11P12
A5P3-NICBUS 2
A5P4-NICBUS 4
A5P5-A11P13-B2P14-B3P4-B4P5-B5P3-C11P13
A5P6-A11P14-B2P16-B3P7-B5P4-C11P14-D4P10-X3P13
A5P7-NICBUS 6
A5P8-NICBUS 8
A5P9-A11P15-B2P18-B3P8-B5P5-C11P15-X3P11
A5P10-Gnd
A5P11-B9P9-B10P11-D10P11
A5P12-A11P16-B2P9-B3P13-B4P4-B5P6-C11P16-X3P9
A5P13-NICBUS 10
A4P14-NICBUS 12
A5P15-A11P17-B2P7-B3P14-B5P7-C11P17-X3P7
A5P16-A11P18-B2P5-B3P17-B5P8-C11P18-X3P5
A5P17-NICBUS 14
A5P18-NICBUS 16
A5P19-A11P19-B2P3-B3P18-B5P9-C11P19-X3P3
A5P20-Vcc

A7P1-C9P3
A7P2-B4P9-D4P10
A7P3-B6P10-D9P5-D10P2-D10P6
A7P4-D4P2-D4P5
A7P5-C11P6
A7P6-B6P12
A7P7-Gnd
A7P8-B3P11
A7P9-B8P4-B8P13-C9P6
A7P10-B10P11
A7P11-NICBUS 58
A7P12-D10P12
A7P13-D4P8
A7P14-Vcc

A9P1-NICBUS 48
A9P2-NICBUS 46
A9P3-A10P2
A9P4-A10P4
A9P5-A10P1
A9P6-B9P1
A9P7-Gnd
A9P8-B10P9-B10P12
A9P9-B9P4-B9P5
A9P10-A9P13-B9P2
A9P11-A10P12-B10P5
A9P12-B9P6
A9P14-Vcc

A10P3-NICBUS 56
A10P5-NICBUS 52
A10P6-NICBUS 50
A10P7-Gnd
A10P8-A10P13-C11P4-D3P11
A10P9-B10P10
A10P10-B8P3-B8P11
A10P11-B9P10-B10P8
A10P14-Vcc

A11P1-A12P25-C12P12
A11P4-B4P1-C8P26-D3P10
A11P5-A11P26-A11P40-Vcc
A11P6-D4P6
A11P7-A11P20-Gnd
A11P8-C11P9-D9P11-D10P5
A11P9-C3P5-C4P5-C11P21-D5P3-X1P5-X2P5-C6P8
A11P10-C3P10-C4P10-C11P10-D9P13-D10P3-X1P10-X2P10-X3P12
A11P21-A12P8-C11P27
A11P22-A12P11
A11P23-A12P5-C11P24
A11P24-A12P22-SW6
A11P27-A12P21
A11P29-A12P23-C12P11
A11P31-A12P24

A11P32-X3P10
A11P33-X3P6
A11P35-X3P4
A11P36-X3P2

A11P34-A12P7
A11P37-C11P36-C12P24
A11P38-A12P6-C11P25
A11P39-A12P3-C11P39-C12P3
A12P1-C11P1-C12P1
A12P2-C11P2
A12P4-C11P26-C12P4
A12P9-C11P37
A12P10-C11P38
A12P12-IEEE488
A12P13-IEEE488
A12P15-IEEE488
A12P16-IEEE488
A12P17-IEEE488
A12P18-IEEE488
A12P19-IEEE488
A12P26-Vcc-A12P28
A12P27-Gnd-A12P14

```
B1P1-D2P1
B1P7-Gnd
B1P14-Vcc

B2P1-B2P19-C9P8
B2P2-RP1P3-SW2
B2P4-RP1P4-SW3
B2P6-RP1P5-SW4
B2P8-RP1P6-SW5
B2P10-Gnd
B2P11-RP1P2-SW1
B2P15-B8P6
B2P17-C11P11
B2P20-Vcc

B3P1-B3P10-Gnd
B3P20-Vcc

B4P2-B6P13
B4P10-B6P2
B4P8-Gnd
B4P16-Vcc

B5P1-C9P10-C9P13-D10P4-X4P2
B5P10-Gnd-B5P19
B5P11-C4P11-C8P13-C6P17-X1P11
B5P12-C4P12-C6P16-C8P10-X1P12
B5P13-C4P13-C6P15-C8P9-X1P13
B5P14-C4P14-C6P14-C8P7-X1P14
B5P15-C3P11-C6P12-C8P8-X2P11
B5P16-C3P12-C6P11-C8P12-X2P12
B5P17-C3P13-C6P10-C8P15-X2P13
B5P18-C3P14-C6P9-C8P14-X2P14
B5P20-Vcc

B6P3-B6P4
B6P5-B6P11
B6P6-B6P9
B6P7-Gnd
B6P8-D3P9
B6P14-Vcc

B8P1-B9P8
B8P2-B8P12-B8P14-Vcc
B8P5-C8P16
B8P7-Gnd
B8P10-D10P13
```

```
B9P3-NICBUS 44
B9P7-Gnd
B9P11-NICBUS 60
B9P12-B10P6
B9P13-NICBUS 62
B9P14-Vcc

B10P7-Gnd
B10P14-Vcc

C3P1-C4P1-C6P2-D5P3-X1P1-X2P1-C9P5-C9P12
C3P2-C4P2-C6P3-D5P9-X1P2-X2P2-D4P1
C3P3-C4P3-C6P4-D5P11-X1P3-X2P3-D4P4
C3P4-C4P4-C6P5-D5P13-X1P4-X2P4-X4P1
C3P6-C4P6-C6P7-D5P5-X1P6-X2P6-C11P22
C3P7-C4P7-C6P6-D5P7-X1P7-X2P7-C11P23-C9P2-C9P9
C3P8-C4P8-X5P6
C3P9-Gnd
C3P15-C4P15-C6P22-D6P13-X1P15-X2P15
C3P16-C4P16-C6P23-D6P7-X1P16-X2P16
C3P17-C4P17-C6P1-D6P5-X1P17-X2P17
C3P18-Vcc

C4P9-Gnd
C4P18-Vcc

C6P12-Gnd
C6P18-D4P13-D6P9
C6P19-D3P1-D6P9-X5P5
C6P20-D3P13-D9P3
C6P21-C6P24-Vcc

C8P1-D6P10
C8P6-C10P3-C10P11-2N3906[C]
C8P11-C8P17-C8P25-Vcc
C8P19-D9P2
C8P20-D9P4
C8P21-D9P12
C8P22-D9P14
C8P24-D10P10
C8P27-C10P2
C8P29-Gnd
C8P30-D5P2
C8P31-D5P4
C8P32-D5P6
C8P33-D5P14
C8P34-D5P12
C8P35-D5P10
C8P36-D6P2
C8P37-D6P4
C8P38-D6P6
C8P39-D6P14
C8P40-D6P12
```

```
C9P1-C9P4-D10P1
C9P7-Gnd
C9P14-Vcc

C10P1-C10P10-C10P13-C10P14-Vcc
C10P4-C10P9
C10P5-C10P12
C10P6-D10P9
C10P7-Gnd
C10P14-Vcc

C11P5-D9P10
C11P7-C11P40-Vcc
C11P8-D4P3
C11P20-Gnd
C12P28-C12P25
C11P29-C12P23
C11P30-C12P10
C11P31-C12P9
C11P32-C12P8
C11P33-C12P7
C11P34-C12P6
C11P35-C12P5

C12P12-IEEE 488
C12P13-IEEE 488
C12P14-Gnd
C12P15-IEEE 488
C12P16-IEEE 488
C12P17-IEEE 488
C12P18-IEEE 488
C12P19-IEEE 488
C12P21-IEEE 488
C12P22-IEEE 488
C12P26-C12P27-C12P28-Vcc

D2P7-Gnd
D2P14-Vcc

D3P2-X5P10
D3P7-Gnd
D3P8-D10P8
D3P12-D4P12
D3P14-Vcc

D4P7-Gnd
D4P11-X5P4-X5P9
D4P14-Vcc

D5P1-D5P8-D5P15-Gnd
D5P16-Vcc
```

D6P1-D6P8-D6P15-Gnd
D6P16-Vcc

D9P1-D9P8-D9P15-Gnd
D9P16-Vcc

D10P7-Gnd
D10P14-Vcc

X1P8-X2P8-X5P8
X1P12-Gnd
X1P24-Vcc

X2P12-Gnd
X2P24-Vcc

X3P1-X3P15-X4P3
X3P8-Gnd
X3P14-X3P16-Vcc

X4P7-Gnd
X4P14-Vcc

X5P7-Gnd
X5P14-Vcc

APPENDIX B - CTL Software

GENERAL PURPOSE PROM CODE LISTING

```
          ;  GENERAL PURPOSE PROM CODE DAN TERPSTRA 7/19/80
          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
          ;
          ;    THIS CODE IS DESIGNED TO BE BURNED INTO A ROM
          ;  TO PROVIDE 3 GENERAL PURPOSE ROUTINES FOR THE
          ;  NIC-488/CTL NICOLET-1080 TO IEEE-488 BUS INTERFACE
          ;  THE 3 COMMANDS ARE:
          ;    READ:   XXXX1XXX BINARY
          ;            ACCEPT COMMAND BYTE FROM NICP.  TREAT
          ;            IT AS PORT ADDRESS.  READ THIS PORT
          ;            AND SEND CONTENTS TO NICP.
          ;    WRITE:  XXXX0XXX BINARY
          ;            ACCEPT COMMAND BYTE AS PORT ADDRESS.
          ;            ACCEPT SECOND BYTE AS CONTENTS.
          ;            SEND SECOND BYTE TO PORT ADDRESSED
          ;            IN FIRST BYTE AND SET DONE.
          ;    BOOT:   00000000 BINARY
          ;            SET DONE TO ACKNOWLEDGE RECEIPT OF
          ;            COMMAND.  ACCEPT NEXT 1024 BYTES IN
          ;            NICP AND STORE IN RAM FROM 800H TO
          ;            BFFH.  TRANSFER CONTROL TO NEWLY
          ;            LOADED PROGRAM AT 800H.  STACK IS
          ;            DESTROYED.
          ;    IF USED ON THE NICP, CTRLI, OR CTRLO PORTS,
          ;  THE READ AND WRITE COMMANDS MAY PRODUCE HARMLESS
          ;  BUT INCORRECT RESULTS, SINCE THESE PORTS ARE MOD-
          ;  IFIED ON EXECUTION OF THE ROUTINES.
          ;    THIS CODE ALSO CONTAINS AN INITIALIZATION
          ;  ROUTINE THAT TURNS ON ALL 8291 AND 8292 MASKS, AND
          ;  SETS THE DEVICE ADDRESS FROM THE USER-SETTABLE
          ;  SWITCHES.  AFTER RESET, THE CTL WILL BE THE ACTIVE
          ;  CONTROLLER-IN-CHARGE, AND WILL BE IN A TALK-ONLY
          ;  STATE.  THE USER MAY FIND IT NECESSARY TO INITIAL-
          ;  IZE OTHER REGISTERS OUTSIDE OF THIS ROUTINE FOR A
          ;  SPECIFIC APPLICATION.
          ;
          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
          ;
          ;
0000'               ORG      0
          ;
          ;    CTL COMMAND EQUATES
  (0001)  DNEWT:          EQU      01       ;WAIT-ON-DONE COMMAND
  (0006)  BUSY:           EQU      06       ;BUSY BIT IN CTRLO
  (0002)  TLRST:          EQU      02       ;TALKER/LISTENER RESET
  (00F2)  CRST:           EQU      0F2H     ;CONTROLLER RESET
```

```
                0047 ;
                0048 ;     CONTROLLER PORT ASSIGNMENTS
(0010)          0049 CONT0:       EQU     10H     ;BASE ADDRESS FOR CONTROLLER
(0011)          0050 CONT1:       EQU     CONT0+1
                0051 ;
                0052 ;     TALKER/LISTENER PORT ASSIGNMENTS
(0020)          0053 TL0: EQU     20H     ;BASE ADDRESS FOR TALKER LSTENER
(0021)          0054 TL1: EQU     TL0+1
(0022)          0055 TL2: EQU     TL0+2
(0023)          0056 TL3: EQU     TL0+3
(0024)          0057 TL4: EQU     TL0+4
(0025)          0058 TL5: EQU     TL0+5
(0026)          0059 TL6: EQU     TL0+6
(0027)          0060 TL7: EQU     TL0+7
                0061 ;     CTL PORT ASSIGNMENTS
(0040)          0062 NICP:EQU     40H     ;BIDIRECTIONAL NICOLET INTERFACE
(0080)          0063 CTRLO:EQU    80H     ;CONTROL OUTPUT PORT
(0080)          0064 CTRLI:EQU    80H     ;CONTROL INPUT PORT
                0065 ;
                0066 ;     CTL MEMORY LOCATIONS
(0800)          0067 RAM:         EQU     0800H   ;FIRST ACTIVE RAM ADDRESS
(0BFF)          0068 RAMTOP:      EQU     0BFFH   ;LAST ACTIVE RAM ADDRESS
                0069 ;
                0070 ;
                0071 ;  COLD START ENTRY POINT
                0072 ;SETS UP THE STACK AND ENTERS THE COMMAND DECODER
                0073 ;
0000  21FF0B    0074 START:LD     HL,RAMTOP       ;GET TOP ADDRESS OF RAM
0003  F9        0075       LD     SP,HL           ;STORE IT IN STACK POINTER
0004  CD5300    0076       CALL   INIT            ;INITIALIZE GPIB INTERFACE
                0077 ;
                0078 ;  COMMAND DECODER
                0079 ;
0007  DB80      0080 CMND:IN      A,(CTRLI)       ;CHECK CONTROL INPUT FOR
                                  BUSY
0009  CB77      0081       BIT    BUSY,A          ;IF BUSY, DATA IN NICP
000B  28FA      0082       JR     Z,CMND          ;NOT BUSY,LOOK AGAIN
000D  DB40      0083       IN     A,(NICP)        ;READ VALID COMMAND
000F  FE00      0084       CP     0               ;IF COMMAND = 0, BOOT
0011  281C      0085       JR     Z,BOOT          ;IF BOOT, EXECUTE
0013  CB5F      0086       BIT    3,A             ;CHECK READ/WRITE BIT
0015  2807      0087       JR     Z,WRITE         ;IF 0, EXECUTE WRITE
                0088 ;
                0089 ;  COMMAND:  READ PORT, RETURN CONTENTS IN NICP
                0090 ;
0017  4F        0091 READ:LD      C,A             ;SET UP PORT ADDRESS
0018  ED78      0092       IN     A,(C)           ;GET PORT CONTENTS
001A  D340      0093       OUT    (NICP),A        ;SEND CONTENTS TO NICP
001C  18E9      0094       JR     CMND            ;LOOK FOR NEXT COMMAND
```

```
                0095 ;
                0096 ;  COMMAND:  WRITE NEXT BYTE TO PORT ADDRESSED IN A
                0097 ;
001E  4F        0098 WRITE:LD     C,A             ;SET UP PORT ADDRESS
001F  D340      0099      OUT     (NICP),A        ;SET DONE WITH COMMAND BYTE
0021  DB80      0100 WR1: IN      A,(CTRLI)       ;LOOK FOR NEXT BYTE
0023  CB77      0101      BIT     BUSY,A
0025  28FA      0102      JR      Z,WR1           ;IF NOT BUSY,LOOK AGAIN
0027  DB40      0103      IN      A,(NICP)        ;BUSY, GET NEXT BYTE
0029  ED79      0104      OUT     (C),A           ;SEND AS DATA TO PORT (C)
002B  D340      0105      OUT     (NICP),A        ;SEND TO NICP TO SET DONE
002D  18D8      0106      JR      CMND            ;OK FOR NEXT COMMAND
                0107 ;
                0108 ;  COMMAND:  BOOT 1K BYTES FROM NICP TO RAM AND EXECUTE
                0109 ;
002F  D340      0110 BOOT:OUT     (NICP),A        ;SET DONE WITH COMMAND BYTE
0031  210008    0111      LD      HL,RAM          ;SET UP RAM POINTER
0034  3E01      0112              LD      A,DNEWT         ;WAIT-ON-DONE
0036  0600      0113              LD      B,0             ;BYTE COUNT = 256
0038  0E40      0114              LD      C,NICP          ;NIC 1080 DATA PORT
003A  1604      0115              LD      D,04            ; # OF 256 BYTE
                        PAGES TO READ
003C  D380      0116              OUT     (CTRLO),A       ;ENABLE DONE WAIT
003E  DB80      0117 BOOT1:       IN      A,(CTRLI)       ;READ CONTROL PORT
0040  CB77      0118              BIT     BUSY,A          ;LOOK FOR BUSY BIT
0042  28FA      0119      JR      Z,BOOT1         ;IF NOT SET LOOK AGAIN
0044  EDA2      0120 BOOT2:       INI                     ;READ A DATA BYTE
0046  ED79      0121              OUT     (C),A           ;SET DONE FLAG
0048  20FA      0122      JR      NZ,BOOT2        ;NOT LAST BYTE,READ ANOTHER
004A  15        0123      DEC     D               ;DECREMENT PAGE COUNT
004B  20F7      0124      JR      NZ,BOOT2        ;NOT LAST PAGE, READ ANOTHER
004D  AF        0125              XOR     A               ;SET A=0
004E  D380      0126              OUT     (CTRLO),A       ;DISABLE DONE WAIT
0050  C30008    0127              JP      RAM             ;EXECUTE AT START OF
                        RAM
```

```
ASM TT CONO CTLSYM TTLMAC CTLMAIN TTLSUBS TTLSUBO CTLSTOR
Tektronix         Z80 ASM V3.3
**** Pass 2
Tektronix         Z80 ASM V3.3  NIC-488-CTL                    Page     1

00002                         LIST
00003                         NOLIST   MEG
00004                            GLOBAL  COM1,COM2,COM3,COM4,COM5,COM6,COM7
00005                            ;GPIB CONTROLLER SUBROUTINES ADAPTED FROM I
00006                            ;PERIPHERAL DESIGN HANDBOOK, AUG. 80,P 2-21
00007                   ;
00008                   ;
00009                   ;       8291 CONTROL VALUES
00010                   ;
00011      0020         PRT91   EQU     20H     ;8291 Base Port #
00012                   ;
00013                   ;       Req #0 data-in &data-out
10014      0020         DIN     EQU     PRT91+0 ;Data-in reg
00015      0020         DOUT    EQU     PRT91+0 ;Data-out reg
00016                   ;
00017                           ;Reg #1 Interrupt 1 Constants
00018      0021         INT1    EQU     PRT91+1 ;INT Reg 1
00019      0001         BOM     EQU     1       ;BO status bit no.
00020      0001         BIM     EQU     01      ;91 BI INTERP Mask
00021      0010         ENDMK   EQU     10H     ;91 END INTERP Mask
00022      0080         CPT     EQU     80H     ;91 command pass through in
00023                   ;
00024                   ;
00025                   ;       Req #2 Interrupt 2
00026      0022         INT2    EQU     PRT91+2
00027                   ;
00028                   ;
00029                   ;       Req #4 Address Mode Constants
00030      0024         ADRMD   EQU     PRT91+4 ;91 address mode register #
00031      0080         TON     EQU     80H     ;91 talk only mode & not li
00032      0040         LON     EQU     40H     ;91 listen only and not ton
00033      0001         MODE1   EQU     01      ;91 mode 1 addressing
00034                   ;
00035                   ;       Req #4 (read)
00036      0024         ADRST   EQU     PRT91+4
00037      0002         TA      EQU     2       ;Talk active
00038                   ;
00039                   ;
00040                   ;       Req #5 (write) Auxillary Mode Register
00041      0025         AUXMD   EQU     PRT91+5 ;91 auxillary mode register
00042      0024         CLKRT   EQU     24H     ;91 4 Mhz clock input
10043      0003         FNHSK   EQU     03      ;91 finish handshake comman
00044      000F         VSCMD   EQU     0FH     ;91 Valid command pass-thro
```

```
00045      0006        SEOI    EQU     06H     ;91 send EOI
00046      0080        AXRA    EQU     80H     ;91aux. reg A pattern
00047      0002        HOEND   EQU     2       ;91 hold off handshake on e
00048      0008        EOIS    EQU     8       ;91 output EOI on EOS sent
10049      0004        EDEOS   EQU     4       ;91 end on EOS received
00050      00A0        AXRB    EQU     0A0H    ;Aux. reg. B pattern
00051      0001        CPTEN   EQU     01H     ;Command pass-through enabl
00052                  ;
00053                  ;       Reg #5  (read)
00054      0025        CPTRG   EQU     PRT91+5 ;Command Pass-through ?Reg
```

```
00055                   ;
0005o                   ;
00057                   ;       Reg #6  Address 0/1 reg. constants
10058   0026    ADR01   EQU     PRT91+6
00059   0060    DTDL1   EQU     60H     ;Disable major talker $ lis
10060   00E0    DTDL2   EQU     0E0H    ;Disable minor talker & lis
10061                   ;
00062                   ;
00063                   ;       Reg #7  EOS     Character Register
10064   0027    EOSR    EQU     PRT91+7
00065                   ;
00066                   ;
00067                   ;       8292    CONTROL VALUES
00068                   ;
00069                   ;
00070   0010    PRT92   EQU     10H     ;8298 Base Port #
00071                   ;
00072   0010    INTMR   EQU     PRT92+0 ;92 INTRP Mask Reg
00073   00A0    INTM    EQU     0A0H    ;TCI
00074                   ;
00075   0010    ERRM    EQU     PRT92+0 ;92 error mask register
10076                   ;
00077   0010    ERFLAG  EQU     PRT92+0 ;error flag pseudo-register
10078   0002    TOUT2   EQU     02      ;92 time for standby
10079   0004    TOUT3   EQU     04              ;92 time out for TC
00080                   ;
00081   0010    TOREG   EQU     PRT92+0 ;92 time out pseudo-registe
00082   007F    TMOUT   EQU     7FH     ;Time out byte for TOREG
00083                   ;
00084   0011    CMD92   EQU     PRT92+1 ;92 Command Register
10085   0011    INTST   EQU     PRT92+1 ;92 Interrupt Status Reqist
10086   0002    IBFBT   EQU     2       ;Input Buffer full bit
10087   0020    SRQBT   EQU     20H     ;SRQ bit
10088   0040    ERRBT   EQU     40H     ;ERR bit
10089                   ;
00090   0010    CLRST   EQU     PRT92+0;92 Controller Status pseudo
00091   0008    SYCS    EQU     08H     ;Control Switch Status
10092   0040    CABT    EQU     40H     ;Controller active bit
10093                   ;
00094   0010    TOST    EQU     PRT92+0 ;92 time out pseudo-registe
00095                   ;
00096   0010    BUSST   EQU     PRT92+0 ;92 GPIB status pseudo-regi
00097   0008    SYCBT   EQU     08H     ;SYC status bit
10098                   ;
00099                   ;       8292    OPERATION COMMANDS
00100   00F2    RSET    EQU     0F2H     ;Reset
10101   00F3    RSTI    EQU     0F3H    ;reset interrupts
10102   00F6    GTSB    EQU     0F6H    ;Goto standby
```

```
00103    00F9    ABORT   EQU     0F9H    ;Interface clear
00104    00FC    TCASY   EQU     0FCH    ;take-control asynchronousl
00105    00FD    TCSY    EQU     0FDH    ;Take control syncronously
00106    00FA    TCNTR   EQU     0FAH    ;Take control (receive cont
00107            ;
```

```
00108                   ;       8292 UTILITY COMMANDS
00109                   ;
00110   00E1    WTOUT   EQU     0E1H    ;write to time out register
10111   00E4    RERF    EQU     0E4H    ;read error flag register
10112   00E6    RCST    EQU     0E6H    ;read Controller Status Reg
00113   00E7    RBST    EQU     0E7H    ;read GPIB status pseudo-re
00114   000B    IACK    EQU     0BH     ;Interrupt acknowledge
00115                   ;
00116                   ;       8292    INTERRUPT PORT
00117                   ;
00118   0008    PRTF    EQU     08H
00119   0001    TCIF    EQU     01H     ;Task complete interrupt
10120                   ;
00121                   ;       GPIB  MESSAGES (COMMANDS)
00122                   ;
00123   0001    MDA     EQU     1       ;My device address is 1
00124   0041    MTA     EQU     MDA+40H ;My talk address is 1 ("A")
00125   0021    MLA     EQU     MDA+20H ;My listen address is 1 ("!
00126   003F    UNL     EQU     3FH     ;Universal unlisten
00127   005F    UNT     EQU     5FH     ;Universal untalk
00128   0018    SPE     EQU     18H     ;Serial poll enable
00129   0019    SPD     EQU     19H     ;Serial poll disable
00130   0009    TCT     EQU     9       ;take control (pass control
00131                   ;
00132                   ;
00133                   ;       CTL PORTS
00134                   ;
00135   0080    CTRLI   EQU     80H     ;CTL 8-bit control input
10136   0001    ASR0    EQU     1       ;address switch 1
00137   0002    ASR1    EQU     2       ;address switch 2
00138   0004    ASR2    EQU     4       ;address switch 3
00139   0008    ASR3    EQU     8       ;address switch 4
00140   0010    ASR4    EQU     10H     ;address switch 5
00141   0005    DONE    EQU     5     ;DONE status bit
10142   0006    BUSY    EQU     6     ;BUSY status bit
10143   0080    INT8ST  EQU     80H     ;8291 interrupt status bit
10144                   ;
00145   0080    CTRLO   EQU     80H     ;CTL 4-bit control output
10146   0001    DNEWT   EQU     1       ;enable WAIT-ON-DONE
00147   0002    SRVC    EQU     2       ;set service request bit
10148   0004    DNECL   EQU     4       ;DONE clear pulse
00149   0010    DMAWT   EQU     10H     ;enable WAIT-ON-DONE
00150                   ;
00151                   ;       NICP PORT (IN/OUT TO NICOLET)
00152                   ;
00153   0040    NICP    EQU     40H
00154                   ;
00155                   ;       MISCELLANEOUS DEFINITIONS
```

```
      0B9D    >  TABLE  EQU     COMN    ;address of parameter table
                 ;
                 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
00161                   ;       JANUARY 27,1981
00162                   ;
00163                   ;
00164                   ;       MACRO DEFINITIONS
00165                           MACRO   WAITO   ;wait for byte out to bus
00166                   LAB'@'  IN      A,(INT1)
00167                           BIT     BOM,A
00168                           JR      Z,LAB'@'
00169                           ENDM
00170                   ;
00171                           MACRO   WRREG   ;REG,VALUE,[LABEL]
00172                   ;       REG=register to write to
00173                   ;       VALUE=value to write
00174                   ;       LABEL=optional jump to
00175                           LD      A,'2'
00176                           OUT     ('1'),A
00177                   K       ASET    3
00178                           IF      K='#'
00179                           JR      '3'
00180                           ENDIF
00181                           ENDM
00182                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00183                           MACRO   CTLNIC
00184                   ;       WRITES BYTES TO NIC FROM CTL
00185                   CTl'@'  IN      A,(CTRLI)
00186                           BIT     DONE,A
00187                           JR      NZ,CTl'@'
00188                           OUTI
00189                           JR      NZ,CTl'@'
00190                           ENDM
00191                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00192                           MACRO   NICCTL
00193                   ;       READS BYTES FROM NIC TO CTL
00194                   NIl'@'  IN      A,(CTRLI)
00195                           BIT     BUSY,A
00196                           JR      Z,NIl'@'
00197                           INI
00198                           OUT     (C),A
00199                           JR      NZ,NIl'@'
00200                           ENDM
00201                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00202                           MACRO   NICCTLI ;READS A SINGLE BYTE FROM N
00203                   NIl'@'  IN      A,(CTRLI)
00204                           BIT     BUSY,A
00205                           JR      Z,NIl'@'
00206                           IN      A,(NICP)
00207                           LD      D,A
00208                           OUT     (NICP),A
```

```
                ENDM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
00212                   ;       JANUARY 30,1981
00213                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00214                   ;
00215                   ;       MAIN CONTROL ROUTINE
00216                   ;
00217                   ;PURPOSE -- This is the CTL excutive routine which
00218                   ;       in that it performs commands issued by NIC
00219                   ;       wait state (waits for input from NIC) after
00220                   ;       If a command can not be completed because o
00221                   ;       considered fatal), the executive returns to
00222                   ;        an error subroutine which, among other thi
00223                   ;       indicate abnormal command termination.
00224                   ;       Each command is started by NIC by transferr
00225                   ;       block from NIC to CTL.The first byte in thi
00226                   ;       number.
00227       0800    >           ORG     800H    ;GPPROM jumps here.
00228 0800 CD6109 >           CALL INIT
00229 0803 3E14       WAIT    LD      A,14H   ;set parameters for NICI ta
00230 0805 32A80B >           LD      (NDAT),A
00231 0808 219D0B >           LD      HL,TABLE        ;Starting address o
00232 080B 22AA0B >           LD      (DATADD),HL
00233 080E 211708 >           LD      HL,WAIT1
00234 0811 22B10B >           LD      (RETADD),HL     ;normal return from
00235 0814 CDFE0A >           CALL    COM5    ;transfer table
00236 0817 210308 > WAIT1     LD      HL,WAIT
00237 081A 22B10B >           LD      (RETADD),HL     ;return address for
00238 081D 213C0B >           LD      HL,START        ;following code is
00239 0820 3A9D0B >           LD      A,(COMN)
00240 0823 4F                 LD      C,A
00241 0824 3AB30B >           LD      A,(LASTC)
00242 0827 B9                 CP      C
00243 0828 FC5708 >           CALL    M,ERROR
00244 082B 79                 LD      A,C
00245 082C A7                 AND     A
00246 082D FC5708 >           CALL    M,ERROR
00247 0830 CC5708 >           CALL    Z,ERROR
00248 0833 0600               LD      B,0
00249 0835 3D                 DEC     A
00250 0836 4F                 LD      C,A
00251 0837 87                 ADD     A,A
00252 0838 81                 ADD     A,C
00253 0839 4F                 LD      C,A
00254 083A 09                 ADD     HL,BC
00255 083B E9                 JP      (HL)
00256 083C CD2E0B > START     CALL    COM1
00257 083F CDD309 >           CALL    COM2
00258 0842 CD0000 >           CALL    COM3
00259 0845 CD650A >           CALL    COM4
```

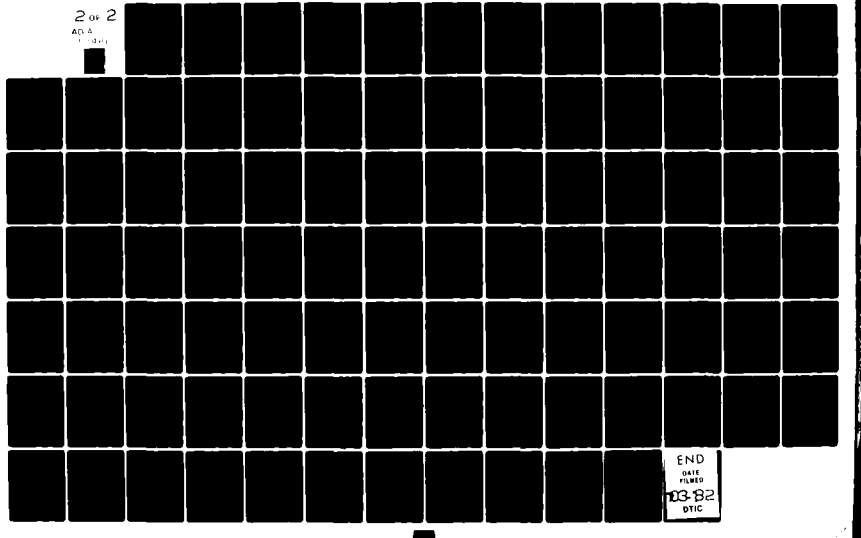
```
00260 0848 CDFE0A  >          CALL     COM5
00261 084B CD170B  >          CALL     COM6
00262 084E CD0000  >          CALL     COM7
00263 0851 CD0000             CALL     COM8
***** ERROR 074: Undefined symbol
```

```
0854 CD0000                CALL    COM9
***** ERROR 074: Undefined symbol
                           ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                           ;
                           ;       ERROR ROUTINE
                           ;
                           ;PURPOSE -- When a fatal error occurs in performin
                           ;       a call to ERROR is made. ERROR sets the SRV
                           ;       byte and the program counter value of the o
                           ;       to NIC.
                           ;
                           ERROR   WRREG   CTRLO,SRVC      ;Set the SRVC bit f
085B DB21                          IN      A,(INT1)
085D 57                            LD      D,A
085E CD5709     >                  CALL    CTLNIC1 ;send status to NIC
0861 E1                            POP     HL
0862 54                            LD      D,H
0863 CD5709     >                  CALL    CTLNIC1 ;send high order byte of th
0866 55                            LD      D,L
0867 CD5709     >                  CALL    CTLNIC1    ;send low order
086A AF                            XOR     A       ;clear SRVC
086B D380                          OUT     (CTRLO),A
086D E5                            PUSH    HL      ;return stack to normal
086E C34209     >                  JP      RETURN
                           ;
                           ;     JANUARY 18,1981
                           ;
                           ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                           ;
                           ;       LISLIST ROUTINE
                           ;
                           ;PURPOSE--send out a list of listeners or a single
                           ;ARGUMENTS-- req A = 1 => talker list
                           ;                    0 => listener list
                           ;
                           ;USES register A,B,DE
                           ;
0871 47                    LISLIST         LD      B,A
0872 A7                            AND     A
0873 200F                          JR      NZ,LI11
0875 3A9E0B     >                  LD      A,(NLIST)
0878 A7                            AND     A
0879 2003                          JR      NZ,LI3
087B 04                            INC     B
087C 1818                          JR      LI4
087E 47                    LI3     LD      B,A
087F 119F0B     >                  LD      DE,LISIP
0882 1803                          JR      LI1
```

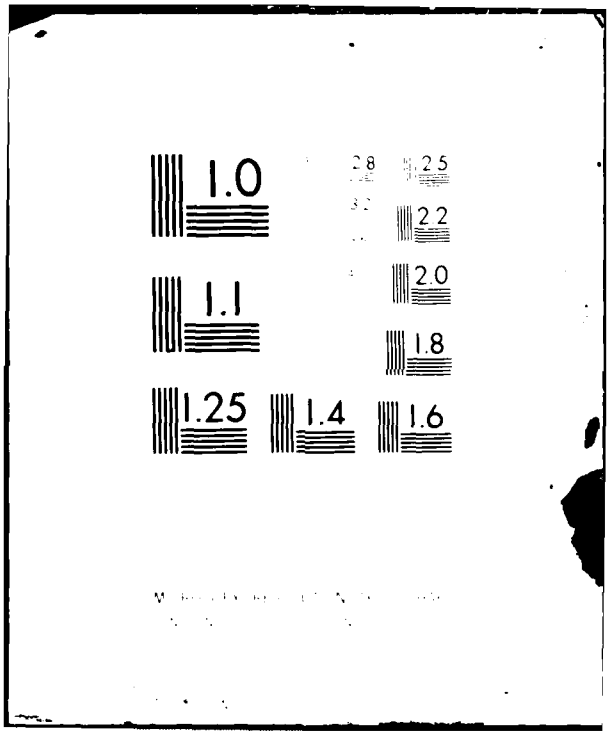
```
00311 0884 11A50B  >  LI11    LD      DE,TALKP
00312 0887 CB10       LI1     RL      B
00313 0889 1A         LI5     LD      A,(DE)
00314 088A A7                 AND     A
00315 088B 2808               JR      Z,LI2
```

AD-A111 481 PARKE MATHEMATICAL LABS INC CARLISLE MASS F/G 9/5
TOPICS IN OPTICAL MATERIALS AND DEVICE RESEARCH - II. VOLUME II--ETC(U)
JAN 82 T B BARRETT, H HASKEL, C E RYAN F19628-78-C-0089
UNCLASSIFIED RADC-TR-81-372-VOL-2 NL

2 OF 2
AD A



END
DATE FILMED
03-82
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5

```
088D D320                 OUT     (DOUT),A
                          WAITO
0895 13             LI2   INC     DE
0896 10F1           LI4   DJNZ    LI5
0898 C9                   RET
                    ;
                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                    ;
                    ;      BYTBLK ROUTINE
                    ;
                    ;PURPOSE-- sets up registers for block read or wri
                    ;      registers set are B,E,HL
                    ;      register C should be set by the caller for
                    ;RETURNS--the Z-flag is set if NDAT and NDATB =0
                    ;
0899 2AAA0B   >     BYTBLK LD     HL,(DATADD)
089C 3AA80B   >           LD      A,(NDAT)
089F 57                   LD      D,A
08A0 1E01                 LD      E,1
08A2 A7                   AND     A
08A3 2805                 JR      Z,BY1
08A5 47                   LD      B,A
08A6 3E01                 LD      A,1
08A8 1806                 JR      BY2
08AA 3AA90B   >     BY1   LD      A,(NDATB)
08AD 57                   LD      D,A
08AE 0600                 LD      B,0
08B0 5F             BY2   LD      E,A
08B1 7A                   LD      A,D
08B2 A2                   AND     D
08B3 C9                   RET
                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                    ;
                    ;      T3OUT ROUTINE
                    ;
                    ;PURPOSE--tests for TOUT3 errors on TCSY.If such a
                    ;      it does a TCASY with possible loss of data.
08B4 DB08           T3OUT IN      A,(PRTF)
08B6 E601                 AND     TCIF
08B8 20FA                 JR      NZ,T3OUT
08BA DB08           T32   IN      A,(PRTF)
08BC E601                 AND     TCIF
08BE 201B                 JR      NZ,T33
08C0 DB11                 IN      A,(INTST)
08C2 E640                 AND     ERRBT
08C4 28F4                 JR      Z,T32
08C6 16FF                 LD      D,0FFH
08C8 CDEB08   >           CALL    WRIND
```

```
00364 08CB 16E6              LD      D,RCST
00365 08CD CDDC08  >         CALL    RDIND
00366 08D0 E640              AND     CABT
00367 08D2 2007              JR      NZ,T33
00368 08D4 3EFC       T34    LD      A,TCASY
```

```
08D6 D311                 OUT     (CMD92),A
08D8 CD4A09  >            CALL    WAITX
08DB C9            T33    RET
                   ;
                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                   ;
                   ;
                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                   ;       SUBROUTINE RDIND
                   ; PURPOSE -- read 8292 indirect registers
                   ; ARGUMENTS -- reg D should contain the utility co
                   ; REVC,REERF,RINM,RCST,RBST,RTOUT or RERM
                   ;
08DC DB11          RDIND  IN      A,(INTST)
08DE E602                 AND     IBFBT
08E0 20FA                 JR      NZ,RDIND
08E2 7A                   LD      A,D
08E3 D311                 OUT     (CMD92),A
08E5 CD4A09  >            CALL    WAITX
08E8 DB10                 IN      A,(PRT92)
08EA C9                   RET
                   ;
                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                   ;
                   ;       SUBROUTINE WRIND
                   ; PURPOSE -- write 8292 indirect registers or to s
                   ; ARGUMENTS -- reg D should contain WTOUT,WEVC or
                   ;              reg E should contain a value to be
                   ;              in the indirect reg (except for IAC
08EB DB11          WRIND  IN      A,(INTST)
08ED E602                 AND     IBFBT
08EF 20FA                 JR      NZ,WRIND
08F1 7A                   LD      A.D
08F2 D311                 OUT     (CMD92),A
08F4 DB11          WR1    IN      A,(INTST)
08F6 E602                 AND     IBFBT
08F8 20FA                 JR      NZ,WR1
08FA CB5A                 BIT     3,D
08FC 2009                 JR      NZ,WR2  ; if IACK this is all
08FE 7B                   LD      A,E
08FF D310                 OUT     (PRT92),A
0901 DB11          WR3    IN      A,(INTST)
0903 E602                 AND     IBFBT
0905 20FA                 JR      NZ,WR3
0907 C9            WR2    RET
                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                   ;
                   ;       SUBROUTINE T2IN
```

```
00417                   ; PURPOSE -- check for data in from the bus and to
00418                   ;            certain actions under various "time o
00419                   ;            conditions
00420                   ; ARGUMENTS -- reg D should contain either hex 01,
00421                   ; indicating DJNZ time out, clear SRQ or get statu
```

```
00422                   ; after a time out respectively.
00423                   ;           -- T2 should be set to 1 if this is th
00424                   ; data in, indicating that the time out condition
00425                   ; T2IN will set T2 to 0. While T2 is 0 only DZNZ t
00426                   ; is used.
00427                   ;           -- reg A returns the INT1 status bits.
00428                   ;
00429                   ; NOTE -- Before calling T2IN, do an EXX then set
00430 0908 3ABA0B  >  T2IN    LD      A,(T2); is this the first line or n
00431 090B 0600            LD      B,0     ;B is used in the DJNZ loop
00432 090D 58              LD      E,B     ; E holds the status bits.
00433 090E A7              AND     A
00434 090F 2827            JR      Z,T23   ;not first time
00435 0911 AF              XOR     A       ;set to not first time
00436 0912 32BA0B  >       LD      (T2),A
00437 0915 DB21        T21 IN      A,(INT1)
00438 0917 B3              OR      E       ;collect status bits
00439 0918 5F              LD      E,A
00440 0919 A2              AND     D
00441 091A 2010            JR      NZ,T26  ;if byte is in,we are done.
00442 091C DB11        T25 IN      A,(INTST)       ;check for TOUT2 er
00443 091E E660            AND     ERRBT+SRQBT
00444 0920 28F3            JR      Z,T21   ;if no error then wait more
00445 0922 CB6A            BIT     5,D     ;if bit 5 is 0 then no SRQ
00446                   ; is wrong
00447 0924 CC5708  >       CALL    Z,ERROR
00448 0927 CB42            BIT     0,D     ;if bit 0 is 0,we are not e
00449                   ;so wait for SRQ
00450 0929 A2              AND     D
00451 092A 28F0            JR      Z,T25
00452 092C 16FF        T26 LD      D,0FFH  ;clear all SPI flags
00453 092E CDEB08  >       CALL WRIND
00454 0931 180C            JR      T24
00455 0933 A2          T22 AND     D       ;wait for SRQ or BI to be s
00456 0934 28DF            JR      Z,T21
00457 0936 1807            JR      T24
00458 0938 DB21        T23 IN      A,(INT1)        ; DJNZ or BI loop
00459 093A B3              OR      E
00460 093B 5F              LD      E,A
00461 093C A2              AND     D
00462 093D 28F9            JR      Z,T23
00463 093F 7B          T24 LD      A,E     ;status bits are returned i
00464 0940 D9              EXX     ;put registers in "normal" mode.
00465 0941 C9              RET
00466                   ;
00467                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00468                   ;
00469                   ;       PSEUDO-SUBROUTINE RETURN
```

```
00470                    ;      Returns subroutines to RETADD and writes a
00471 0942 2AB10B  >  RETURN LD       HL,(RETADD)
00472 0945 C1              POP      BC
00473 0946 CD5709  >       CALL     CTLNIC1
00474 0949 E9              JP       (HL)
```

```
                   ;
                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                   ;       SUBROUTINE WAITX
                   ; PURPOSE -- wait for TCI
094A DB08          WAITX  IN      A,(PRTF)
094C E601                 AND     TCIF
094E 20FA                 JR      NZ,WAITX
0950 DB08          WX1    IN      A,(PRTF)
0952 E601                 AND     TCIF
0954 28FA                 JR      Z,WX1
0956 C9                   RET
                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                   ;       SUBROUTINE      CTLNC1
                   ;       writes a single byte from CTL to NIC
                   ;       the byte should be in the D register
0957 DB80          CTLNIC1        IN      A,(CTRLI)
0959 CB6F                 BIT     DONE,A
095B 20FA                 JR      NZ,CTLNIC1
095D 7A                   LD      A,D
095E D340                 OUT     (NICP),A
0960 C9                   RET
                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
00498                   ;    JANUARY 18,1981
00499                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00500                   ;initializaton routine -
00501                   ;       sets up CTL to be CTL.  If the SYC switch (
00502                   ;is off then CTL waits to be put in charge by anot
00503                   ;HP9835 calculator).  In this state, with the curr
00504                   ;nothing.  After control is transfered to CTL, it
00505                   ;switch were on.  Note that this initialization ro
00506                   ;action by a "pre-initializer" such as GPPROM (Gen
00507                   ;by Don Terpstra).  If this initializer replaces G
00508                   ;functions done in GPPROM should be done by INIT.
00509                   ;
00510                   INIT    WRREG   CMD92,RSET
00511 0965 0600                   LD      B,0     ;wait for abort to go out,e
00512 0967 10FE         INIT0     DJNZ    INIT0
00513                             WRREG   INTMR,INTM      ;set TCI interrupt
00514 096D 16E7                   LD      D,RBST
00515 096F CDDC08   >             CALL    RDIND
00516 0972 E608                   AND     SYC8T
00517 0974 2039                   JR      NZ,INIT2        ;go to immediate co
00518                             WRREG   ADRMD,MODE1     ;not ton,mode,addre
00519                             WRREG   ADR01,MDA
00520                             WRREG   AUXMD,AXRB+CPTEN        ;enable com
00521 0982 DB21         INIT1     IN      A,(INT1)        ;wait here for TCT
00522 0984 E680                   AND     CPT
00523 0986 28FA                   JR      Z,INIT1         ;if not TCT or my a
00524 0988 DB25                   IN      A,(CPTRG)
00525 098A FE09                   CP      TCT
00526 098C 2806                   JR      Z,INIT11
00527                   INIT12  WRREG   AUXMD,VSCMD,INIT1       ;if not TCT
00528 0994 DB24         INIT11    IN      A,(ADRST)
00529 0996 E602                   AND     TA
00530 0998 28F4                   JR      Z,INIT12
00531                             WRREG   ADR01,DTDL1     ;disable talker/lis
00532                             WRREG   ADRMD,TON       ;talk only
00533                             WRREG   CMD92,TCNTR     ;take(receive)contr
00534                             WRREG   AUXMD,VSCMD     ;continue command p
00535 09AA CD4A09   >             CALL    WAITX
00536 09AD 180C                   JR      INIT3   ;rest of initialization as
00537                   INIT2   WRREG   ADR01,DTDL1
00538                             WRREG   ADR01,DTDL2
00539                             WRREG   ADRMD,TON
00540                   INIT3   WRREG   AUXMD,CLRT
00541                             WRREG   INT1,0
00542                             WRREG   INT2,0
00543                             WRREG   ERRM,TOUT2+TOUT3
00544 09CB 16E1                   LD      D,WTOUT
00545 09CD 1E7F                   LD      E,TMOUT
```

```
00546 09CF CDEB08  >          CALL    WRIND
00547 09D2 C9                 RET
00548                    ;
00549                    ;    RECV ROUTINE (alias COM2)
00550                    ;
```

```
00551                   ;PURPOSE--Transfers data from GPIB to CTL (or to o
00552                   ;PARAMETERS -
00553                   ;       (1) no. of listeners (other than CTL)
00554                   ;       (2) primary address-first listener
10555                   ;       (3   secondary address-first listener
10556                   ;       (4) --second listener
10557                   ;       (5) --
00558                   ;       (6) -- third listener
10559                   ;       (7)
00560                   ;       (8) primary address-talker
10561                   ;       (9) secondary address -talke:
00562                   ;       (10) EOS character. (see below)
00563                   ;       (11) no. of data bytes to receive
00564                   ;       (12) no. of 256 blocks to receive. (see bel
00565                   ;       (14) starting address for data storage.
00566                   ;
00567                   ;RETURNS - If (11) and (12) are both 0,RECV return
00568                   ;       data in (12) and (11). If do not get EOI at
00569                   ;        or if the amount of data received does not
00570                   ;       an error condition exists.
00571                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
00572 09D3 3E01        COM2    LD      A,1
00573 09D5 CD7108   >           CALL    LISLIST
00574 09D8 AF               XOR     A
00575 09D9 32BA0B   >           LD      (T2),A  ;get ready for T2OUT
00576                           WRREG   DOUT,UNL
00577                           WAITO
00578 09E6 AF               XOR     A
00579 09E7 CD7108   >           CALL    LISLIST
00580 09EA 0682             LD      B,AXRA+HOEND
00581 09EC 3AA70B   >           LD      A,(EOSC)
00582 09EF A7               AND     A
00583 09F0 2804             JR      Z,COM25
00584 09F2 D327             OUT     (EOSR),A
00585 09F4 0686             LD      B,AXRA+HOEND+EDEOS
00586                   COM25   WRREG   AUXMD,B
00587                           WRREG   ADRMD,LON
00588                           WRREG   AUXMD,0
00589 0A01 3E01             LD      A,1
00590 0A03 32BA0B   >           LD      (T2),A
00591                           WRREG   CMD92,GTSB
00592 0A0A CD4A09   >           CALL    WAITX
00593 0A0D 0E20             LD      C,DIN
00594 0A0F CD9908   >           CALL    BYTBLK
00595 0A12 2006             JR      NZ,COM27
00596 0A14 3AB40B   >           LD      A,(MAXBLK)          ;If (11) and (12)
00597 0A17 5F               LD      E,A
00598 0A18 0600             LD      B,0
```

```
00599 0A1A DD210000    COM27  LD      IX,0        ;set data counter to 0
00600 0A1E D9          COM28  EXX
00601 0A1F 1601               LD      D,BIM
00602 0A21 CD0809   >         CALL    T2IN
00603 0A24 E610               AND     ENDMK
```

```
0A26 200B              JR     NZ,COM29
0A28 DD23              INC    IX
0A2A EDA2              INI
0A2C 20F0              JR     NZ,COM28
0A2E 1D                DEC    E
0A2F 20ED              JR     NZ,COM28
0A31 1818              JR     COM212
0A33 EDA2       COM29  INI
0A35 DD23              INC    IX
0A37 DD22B70B>         LD     (COUNT),IX
0A3B 3AA80B  >         LD     A,(NDAT)         ;if (11) and (12) a
0A3E 47                LD     B,A
0A3F 3AA90B  >         LD     A,(NDATB)
0A42 80                ADD    A,B
0A43 2006              JR     NZ,COM212
0A45 2AB70B  >         LD     HL,(COUNT)
0A48 22A90B  >         LD     (NDATB),HL
                COM212 WRREG  CMD92,TCSY
0A4F CDB408  >         CALL   T3OUT
                       WRREG  AUXMD,AXRA
                       WRREG  ADRMD,TON
                       WRREG  AUXMD,FNHSK
                       WRREG  AUXMD,0
0A62 C34209  >         JP     RETURN
                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                ;
                ;      POLL ROUTINE  (alias COM4)
                ;
                ;PURPOSE -- wait for SRQ and do a serial poll of t
                ; device requesting service. This routine has been
                ; to match some of the ideosyncrasies of the Hamam
                ;PARAMETERS--
                ;      (8) primary address-device to be polled
                ;      (9) secondary address
                ;
                ;RETURNS-- the status byte is written to NICP if o
                ;      Otherwise an error return is done.
0A65 AF         COM4   XOR    A       ;initialize the subroutine
0A66 4F                LD     C,A
0A67 32BA0B  >         LD     (T2),A
0A6A 3E20              LD     A,SRQBT ;save the initial value to
                ;             for use by T2IN
0A6C 32B90B  >         LD     (TEMP),A
                COM41  WRREG  DOUT,UNL
                       WAITO
                       WRREG  DOUT,MLA
                       WAITO
0A83 79                LD     A,C
```

```
00652 0A84 A7                   AND     A       ;check control reg. to see
00653 0A85 2819                 JR      Z,COM42
00654 0A87 180A                 JR      COM43
00655                   COM44   WRREG   DOUT,SPE
00656                           WAITO
```

```
00657 0A93 3E01       COM43  LD      A,1
00658 0A95 CD7108  >         CALL    LISLIST ;sends out talkeer address
00659                        WRREG   ADRMD,LON
00660                        WRREG   AUXMD,0
00661 0AA0 3E01       COM42  LD      A,1     ;get ready for T2IN
00662 0AA2 32BA0B  >         LD      (T2),A
00663                        WRREG   CMD92,GTSB
00664 0AA9 CD4A09  >         CALL    WAITX
00665 0AAC D9                EXX     ;exchange registers for T2IN
00666 0AAD 3AB90B  >         LD      A,(TEMP)
00667 0AB0 57                LD      D,A
00668 0AB1 CD0809  >         CALL    T2IN
00669 0AB4 DB20              IN      A,(DIN) ;get RQS and STB for later
00670 0AB6 32B90B  >         LD      (TEMP),A
00671                        WRREG   CMD92,TCSY
00672 0ABD CDB408  >         CALL    T3OUT
00673 0AC0 79                LD      A,C
00674 0AC1 3C                INC     A
00675 0AC2 4F                LD      C,A
00676 0AC3 CB49              BIT     1,C
00677 0AC5 2007              JR      NZ,COM45
00678 0AC7 3E21              LD      A,SRQBT+BIM     ;second pass throug
00679 0AC9 32B90B  >         LD      (TEMP),A
00680 0ACC 18BB              JR      COM44
00681 0ACE CB41       COM45  BIT     0,C
00682 0AD0 2021              JR      NZ,COM46
00683 0AD2 3AB90B  >         LD      A,(TEMP)
00684 0AD5 57                LD      D,A
00685 0AD6 CD5709  >         CALL    CTLNIC1
00686                        WRREG   ADRMD,TON       ;do serial poll dis
00687                        WRREG   AUXMD,0
00688                        WRREG   DOUT,SPD
00689                        WAITO
00690 0AEB 3E01              LD      A,BIM
00691 0AED 32B90B  >         LD      (TEMP),A
00692 0AF0 C36F0A  >         JP      COM41   ;if necessary
00693                 COM46  WRREG   ADRMD,TON
00694                        WRREG   AUXMD,0
00695 0AFB C34209  >         JP      RETURN
00696                 ;
00697                 ;      NICI ROUTINE  (alias COM5)
00698                 ;
00699                 ;PURPOSE --read data from NIC to CTL
00700                 ;PARAMETERS--
00701                 ;      (11)  no. of bytes or
00702                 ;      (12) no. of 256 byte blocks
00703                 ;      (14) starting address for data storage
00704                 ;
```

```
00705 0AFE 0E40        COM5    LD      C,NICP
00706 0B00 CD9908  >           CALL    BYTBLK
00707 0B03 280F                JR      Z,COM53
00708                  COM52   NICCTL
00709 0B11 1D                  DEC     E
```

```
0B12 20F1              JR      NZ,COM52
0B14 C34209  >  COM53  JP      RETURN
                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                ;
                ;      NICO ROUTINE  (alias COM6)
                ;
                ;PURPOSE -- write data from CTL to NIC
                ;
                ;PARAMETERS --
                ;      (11) no. of data bytes or
                ;      (12) no. of 256 byte blocks
                ;      (14)  starting address of the data
                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
0B17 0E40       COM6   LD      C,NICP
0B19 CD9908  >         CALL    BYTBLK
0B1C 280D              JR      Z,COM63
                COM62  CTLNIC
0B28 1D                DEC     E
0B29 20F3              JR      NZ,COM62
0B2B C34209  >  COM63  JP      RETURN
                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                ;     JANUARY 18,1981
                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                ;
                ;      SEND ROUTINE (ALIAS COM1)
                ;
                ;      sends data from CTL to the GPIB
                ;
                ;INPUT (1) no. of listeners
                ;      (2) primary address-first listener
                ;      (3) secondary address - first listener
                ;      (4) --second listener
                ;      (5) --
                ;      (6) --third listener
                ;      (7)--
                ;      (10)  EOS character (if EOS=0 then EOI is s
                ;      according to the data count given by (10) a
                ;      non-zero ,the no. of characters sent is det
                ;      location of the EOS character in the data s
                ;      the amount of data as determined by (10) or
                ;      or equal to this location. Otherwise EOI is
                ;      (11)  no. of bytes to send or
                ;      (12)  no. of 256 byte blocks to send. (If b
                ;       no data is sent.
                ;      (14)  sarting address of th data to be sent
                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
0B2E AF         COM1   XOR     A
0B2F 32BA0B  >         LD      (T2),A
```

```
00758                          WRREG   DOUT,MTA
00759                          WAITO
00760                          WRREG   DOUT,UNL
00761                          WAITO
00762 0B46 AF                  XOR     A
```

```
0B47 CD7108  >         CALL    LISLIST
0B4A 3E01              LD      A,1
0B4C 32BA0B  >         LD      (T2),A
                       WRREG   CMD92,GTSB
0B53 CD4A09  >         CALL    WAITX
0B56 0E20              LD      C,DOUT
0B58 CD9908  >         CALL    BYTBLK
0B5B 2836              JR      Z,COM17
0B5D 78          COM15 LD      A,B
0B5E A7                AND     A
0B5F 280B              JR      Z,COM16
0B61 83                ADD     A,E      ;if B+E is 2, then this is
                                        ;providing bB .ne. 0
0B62 3D                DEC     A
0B63 3D                DEC     A
0B64 2006              JR      NZ,COM16
                       WRREG   AUXMD,SEOI
0B6A 1812              JR      COM18
0B6C 3AA70B  >   COM16 LD      A,(EOSC)         ;if EOSC is 0,don t
0B6F A7                AND     A
0B70 280C              JR      Z,COM18
0B72 56                LD      D,(HL)
0B73 BA                CP      D
0B74 2008              JR      NZ,COM18
                       WRREG   AUXMD,SEOI
0B7A 0601              LD      B,1      ;send EOI and quit.
0B7C 1E01              LD      E,1
0B7E EDA3        COM18 OUTI
0B80 2808              JR      Z,COM19
                       WAITO
0B88 18D3              JR      COM15
                 COM19 WAITO
0B90 1D                DEC     E
0B91 20CA              JR      NZ,COM15
                 COM17 WRREG   CMD92,TCSY
0B97 CDB408  >         CALL    T3OUT
0B9A C34209  >   COM110 JP     RETURN
                 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
                      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                      ;
                      ;       PARAMETER TABLE
                      ;
0B9D 00               COMN   BYTE    0        ;command number
0B9E 00               NLIST  BYTE    0        ;no. of listeners (or can b
0B9F 00               LIS1P  BYTE    0        ;primary address,1st listen
0BA0 00               LIS1S  BYTE    0        ;secondary address,first li
0BA1 00               LIS2P  BYTE    0        ;--second listener
0BA2 00               LIS2S  BYTE    0
0BA3 00               LIS3P  BYTE    0        ;--third listener
0BA4 00               LIS3S  BYTE    0
0BA5 00               TALKP  BYTE    0        ;talker primary address
0BA6 00               TALKS  BYTE    0        ;talker secondary address
0BA7 00               EOSC   BYTE    0        ;EOS character (0 means non
0BA8 00               NDAT   BYTE    0        ;no. of data bytes to be tr
0BA9 00               NDATB  BYTE    0        ;no. of 256 byte blocks to
0BAA 0000             DATADD WORD    0        ;starting address of the da
0BAC 00               DUM1L  BYTE    0
0BAD 00               DUM1H  BYTE    0
0BAE 00               MESS   BYTE    0        ;message print indicator
0BAF 00               STAT1  BYTE    0
0BB0 00               STAT2  BYTE    0
                      ;
                      ;       OTHER DATA
                      ;
0BB1 0000             RETADD WORD    0
0BB3 08               LASTC  BYTE    8        ;set to the last valid comm
0BB4 04               MAXBLK BYTE    4        ;set to the max. no. of 256
0BB5 0000             STACKP WORD    0        ;temporary storage for stac
0BB7 0000             COUNT  WORD    0        ;data counter location
                      ;
                      ;       TEMPORARY STORAGE
                      ;
0BB9 00               TEMP   BYTE    0
0BBA 00               T2     BYTE    0
```

Strings and Macros

```
CTLNIC - 007A M          NICCTL - 008D M          NICCTL1  0097 M
WAITO -- 0052 M          WRREG -- 007F M
```

Scalars

```
ABORT -- 00F9            ADR01 -- 0026            ADRMD -- 0024
ADRST -- 0024            ASR0 --- 0001            ASR1 --- 0002
ASR2 --- 0004            ASR3 --- 0008            ASR4 --- 0010
AUXMD -- 0025            AXRA --- 0080            AXRB --- 00A0
BIM ---- 0001            BOM ---- 0001            BUSST -- 0010
BUSY --- 0006            CABT --- 0040            CLKRT -- 0024
CLRST -- 0010            CMD92 -- 0011            COM8 --- ****
COM9 --- ****            CPT ---- 0080            CPTEN -- 0001
CPTRG -- 0025            CTRLI -- 0080            CTRLO -- 0080
DIN ---- 0020            DMAWT -- 0010            DNECL -- 0004
DNEWT -- 0001            DONE --- 0005            DOUT --- 0020
DTDL1 -- 0060            DTDL2 -- 00E0            EDEOS -- 0004
ENDMK -- 0010            EOIS --- 0008            EOSR --- 0027
ERFLAG - 0010            ERRBT -- 0040            ERRM --- 0010
FNHSK -- 0003            GTSB --- 00F6            HOEND -- 0002
IACK --- 000B            IBFBT -- 0002            INT1 --- 0021
INT2 --- 0022            INT8ST - 0080            INTM --- 00A0
INTMR -- 0010            INTST -- 0011            K ------ 0003 V
LON ---- 0040            MDA ---- 0001            MLA ---- 0021
MODE1 -- 0001            MTA ---- 0041            NICP --- 0040
PRT91 -- 0020            PRT92 -- 0010            PRTF --- 0008
RBST --- 00E7            RCST --- 00E6            RERF --- 00E4
RSET --- 00F2            RSTI --- 00F3            SEOI --- 0006
SPD ---- 0019            SPE ---- 0018            SRQBT -- 0020
SRVC --- 0002            SYCBT -- 0008            SYCS --- 0008
TA ----- 0002            TCASY -- 00FC            TCIF --- 0001
TCNTR -- 00FA            TCSY --- 00FD            TCT ---- 0009
TMOUT -- 007F            TON ---- 0080            TOREG -- 0010
TOST --- 0010            TOUT2 -- 0002            TOUT3 -- 0004
UNL ---- 003F            UNT ---- 005F            VSCMD -- 000F
WTOUT -- 00E1
```

8TT (default) Section (0BBB)

```
BY1 ---- 08AA            BY2 ---- 08B0            BYTBLK - 0899
COM1 --- 0B2E G          COM110 - 0B9A            COM15 -- 0B5D
COM16 -- 0B6C            COM17 -- 0B93            COM18 -- 0B7E
COM19 -- 0B8A            COM2 --- 09D3 G          COM212 - 0A4B
COM25 -- 09F6            COM27 -- 0A1A            COM28 -- 0A1E
COM29 -- 0A33            COM4 --- 0A65 G          COM41 -- 0A6F
```

```
COM42 -- 0AA0      COM43 -- 0A93      COM44 -- 0A89
COM45 -- 0ACE      COM46 -- 0AF3      COM5 --- 0AFE G
COM52 -- 0B05      COM53 -- 0B14      COM6 --- 0B17 G
COM62 -- 0B1E      COM63 -- 0B2B      COMN --- 0B9D
COUNT -- 0BB7      CT10300  0B1E      CTLNIC1  0957
```

```
DATADD - 0BAA      DUM1H -- 0BAD      DUM1L -- 0BAC
EOSC --- 0BA7      ERROR -- 0857      INIT --- 0961
INIT0 -- 0967      INIT1 -- 0982      INIT11 - 0994
INIT12 - 098E      INIT2 -- 09AF      INIT3 -- 09BB
LAB0200  0A73      LAB2000  088F      LAB2200  0A7D
LAB2300  0B36      LAB4200  0A8D      LAB4300  0B40
LAB5100  09E0      LAB8300  0B82      LAB9300  0B8A
LABC200  0AE5      LASTC -- 0BB3      LI1 ---- 0887
LI11 --- 0884      LI2 ---- 0895      LI3 ---- 087E
LI4 ---- 0896      LI5 ---- 0889      LIS1P -- 0B9F
LIS1S -- 0BA0      LIS2P -- 0BA1      LIS2S -- 0BA2
LIS3P -- 0BA3      LIS3S -- 0BA4      LISLIST  0871
MAXBLK - 0BB4      MESS --- 0BAE      NDAT --- 0BA8
NDATB -- 0BA9      NI1F200  0B05      NLIST -- 0B9E
RDIND -- 08DC      RETADD - 0BB1      RETURN - 0942
STACKP - 0BB5      START -- 083C      STAT1 -- 0BAF
STAT2 -- 0BB0      T2 ----- 0BBA      T21 ---- 0915
T22 ---- 0933      T23 ---- 0938      T24 ---- 093F
T25 ---- 091C      T26 ---- 092C      T2IN --- 0908
T32 ---- 08BA      T33 ---- 08DB      T34 ---- 08D4
T3OUT -- 08B4      TABLE -- 0B9D      TALKP -- 0BA5
TALKS -- 0BA6      TEMP --- 0BB9      WAIT --- 0803
WAIT1 -- 0817      WAITX -- 094A      WR1 ---- 08F4
WR2 ---- 0907      WR3 ---- 0901      WRIND -- 08EB
WX1 ---- 0950

COM3 Unbound Global

COM7 Unbound Global


  838 Source Lines    1213 Assembled Lines    9942 Bytes available
  838 Source Lines    1213 Assembled Lines    9942 Bytes available

    2 ERRORS            2 UNDEFINED SYMBOLS
    2 ERRORS            2 UNDEFINED SYMBOLS
*ASM* EOJ
TTL;C COMPLETED

>
```

APPENDIX C - NIC Software

```
// CAMERA MAIN (JANUARY 30,1981)
// MAIN DRIVER PROGRAM FOR CAMERA CONTROL. THIS PROGRAM PERFORMS
/ VARIOUS BASIC OPERATIONS HAVING TO DO WITH CONTROLLING THE
/ HAMAMATSU C1000 CAMERA VIA THE NIC-488/CTL INTERFACE.
/ IN ADDITION VARIOUS CONTROLLER FUNCTIONS ARE PERFORMED. EACH FUNCTION
/ IS CALLED A MODE (OF OPERATION). AT PRESENT 7 MODES HAVE BEEN
/ IMPLEMENTED AS FOLLOWS:
/
/ MODE1 - RESET THE CAMERA,CONTROLLER AND LOAD THE CONTROLLER MEMORY
/ THIS OPERATION SHOULD USUALLY BE THE FIRST OPERATION ON SYSTEM
/ START AND MAY BE REPEATED ANY TIME THE USER WISHES TO
/ RESTART. THE QUERY IS -
/  FILE NAME?- (REPLY WITH THE NAME OF THE NIC FILE CONTAINING
/ THE MAIN CONTROLLER PROGRAM. (NOTE THAT AT PRESENT THIS IS STORED
/ IN RAM WHICH MUST BE RELOADED EACH TIME THE CTL IS TURNED OFF.)
/
/ MODE2 - LOAD AND EXECUTE SPECIAL "COMMANDS". AT PRESENT THE
/ CTL RECOGNIZES TWO OPTIONAL COMMANDS (NOS. 7 & 8). THESE
/ COMMANDS ARE NOT PART OF THE MAIN SEQUENCE OF CTL PROGRAMS AND
/ MUST BE LOADED BEFORE CALLING. AT PRESENT, IT IS ASSUMED THAT THE
/ CORRESPONDING NIC COMMAND PROGRAM IS IN NIC CORE. THE MODE QUERY IS-
/  COMMAND #?- (REPLY 7 OR 8)
/  FILE NAME?- (REPLY WITH THE NIC FILE TO BE LOADED)
/       [OPTIONAL SPECIAL DATA, DEPENDING ON THE COMMAND]
/
/ MODE3 - TRANSFER AND TRANSFORM A TEKTRONIX 8002 TEK-HEX
/ FILE FROM COMM-STOR TO A NIC LOAD FILE. THIS MODE PROVIDES FOR
/ EASY TRANSFER OF ASSEMBLED Z-80 CODE TO NIC AND THENCE TO CTL.
/ THE MODE QUERY IS -
/  FILE NAME?- (RESPOND WITH THE COMM-STOR FILE NAME WHICH
/    ALSO BECOMES THE NIC FILE NAME)
/ NOTE- IF THE FILE DOES NOT EXIST IN COMMSTOR, THE USER WILL
/ BE INFORMED OF THIS. HE MUST THEN TYPE ^Z TO GET PROGRAM CONTROL.
/
/ MODE4 - CAMERA SET-UP. USING THIS MODE, ANY OF 5 LEGITIMATE
/ CAMERA SET-UP COMMANDS GIVEN IN CAMERA TABLE ARE SENT TO THE CAMERA.
/ THE MODE QUERY IS -
/  MNEMONIC?- (REPLY WITH ONE OF THE 3 LETTER MNEMONIC
/  CODES: OUT INF XCO INT MAR
/ THE CURRENT VALUE IS THEN PRINTED AFTER WHICH THE USER MAY TYPE
/ IN A NEW VALUE OR CR TO RETAIN THE GIVEN ONE.
/ NOTE THAT AT PRESENT THE USER SHOULD NOT CHANGE OUT WHICH GIVES
/ THE FORMAT OF DATA SENT TO THE CAMERA. IT SHOULD STAY AT
/ ITS DEFAULT VALUE OF 1 MEANING ASCII DATA.
/
/ MODE5 - OBTAIN A "FRAME" OF VIDEO DATA (THE SIZE OF A FRAME DEPENDS
/ ON THE CAMERA TABLE VALUES FOR XCO AND INT AND ON THE COMMAND).
/ THE MODE QUERY IS -
```

```
/  FILE NAME?- (REPLY WITH THE NAME OF THE NIC FILE TO RECEIVE
/  THE DATA OR WITH "PRINT" WHICH WILL CAUSE THE DATA TO BE
/  PRINTED BUT NOT STORED. IF "PRINT" IS GIVEN, THE ACTUAL FORM
/  OF THE PRINTED VIDEO DEPENDS ON THE INPUT FORMAT (INF). IF 1 OR 3,
/  THE DISPLAY IS 3 CHARACTER (000-256) FOLLOWED BY "BLANK", 16
/  PIXELS/LINE. IF 2, THE DISPLAY IS 2 CHARACTER HEX (00-FF) WITH
/  NO BLANKS AND 32 PIXELS/LINE.)
/  FRAME?- (REPLY WITH VI1,VII,VID FOR 1-LINE VIDEO,A FRAME
/  STARTING AT X-COORD GOING TO HIGHER X-VALUES, A FRAME
/  STARTING AT X-COORD GOING TO LOWER X-VALUES RESPECTIVELY.
/
/ MODE6 - TRANSFER CTL MEMORY TO NIC AT 100000 IN PACKED FORM.
/ THE MODE QUERY IS -
/  HEXN- (REPLY WITH 4 HEX DIGIT ADDRESS AND A SINGLE DIGIT
/         SPECIFYING THE NO. OF 256 BYTE BLOCKS WANTED.
/       E.G. 0C004 MEANS ADDRESS C00 AND 4 BLOCKS)
/ THE DATA OBTAINED MAY BE DISPLAYED BY CALLING MODE7
/
/MODE7 - DISPLAY MEMORY IN HEX FORMAT. THE REGION DDISPLAYED STARTS
/ AT 100000 AND IS NW3 WORDS LONG WHERE NW3 IS OBTAINED BY RUNNING MODE3
/ OR MODE6. THE USER CAN ALSO CHANGE NW3 HOWEVER.
/ THE WORDS ARE ASSUMED TO BE IN PACKED FORM.
/
/MODE8 - DISPLAY NIC WORDS IN HEX AND PERMIT THE USER TO CHANGE THE
/ DISPLAYED WORD (ALSO IN HEX). THIS MODE CAN BE USED TO PUT PROGRAM
```

```
/ PATCHES INTO CTL.
    / THE MODE QUERY IS -
    /  OCT - ( REPLY WITH THE STARTING ADDRESS (OCTAL) OF NIC
    /        MEMORY TO BE OBSERVED AND/OR CHANGED )
    /  THE PROGRAM THEN DISPLAYS THE FIRST WORD AS 5 HEX DIGITS AND
    /  WAITS FOR USER RESPONSE. A SPACE (OR ANY SEQUENCE OF 1 TO 4
    /  CHARACTERS) THEN CR WILL CAUSE NO CHANGE.
    /  IF 5 HEX CHARACTERS ARE ENTERED THEY WILL REPLACE THE DISPLAYED
    /  WORD. THE NEXT WORD WILL THEN BE DISPLAYED, ETC. TO EXIT THIS
    /  MODE REPLY WITH CR ONLY. THE CHANGED CODE CAN BE
    /  LOADED INTO CTL MEMORY VIA MODE 1 WITH A NON-EXISTANT FILE
    /  NAME.
    /
    /
    / THE USER MAY EXIT CTLSYS VIA NICBUG TO NICSYS AND STORE THIS PROGRAM
    / DATA FOR LATER USE. USE STORE NAME 100000-100632;100000:P
    /
    / NOTE -- WHEN RESPONDING TO A QUERY, THE USER MAY TYPE ^G TO ABORT
    / THE MODE. TYPING ^Q WILL BRING THE USER TO NICBUG (IF LOADED).
    / VALUE ASSIGNED TO ABORT.
    // PARAMETER TABLE
    COMN,   0
    NLIST,  0
    LIS1P,  0
    LIS1S,  0
    LIS2P,  0
    LIS2S,  0
    LIS3P,  0
    LIS3S,  0
    TALKP,  0
    TALKS,  0
    EOSC,   0
    NDAT,   0
    NDATB,  0
    DATAL,  0
    DATAH,  0
    RETAL,  0
    RETAH,  0
    MESS,   0
    DUM1,   0
    DUM2,   0
    /
    /
    /  DEFINITIONS
      LISTEN=40   /OCTAL BASE FOR LISTENERS
      TALK=100    /OCTAL BASE FOR TALK
      CTL=1   /ADDRESS ASSIGNED TO CONTROLLER
      CAMERA=2   /ADDRESS ASSIGNED TO CAMERA
      BASE=140   /BASE FOR CAMERA "SECONDARY ADDRESSES"
      OUTF=1   /OUTPUT FORMAT (1,2)
```

```
  INF=2  /INPUT FORMAT (1,2,3)
  XCOORD=3  /X-COORDINATE (0 - 1023)
  INTERL=4  /INTERLACE (1,2,4)
  HORRES=5  /HORIZONTAL RESOLUTION (1,2,3,4)
  EXTAN=6  /EXTERNAL ANALOG (1=OFF)
  MARK=7  /MARKER ON/OFF (1=ON)
  VIDIN=10  /VIDEO INPUT
  VIDINI=11  /VIDEO IN & INCREMENT
  VIDIND=12  /VIDEO IN &DECREMENT
  SLICE=13  /SLICE INPUT
  SLICEI=14  /SLICE IN & INCREMENT
  SLICED=15  /SLICE IN & DECREMENT
  BUFFER=16  /BUFFER DISPLAY/
/
  CTLCF=4062
  CTLRD=44064
  CTLRDC=44066
  CTLSK=6064
  CTLWR=4071
  CTLRS=4072

// CAMERA TABLE (DEFINES CAMERA COMMANDS AND MNEMONICS)
/ EACH ENTRY CONTAINS THE FOLLOWING DATA IN THE SEQUENCE SHOWN-
/  MNEMONIC - 3 LETTERS IN PACKED ASCII (RIGHT JUSTIFIED)
/  CODE - OCTAL SECONDARY ADDRESS DEFINING THE COMMAND
/  NNIB - NO. OF NIBBLES IN THE COMMAND DATA (0-5)
/  C-DATA - UP TO 5 NIBBLES OF NUMERICAL DATA (LEFT JUST.)
CTABLE,  0576564  /OUTPUT FORMAT (OUT)
  BASE+OUTF
  1
  0200000  /DEFAULT 1
  0515646  /INPUT FORMAT (INF)
  BASE+INF
  1
  0200000  /DEFAULT 1
  0704357  /X-COORD (XCO)
  BASE+XCOORD
  3
  1211000  /DEFAULT 512
  0515664  /INTERLACE (INT)
  BASE+INTERL
  1
  0400000  /DEFAULT 2
  0554162  /MARKER ON/OFF (MAR)
  BASE+MARK
  1
  0  /DEFAULT 0
  0665121  /1-LINE VIDEO (VI1)
  BASE+VIDIN
```

```
   0
   0
   0665151 /1-LINE VIDEO + INCREMENT (VII)
   BASE+VIDINI
   0
   0
   0665144  /1-LINE+DECREMENT (VID)
   BASE+VIDIND
   0
   0
   0426546  /BUFFER DISPLAY (BUF)
   BASE+BUFFER
   0
   0
   300  /@ TO TERMINATE THE TABLE
START,  JMS  @CRLF
   JMS  @UNP
   7
   0
  .TEXT % MODE?-%
RPT,  JMS  @ECHO
   A-MZ  (223  /CONTROL G
   ZERZ
   JMP  @ABORT
   A-MZ  ("1
   ZERZ
   JMP  MODE1
   A-MZ  ("2
   ZERZ
   JMP  MODE2
   A-MZ  ("3
   ZERZ
   JMP  MODE3
   A-MZ  ("4
   ZERZ
   JMP  MODE4
   A-MZ  ("5
   ZERZ
   JMP  MODE5
   A-MZ  ("6
   ZERZ
   JMP  MODE6
   A-MZ  ("7
   ZERZ
   JMP  MODE7
   A-MZ  ("8
   JMP  START
   JMP  MODE8
/
```

```
MODE1,  JMS  @GETFIL  /GET THE FILE AND STORE IT IN CORE
        ACCA  /IF NO FILE, ASSUME IT IS ALREADY IN CORE (NIC)
        JMS  @ZERTAB  /ZERO THE COMMAND TABLE
        TABLA
      / INITIALIZE THE CAMERA TABLE TO DEFAULT VALUES
        JMS  @SEARCH
        0576564
        0
        0200000
        1
        CTABLE
        1
        JMS  @SEARCH
        0515646
        0
        0200000
        1
        CTABLE
        1
        JMS  @SEARCH
        0704357
        0
        1211000
        3
        CTABLE
        1
        JMS  @SEARCH
        0515664
        0
        0400000
        1
        CTABLE
        1
        JMS  @SEARCH
        0554162
        0
        0
        1
        CTABLE
        1
        CTLRS  /RESET THE CTL
        ZERA  /FORCE CTL TO BOOT LOAD
        ZERM  SCTLC
        JMS  SCTL
        JMS  @UNPF
        SCTLA
        2000
        0
        100000
        CTLCF  /CLEAR
```

```
  JMP  START

/
////////////////////////////////////////////////////////////
/
MODE2,  JMS  @GETFIL  /PUT FILE IN NIC CORE
  JMP  START  /ERROR EXIT FROM GETFIL
  JMS  @CRLF
MODE21,  JMS  @UNP
  13
  0
  TEXT % COMMAND#?-%
  JMS  @ZERTAB
  TABLA
  JMS  @ECHO
  ACCM  TEMP  /COMMAND SHOULD BE ASCII 267 OR 270
  A-MZ  (267
  JMP  #+3
  MEMA  (14  /STORE AT C00
  JMP  #+4
  A-MZ  (270
  JMP  MODE21  /WRONG COMMAND #
  MEMA  (16  /STORE AT E00
  ACCM  DATAH
  MEMA  BUFS  /SET UP COMMAND TABLE FOR FILE TRANSFER
  ACCM  NLIST
  MEMA  (5
  ACCM  COMN
  MEMA  (2  /ASSUME 2 256 BYTE BLOCKS FOR TRANSFER
  ACCM  NDATB
  ZERM  DUM1
  JMS  COM5
  MEMA  TEMP
  A-MZ  (267
  JMP  #+3
  JMS  @COM7
  JMP  #+2
  JMS  @COM8
  JMP  START
```

```
/
      /
      ////////////////////////////////////////////////////////////
      /
      MODE3,  JMS  FILEQ
        JMS  @PKR
        FIL1
      NW1,  0
        JMS  @CRLF
        MEMA  (3
        A+MA  NW1
        ACCM  NW2
        JMS  @UNP  /SEND COMMAND TO COMM-STOR TO SEND DATA
      NW2,  0
        1
        16300
      FIL1,  BLOCK  2
        0770000
        JMS  @PAKF  /RECEIVE THE DATA AND PACK IT
        COMSTO
        100000
        0
        100000
      NBYTES,  0
        MEMAZ  NBYTES
        ACCM  NW3
        A-MA  (4
        EXCT  AC19
        JMP  MODE3
        JMS  @TEKHEX
      NW3,  0
      NBY5,  0
        MEMA  NW3
        ACCM  NW4
        MEMA  BUFS
        ACCM  @OARG3
        JMS  @NICFIL
      NW4,  0
        FIL1
        JMP  MODE31  /ERROR RETURN FROM NICFIL-NO ROOM
        JMP  START
      MODE31,  JMS  @UNP
        21
        TEXT % NO ROOM FOR FILE%
        JMP  @ABORT
      COMSTO,  0
        TTYRF
        JMP  #-1
        RDTTY
        A-MZ  (232  /CHECK FOR ^Z (END OF FILE)
```

```
  JMP  @COMSTO
  ONEM  @PCOUNT
  JMP  @COMSTO
/ NOTE THAT SETTING PCOUNT TO 1 FORCES PAKF TO STOP. COMM-STOR
/    SHOULD BE CONFIGURED TO SEND EOF.
// SUBROUTINE FILEQ
/ PURPOSE -- SEND MESSAGE - FILE NAME?-
FILEQ,  0
  JMS  @CRLF
  JMS  @UNP
  14
  0
  TEXT  % FILE NAME?-%
  JMP  @FILEQ
/
/
////////////////////////////////////////////////////////////////////
/
MODE4,  JMS  @ZERTAB
  TABLA
  MEMA  (5
  ACCM  COMN  /INITIALIZE COMMAND TABLE FOR DATA TRANSFER
  MEMA  (311  /FROM NIC TO CTL
  ACCM  NLIST
  MEMA  (14
  ACCM  DATAH
  MONM  DUM1  /WILL BE SENDING NIBBLES
MODE41,  JMS  @UNP
  13
  0
  TEXT % MNEMONIC?-%
  JMS  @PKR
  FIL2
NC,  0
  JMS  @CRLF
  ZERM  FLAG
  MEMA  NC
  A-MZ  (3  /MUST HAVE 3 CHARS.
  JMP  MODE41
  JMS  @SEARCH
FIL2,  0
CODE,  0
VAL,  0
NBYTE,  0
  CTABLE
FLAG,  0
  MEMZ  FLAG
  JMP  MODE41  /NON-ZERO FLAG MEANS MNEMONIC NOT FOUND
  MEMAZ  NBYTE
```

```
  JMP  #+2
  JMP  MODE42
  ACCM  NBY1
  MEMA  FIL2
  ACCM  MM1
  JMS  @UNPF  /DISPLAYS THE NIBBLES
  TYPE1  /TYPE1 CONVERTS 4 BIT HEX TO NIC-ASCII AND TYPES IT
NBY1,  0
  7777777  /-1 FOR NIBBLES
  VAL
  JMS  NIB  /PACK THE USER GIVEN NIBBLES
  VAL4
NBY2,  0
  JMS  @CRLF
  MEMAZ  NBY2  /IF NO BYTES, USE DEFAULT
  JMP  #+6
  MEMA  VAL
  ACCM  LIS1P
  MEMA  NBYTE
  ACCM  NDAT
  JMP  MODE43
  ACCM  NDAT
  ACCM  NBY4
  MEMA  VAL4
  ACCM  LIS1P
  ONEM  FLG4
  JMS  @SEARCH  /UPDATE THE C-TABLE WITH THE NEW VALUE
MM1,  0
  0
VAL4,  0
NBY4,  0
  CTABLE
FLG4,  0
MODE43,  JMS  COM5
MODE42,  ONEM  COMN  /SET UP COMMAND TABLE FOR CTL-CAM
  ONEM  NLIST
  MEMA  (LISTEN+CAMERA
  ACCM  LIS1P
  MEMA  CODE
  ACCM  LIS1S
  JMS  COM1
MODE44,  JMP  START
// FUNCTION TYPE1
/ PURPOSE -- CONVERT 4 BIT HEX (LEFT JUSTIFIED) IN ACC TO 8-BIT
/  ASCII AND TYPE IT.
TYPE1,  0
  A+MA  (260
  ACCM  TTEMP
  M-AA  (271
```

```
  SKIP  AC19
  JMP  #+3
  MEMA  (7
  A+MM  TTEMP
  MEMA  TTEMP
  JMS  @TYPE
  JMP  @TYPE1
TTEMP,  BLOCK 2
////////////////////////////////////////////////////
// SUBROUTINE NIB(VALA,NNIB)
/ PURPOSE -- PACKS NNIB USER GIVEN NUMBERS INTO THE LOCATION
/  GIVEN BY VALUA. UP TO 5 NIBBLES ,LEFT JUSTIFIED 0 FILL
/  MAY BE PACKED. THE NUMBER OF NIBBLES IS RETURNED IN NNIB
/  AND IS DETERMINED BY CR.
NIB,  0
  JMS  @PAKF
  ECHO1
  6
  0
  PENDA
NNIB,  0
  MEMA  NNIB
  ACCM  @COUNTN
  MEMA  (6
  ACCM  @WCNT
  MMOA  PEND
  ACCM  @WPNT
  ZERM  @WORD
  MEMA  PEND
  AMOM  @APNT
  ONEM  @BCNT
  JMS  @DEC
  MEMA  @NIB
  ACCM  TEMP
  MEMA  @WORD
  ACCM  @TEMP
  MPOM  NIB
  MEMA  NNIB
  AMOM  @NIB  /-1 BECAUSE OF CR
  A-MZ  (6  /IF HAVE FULL WORD, GET FROM PEND
  JMP  NIB3
  MEMA  @PEND
  ACCM  @TEMP
NIB3,  MPOM  NIB
  JMP  @NIB
/COMMON DEFINITION
WPNT,  WPNTD
WORD,  WPNTD+1
WCNT,  WPNTD+2
```

```
COUNTN,  WPNTD+3
BCNT,  WPNTD+4
APNT,  WPNTD+5
////////////////////////////////////////////////////////}
ECHO1,  0
  JMS  @ECHO
  A-MZ  (215
  JMP  @ECHO1
  ONEM  @PCOUNT
  JMP  @ECHO1
```

```
/
/
////////////////////////////////////////////////////////////
/
/
MODE5,  JMS  @ZERTAB
  TABLA
  MEMA  T3  /SET DIVIDE ARGUMENTS
  ACCM  REM2
  MEMA  (5
  ACCM  REM
  ACCM  REM1
  MEMA  PA1  /DEFAULT PAUSE CONSTANT
  ACCM  PAUC
  MEMA  (44  /SET POLL STATUS BYTE COUNTER
  ACCM  NCNT
  JMS  FILEQ  /GET FILE NAME, MAY BE "PRINT"
  JMS  @PKR
  FILNM
  0
  JMS  @CRLF
  JMS  @SEARCH  /GET SOME CAMERA PARAMETERS TO DETERMINE
  0704357  /SPACE REQUIREMENTS
  0
VALM5,  0
NBM5,  0
  CTABLE
  0
  MEMA  VALM5
  ACCM  VALM51
  MEMA  NBM5
  ACCM  NBM51
  JMS  @NIBBIN  /CONVERT TO BINARY INTEGER
VALM51,  0
NBM51,  0
  12
BIN,  0
  JMS  @SEARCH  /LOOK AT INTERLACE TO GET NO. OF ELEMENTS/LINE
  0515664
  0
VALM52,  0
  0
  CTABLE
  0
  MEMA  VALM52  /GET INTERLACE NIBBLE AND CONVERT TO INTEGER
  LLSH  4
  ACCM  INT
  ONEM  N  /N IS THE NO. OF LINES/FRAME; DEFAULT TO 1
MOD50,  JMS  @UNP
  10
```

```
  0
  TEXT % FRAME?-%
  JMS  @PKR
  FILM5
  0
  JMS  @CRLF
  MEMA  FILM5
  ACCM  FILM51
  ZERM  FLGM5
  JMS  @SEARCH  /GET CORRESPONDING CODE
FILM51,  0
CODM5,  0
  0
  0
  CTABLE
FLGM5,  0
  MMOZ  FLGM5  /FLAG=1 MEANS WRONG COMMAND
  JMP  #+2
  JMP  MOD50
  MEMA  CODM5
  A-MZ  (BASE+VIDIN
  JMP  #+2
  JMP  MODE51  /N=1 FOR VIDIN,ELSE CALCULATE N=
  A-MZ  (BASE+VIDINI
  JMP  MODE52  /(1024-XCOORD)*INT/4 OR
  MEMA  (1777  / (XCOORD+1)*INT/4
  APOA
  A-MA  BIN
  JMP  MODE53
MODE52,  A-MZ  (BASE+VIDIND
  JMP  MOD50
  MPOA  BIN
MODE53,  JMS  @MULTP
INT,  0
  RISH  2  /DIVIDE BY 4
  ACCM  N
MODE51,  JMS  @SEARCH  / FIND THE NO. OF BYTES/LINE
  0515646  /INF
  0
VALM53,  0
  0
  CTABLE
  0
  ZERM  FACT1
  MEMA  VALM53
  LLSH  4  /CHANGE TO 1,2 OR 3
  ACCM  VALM53
  MEMA  (4  /NO. OF BYTES/LINE=256*INT*
  ACCM  FACTOR  / (4 OR 1 OR 4  +  0 OR 0 OR 1/16)
```

```
  MEMA  VALM53
  A-MZ  (2
  JMP   #+2
  ONEM  FACTOR
  A-MZ  (3
  JMP   #+3
  MEMA  (20
  ACCM  FACT1
  MEMA  INT
  JMS   @MULTP
FACTOR,  0
  ACCM  NDATB1  /NO. OF 256 BYTE BLOCKS/LINE
  JMS   @MULTP
  400
  A+MA  FACT1
  ACCM  NWORD   /TOTAL NO. OF BYTES/LINE
/ IF INF IS NOT 2 OR IF THE FILE NAME IS PRI(NT), THE CAMERA
/ VIDIO IS PRINTED ONLY AND NO FILE IS CREATED ON THE DISK.
  ONEM  PRTFLG  /1 MEANS THIS IS A "PRINT FILE"
  ONEM  J1
  ZERM  @SMODE  /SET SENDF MODE SWITCH
  MEMA  (100
  ACCM  @SWCNT
  ACCM  @SWCNT0
  MEMA  VALM53
  A-MZ  (2
  JMP   MODE55
  MEMA  FILNM   /SEE IF PRINT
  A-MZ  PRI     /PRI=PACKED "PRI"
  JMP   MOD54
  ONEM  @SMODE
  MEMA  (40
  ACCM  @SWCNT
  ACCM  @SWCNT0
  JMP   MODE55
/ FOR A DISK FILE , WE HAVE TO ESTIMATE THE SPACE REQUIRED, OPEN THE
/ FILE ETC.
MOD54,  ZERM  PRTFLG
  MEMA  NWORD   /BYTES/LINE*N*TOTAL NO. OF BYTES
  ACCM  #+4
  MEMA  N   /THEY ARE REDUCED BY 2/5 IN PACKING
  LASH  1   /MULT BY 2
  JMS   @MULTP
  0
  JMS   @DIVDE  /DIVIDE BY 5
REM,  0
  APOM  SIZE   /MUST BE A REMAINDER OF AT LEAST 1
  MEMA  REM   /TAKE CARE OF THE REMAINDER
  A-MA  (3
```

```
  SKIP  AC19
  MPOM  SIZE
  MEMA  SIZE  /TOTAL NO. OF PACKED WORDS
  ACCM  @OARG2
  JMS   @OPENW  /CHECKS TO MAKE SURE THERE IS ROOM
  MEMA  @OARG2
  A-MZ  SIZE
  SKIP  AC19
  JMP   MODE56
  JMS   @UNP
  10
  1
  TEXT % NO ROOM%
  JMP   MOD5E
MODE56,  MEMA  @OARG1
  ACCM  IT0  /STARTING TRACK FOR THE FILE
  ACCM  IT
  MEMA  INT  /NEXT GET APPROPRIATE VALUES FOR SOME LOOP
  A-MZ  (4  /LIMITS FOR STORING THE DATA
  JMP   MODE57  /AND FOR PAUSE
  MEMA  (17
  ACCM  J1
  MEMA  (4
  ACCM  L1
  MEMA  PA4
  ACCM  PAUC
  JMP   MODE55
MODE57,  A-MZ  (2
  JMP   MODE58
  MEMA  (36
  ACCM  J1
  MEMA  (2
  ACCM  L1
  MEMA  PA2
  ACCM  PAUC
  JMP   MODE55
MODE58,  MEMA  (74
  ACCM  J1
  MEMA  (20
  ACCM  L1
MODE55,  MEMA  N  /SET COUNTER WITH THE TOTAL NO. OF LINES
  ACCM  COUNT
/SET UP THE FIXED PORTION OF THE COMMAND TABLE
  MEMA  (14
  ACCM  DATAH
  MEMA  (TALK+CAMERA
  ACCM  TALKP
  ONEM  DUM1  /DATA IS PACKED UNLESS PRINTED
  MMOZ  PRTFLG
```

```
  ZERM  DUM1
MODE5I,  ZERM  NBYT5  /I LOOP
  MEMA  J1
  ACCM  J
MODE5J,  ONEM  COMN  /SEND
  ONEM  NLIST
  ZERM  NDAT
  ZERM  NDATB
  MEMA  CODM5
  ACCM  LIS1S
  MEMA  (LISTEN+CAMERA
  ACCM  LIS1P
  JMS  COM1
  MEMA  (4  /POLL
  ACCM  COMN
  JMS  COM4
  MEMA  (2  /RECV
  ACCM  COMN
  ZERM  NLIST
  MEMA  NDATB1
  ACCM  NDATB
  JMS  @PAUSE
PAUC,  Ø
  JMS  COM2
  MEMA  (6  /NICO
  ACCM  COMN
  MEMA  BUFS
```

```
ACCM  NLIST
      MEMA  J
      A-MZ  J1
      MONM  NLIST  /DON'T RESET COM6 EXCEPT WHEN J=1
      JMS  COM6
      MEMA  NWORD
      A+MM  NBYT5  /ACCUMULATE TOTAL NO. OF BYTES
      MMOMZ  COUNT
      JMP  #+2
      JMP  MOD5J1
      MMOMZ  J
      JMP  MODE5J
      MEMA  (4
      ACCM  NTRCK
      JMP  MOD5J2
    MOD5J1,  MEMA  NBYT5
      JMS  @MULTP
      2
      JMS  @DIVDE  /DIVIDE BY 5
    REM1,  0
      APOM  SUM  /MUST HAVE A REMAINDER
      MEMA  REM1
      A-MA  (3
      SKIP  AC19
      MPOM  SUM
      ZERA  /GET READY FOR DIVIDE
      TACMQ
      MEMA  SUM
      JMS  @DIVDE
    REM2,  0
      ACCM  NTRCK
      MEM2  REM2
      MPOM  NTRCK
    MOD5J2,  MMOZ  PRTFLG  /PRINT OR STORE?
      JMP  MOD5J3
      JMS  @UNPF
      SENDFA
    NWORD,  0
      1  /DATA IS NOT PACKED FOR PRINT
      100000
      JMP  MOD5J4
    MOD5J3,  MEMA  NTRCK  /STORE THE PACKED DATA ON NTRCKS TRACKS
      ACCM  K
      MEMA  BUFS  /POINTS TO START OF BUFFER AREA
      ACCM  ISTART
    MOD5K,  JMS  @WRITE  /START OF K LOOP
    IT,  0
    TRKSZ,  3000
    ISTART,  0
      MPOM  IT
```

```
  MEMA  TRKSZ
  A+MM  ISTART
  MMOMZ  K
  JMP  MOD5K
MOD5J4,  MEMZ  COUNT
  JMP  MODE5I  /END OF I-LOOP
  MMOZ  PRTFLG  /IF THIS IS A PRINT FLAG GOTO THE END
  JMP  #+2
  JMP  MOD5E
  MEMA  @SYSTRT  /SAVE TO RESTORE
  ACCM  TEMP
  MEMA  INT  /CLOSE THE FILE
  RLSH  3  /THE INTERLACE NO. IS STORED AS THE 2 HIGH
  A+MA  N  /ORDER BITS OF THE PSA PORTION OF THE
  ACCM  @SYSTRT  /DIRECTORY ENTRY; THE NO. OF LINES
  MEMA  IT0  /IN THE LOW ORDER BITS
  ACCM  @OARG1
  MEMA  IT
  A-MA  IT0
  JMS  @MULTP
  3000
  ACCM  @OARG2
  MEMA  BUFS
  ACCM  @OARG3
  JMS  @CLOSE
FILNM,  0
  0
  MEMA  TEMP
  ACCM  @SYSTRT
MOD5E,  JMP  START
LIST
/
MODE6,  JMS  @UNP
  5
  0
  TEXT %HEXN-%
  JMS  NIB
  N
NBY6,  0
  MEMA  NBY6
  A-MZ  (5
  JMP  MODE6  /MUST GET 4 NIBBLES FOR ADDRESS AND
  MEMA  N  /1 FOR NO. OF BLOCKS (EG. 0C001)
  ANDA  (7
  ACCM  NDATB
  MEMA  N
  RLSH  4  /GET LOW ADDRESS
  ANDA  (377  /MASK IT
  ACCM  DATAL  /STORE FOR TRANSMISSION
```

```
  MEMA  N
  RLSH  14  /GET HIGH ADDRESS
  ANDA  (377
  ACCM  DATAH
  ZERM  DUM1
  MEMA  BUFS
  ACCM  NLIST
  MEMA  (6
  ACCM  COMN
  JMS  COM6
  MEMA  NBYT1  /TRANSFER NO. OF BYTES RECEIVED
  ACCM  NBY5  /FOR POSSIBLE MODE 7 CALL.
  JMP  START
  /
MODE7,  ONEM  @SMODE  /SET UP FOR SENDF
  MEMA  (20
  ACCM  @SWCNT
  ACCM  @SWCNT0
  JMS  @CRLF
  JMS  @CRLF
  MEMA  NBY5  /NO. OF BYTES
  ACCM  MOD71
  JMS  @UNPF
  SENDFA
MOD71,  0
  0
  100000
  JMS  @CRLF
  JMS  @CRLF
  JMP  START
/
MODE8,  JMS  @UNP
  4
  0
  TEXT  %OCT-%
  JMS  @OCT  /PACK OCTAL ADDRESS INTO MOD8A
  MOD8A
  MMOA  MOD8A  /STORE POINTER FOR CHANGE WORDS
  ACCM  COUNT
  ZERM  J  /FOR COUNTING WORDS
MOD81,  JMS  @CRLF
  JMS  @UNPF  /DISPLAY CONTENTS OF NEXT WORD
  TYPE1    /AS 5 HEX CHARACTERS
  5
  3777777
MOD8A,  0
  MONM  MOD8A  /CAUSES UNPF TO KEEP GOING WITHOUT
  MEMA  (240  /REINITIALIZING
  JMS  @TYPE  /PUT IN SPACE
```

```
  JMS  NIB  /COLLECT THE NIBBLES
  N
NBY8,  0
  MPOM  COUNT
  MPOM  J
  MEMAZ  NBY8  /0 NIBBLES MEANS EXIT
  ZERZ
  JMP  MOD8E
  A-MZ  (5  /1-4 NIBS MEANS NO CHANGE
  JMP  MOD81
  MEMA  N
  ACCM  @COUNT
  JMP  MOD81
MOD8E,  MEMA  J
  JMS  @MULTP
  5
  RASH  1  /NO. OF BYTES=5/2 * NO. OF WORDS
  ACCM  NBY5
  JMP  START
////////////////////////////////////////////////////////////////////}
/
/SCRATCH STORAGE
TEMP,  0
N,  0
FILM5,  0
NDATB1,  0
PRTFLG,  0
J1,  0
SIZE,  0
IT0,  0
L1,  0
COUNT,  0
NBYT5,  0
J,  0
NTRCK,  0
FACT1,  0
SUM,  0
K,  0
```

```
//////  C O M M A N D  S U B R O U T I N E S  /////////////////
      /
      // SUBROUTINE SEND (ALIAS COM1)
      / REVISION -- NOVEMBER 25,1980
      / AUTHOR   -- BARRETT,TB
      / PURPOSE  -- SEND DATA FROM CTL TO LISTENER(S)
      / PARAMETERS USED -- NONE PARAMETERS IN THE PARAMETER
      /  TABLE ARE USED ONLY BY CTL-SEND
      ////////////////////////////////////////////////////////////////
      *1500
      COM1,  0
        JMS  @WCTL  /TRANSFER TABLE VALUES
        TABLA
        24
        JMS  @MONITOR  /CATCH TABLE TRANSFER BYTE FROM COM5
        JMS  @MONITOR  /WAIT UNTIL CTL IS DONE
        JMP  @COM1
      ////////////////////////////////////////////////////////////////////////
      // SUBROUTINE RECV (ALIAS COM2)
      / REVISION -- NOVEMBER 25,1980
      / AUTHOR   -- BARRETT,TB
      / PURPOSE -- TRANSFER DATA FROM TALKER TO CTL
      / PARAMETERS USED -- NONE
      ////////////////////////////////////////////////////////////////////////
      COM2,  0
        JMS  @WCTL
        TABLA
        24
        JMS  @MONITOR
        JMS  @MONITOR
        JMP  @COM2
      /
      ////////////////////////////////////////////////////////////////////////
      /
      //SUBROUTINE POLL (ALIAS COM4)
      / REVISION -- JANUARY 19,1981
      / AUTHOR   -- BARRETT,TB
      / PURPOSE -- CONDUCT A SERIAL POLL (THE STATUS BYTE IS TYPED)
      / PARAMETERS -- NONE
      COM4,  0
        JMS  @WCTL
        TABLA
        24
        JMS  @MONITOR
        JMS  @MONITOR
        ACCM  TEMP  /STORE STATUS
        RASH  4  /GET READY FOR FIRST HEX DIGIT
        JMS  @HEXT
        MEMA  TEMP
        JMS  @HEXT
```

```
  MMOMZ  NCNT
  JMP  COM4E
    MEMA  (44
  ACCM  NCNT
  JMS  @CRLF
  JMS  @MONITOR
COM4E,  JMP  @COM4
NCNT,  0
////////////////////////////////////////////////////////////////////////
/
// SUBROUTINE NICI  (ALIAS COM5)
/ REVISION -- DECEMBER 29,1980
/ AUTHOR   -- BARRETT,TB
/ PURPOSE  -- WRITE DATA FROM NIC TO CTL
/ PARAMETERS USED --

/  (1) "I" FOR IMMEDIATE DATA (THE DATA TO SEND IS IN TABLE
/  LOCATIONS 2 => 10), OR THE STARTING ADDRESS IN NIC OF THE
  /BLOCK OF DATA TO BE SENT (CAN NOT BE "I" = 311 OCTAL).
/  (11) NO.OF DATA WORDS (1 BYTE/WORD) TO BE SENT OR
/  (12) NO. OF 256 BYTE BLOCKS TO BE TRANSFERRED IF (11)=0.
/  (18) -1 => DATA IS PACKED NIBBLES (WHEN UNPACKING ADD OCTAL 60
/    TO TRANSFORM TO ASCII NUMBER.
/        0 => DATA IS PACKED
/        1 => DATA IS UNPACKED (5BYTES IN 2 WORDS)
/
/ NOTE -- THE STARTING ADDRESS IN (1) CAN BE -1 TO INDICATE THAT
/  THE UNPACKING PROCESS SHOULD CONTINUE FROM WHERE IT LEFT
/  OFF ON THE PREVIOUS CALL TO UNPF.
////////////////////////////////////////////////////////////////////////
COM5,  0
  JMS  @WCTL
  TABLA
  24
  JMS  @MONITOR
  MEMA  DUM1
  ACCM  FLAG5
  ACCM  SCTLC
  MEMA  NLIST  /IT IS IMMEDIATE MODE ?
  A-MZ  ("I
  JMP  #+2
  MEMA  (TABLA+2
  ACCM  STADD5
  MEMAZ  NDAT
  JMP  #+2
  JMP  #+3
  ACCM  NBYTE5
  JMP  COM51
  MEMAZ  NDATB
```

```
ZERZ
JMP  COM53
ACCM  COUNTS  /DO ADD INSTEAD OF MULT.
```

```
ZERA
     A+MA   (400
     MMOMZ  COUNTS
     JMP    #-2
     ACCM   NBYTE5
   COM51,   JMS   @UNPF
     SCTL
   NBYTE5,   0
   FLAG5,   0
   STADD5,   0
   COM53,   CTLCF
     JMS   @MONITOR
     JMP   @COM5
   ////////////////////////////////////////////////////////////////////////
   /
   //SUB ROUTINE NICO (ALIAS COM6)
   / REVISION -- DECEMBER 29,1980
   / AUTHOR   -- BARRETT,TB
   / PURPOSE --READ DATA FROM CTL TO NIC
   / PARAMETERS USED --
   /  (1) "I" FOR DATA TO BE STORED IN TABLE LOCATIONS 2 => 10,
   /  OR STARTING ADDRESS FOR DATA STORAGE.
   /  -1 MEANS USE LAST ADDRESS FROM PRIOR RUN
   /  (11) NO. OF DATA WORDS TO BE TRANSFERRED OR
   /  (12) NO. OF 256 BYTE BLOCKS TO BE TRANSFERRED IF (11)=0.
   /  (18) 1 => DO NOT PACK THE DATA
   /       0 => PACK THE DATA (5 BYTES/2 WORDS)
   COM6,   0
     JMS   @WCTL
     TABLA
     24
     JMS   @MONITOR
     MEMA   DUM1
     ACCM   FLAG6
     MEMA   NLIST
     A-MZ   ("I
     JMP   #+2
     MEMA   (TABLA+2
     ACCM   STADD6
     MEMAZ   NDAT
     JMP   #+2
     JMP   #+3
     ACCM   NBYT1
     JMP   COM61   /NOT 0
     MEMAZ   NDATB
     ZERZ
     JMP   COM63   /NOTHING TO TRANSFER
     ACCM   COUNTS
     ZERA
     A+MA   (400
```

```
  MMOMZ  COUNTS
  JMP  #-2
  ACCM  NBYT1
COM61,  JMS  @PAKF
  MONITA
NBYT1,  0
FLAG6,  0
STADD6,  0
  0
COM63,  JMS  @MONITOR
  JMP  @COM6
////////////////////////////////////////////////////////////////
/
/ SUBROUTINE SCTL
/ PURPOSE -- SEND BYTE TO CTL
/ THERE ARE 2 MODES OF OPERATION SET BY SCTLC. IF SCTLC=-1
/ THE BYTE IS SENT AS ASCII (60H IS ADDED TO ACC), OTHERWISE
/ IT IS SENT WITH NO CHANGE. SCTL MAY BE ABORTED
/ BY TYPING ANY CHARACTER ON THE TTY IN CASE THE CTL
/ GETS HUNG.
*1650
SCTL,  0
  MPOZ  SCTLC
  ZERZ
  A+MA  (60
  CTLWR
SC1,  CTLSK
  JMP  SC2
  JMP  @SCTL
SC2,  TTYRF
  JMP  SC1
  JMP  @ABORT
SCTLC,  0
```

```
/
      / SCRATCH STORAGE
      COUNTS,  0
      /
      / ADDRESSES
        TABLA=0
        SENDFA=4242
        SCTLA=1650
        MONITA=2123
        WPNTD=3414
        PENDA=1770
      / EXTERNALS
      MONITOR,  MONITA
      HEXT,  2330
      ECHO,  2257
      OPENW,  2420
      OPENR,  2432
      CLOSE,  2451
      WRITE,  2470
      READD,  2511
      PRTOCT,  2605
      UNP,  2650
      TYPE,  2731
      CRLF,  2736
      UNPF,  2750
      PAKF,  3074
      PKR,  3240
      DEC,  3310
      NIBBIN,  3470
      TEKHEX,  3565
      NICFIL,  4010
      SEARCH,  4070
      ZERTAB,  4160
      MULTP,  4175
      DIVDE,  4213
      WCTL,  2136
      GETFIL,  4313
      PAUSE,  4362
      OCT,  4371
      / DEFINITIONS AND COMMON
      OARG1,  7770
      OARG2,  7771
      OARG3,  7772
      PEND,  PENDA  /PAGE END FOR SCRATCH STORAGE
      BUFS,  100000
      SMODE,  SENDFA+47
      SWCNT,  SENDFA+46
      SWCNT0,  SENDFA+50
      ABORT,  4700
      SYSTRT,  7600
```

```
PCOUNT,  3223
/ CONSTANTS
PRI,  606251
T3,  3000
PA1,  400  /PAUSE CONSTANTS
PA2,  1000
PA4,  100000
```

```
//////////// S E R V I C E   S U B R O U T I N E S   ////////////
      /
      // FUNCTION ERROR
      /  REVISION -- JANUARY 2271981
      / AUTHOR    -- BARRETT,TB
      /PURPOSE  -- WHEN CTL-ERROR IS JUMPED TO A SERVICE BIT IS SET. WHEN
      /  MONITOR FINDS THIS , IT JUMPS TO ERROR WHICH PRINTS
      /  OUT SOME ERROR MESSAGES AS FOLLOWS-
      /  (1) THE NIC COMMAND BEING EXECUTED
      /  (2) A CTL ERROR STATUS BYTE (USUALLY CONTENTS
      /    OF THE A REGISTER)
      /  (3) THE PROGRAM COUNTER OF NIC AT THE ERROR
      /  (4) THE PROGRAM COUNTER OF CTL AT THE ERROR
      /
      /  NOTE THAT ON ENTRY TO ERROR ACC IS ASSUMED TO HOLD THE
      /  ERROR STATUS BYTE
        CTLCF=4062
        CTLRD=44064
        CTLRDC=44066
        CTLSK=6064
        CTLWR=4071
        CTLRS=4072
      *2020
      ERROR,  0
        ACCM  STAT  /STORE STATUS
        JMS  @UNP
        14
        0
        TEXT %ERROR IN COM%
        MEMA  @TABLE
        A+MA  (260  /COMMAND NO. TO ASII
        JMS  @TYPE
        JMS  @UNP
        23
        0
        TEXT %,STATUS BYTE (HEX)=%
        MEMA  STAT
        RASH  4  /CHANGE TO HEX ASCII
        JMS  HEXT
        MEMA  STAT
        JMS  HEXT
        JMS  @CRLF
        JMS  @CRLF
        JMS  @UNP
        14
        0
        TEXT %NIC PC(OCT)=%
        MEMA  ERROR
        AMOA
        JMS  @PRTOCT
```

```
  JMS   @CRLF
  JMS   @UNP
  14
  0
  TEXT %CTL PC(HEX)=%
  JMS   RCTL
  ACCM   TEMP
  RASH   4
  JMS   HEXT
  MEMA   TEMP
  JMS   HEXT
  JMS   RCTL   /GET LOW ORDER ADDRESS
  ACCM   TEMP
  RASH   4
  JMS   HEXT
  MEMA   TEMP
  JMS   HEXT
  JMS   @CRLF
  JMS   RCTL   /GET NORMAL RETURN BYTE
  JMP   @SYSTRT   /RETURN TO NIC MAIN MONITOR ON ERROR
STAT,   0
TEMP,   0

/
//FUNCTION MONITOR
/ REVISION -- JANUARY 22,1981
/ AUTHOR   -- BARRETT,TB
/ PURPOSE   -- "MONITOR" INPUT FROM CTL. IT READS DATA FROM CTL
/  AND JUMPS TO ERROR IF SRVC BIT IS SET. OTHERWISE IT
/  RETURNS THE BYTE READ IN ACC.
/
  SRVC=400
MONITOR,   0
MON1,   CTLSK
  JMP   MON2
  CTLRDC
  ANDZ   (SRVC
  JMS   ERROR
  JMP   @MONITOR
MON2,   TTYRF
  JMP   MON1
  RDTTY
  JMP   @ABORT
/
//SUBROUTINE WCTL(STADD,NBYTES)
/ REVISION -- JANUARY 22,1981
/ AUTHOR    -- BARRETT,TB
/ PURPOSE    -- TRANSFER NBYTES OF DATA FROM NIC MEMORY STARTING AT
/  ADDRESS STADD TO CTL
```

```
/ PARAMETERS --
/  STADD  STARTING ADDRESS OF DATA BLOCK IN NIC
/  NBYTES  SIZE OF DATA BLOCK IN BYTES (1 BYTE/NIC WORD
/
WCTL,  0
  ZERM  @SCTLC
  MEMA  @WCTL
  ACCM  POINT  /POINTS TO DATA BLOCK
  MPOM  WCTL  /GET COUNT
  MEMA  @WCTL
  ACCM  COUNT
  MPOM  WCTL  /SET FOR RETURN FROM WCTL
WCTL1,  MEMA  @POINT  /GET NEXT DATUM
  JMS  @SCTL
  MPOM  POINT
  MMOMZ  COUNT
  JMP  WCTL1
  CTLCF  /CLEAR DONE ON LAST WRITE
  JMP  @WCTL
POINT,  0
COUNT,  0
/
/
RCTL,  0
  CTLSK
  JMP  #-1
  CTLRDC
  JMP  @RCTL
```

```
//PROGRAM CTLTST
      /REVISION -- JANUARY 22,1981
     / AUTHOR --BARRETT,TB &TERPSTRA,D (U. OF FLORIDA)
     /PURPOSE  READS CTL "REGISTERS" (RHH)
     /    WRITES CTL "REGISTERS" (WHH)
     /    CLEARS CTL (C)
     /
     /  WHERE HH IS 2 HEX CHARACTERS REPRESENTING
     /  THE REGISTER (SEE WRITEUP ON NIC-488/CTL)
     /
     /DEFINITIONS FOR I/O COMMANDS TO CTL
     /
     *2170

     START,  JMS   ECHO
       ZERM  @SCTLC
       ONEM  CC  /CC IS USED TO HOLD READ WRITE BIT
       ZERM  TBIT
       A-MZ  ("R
       ZERZ
       JMP  SEND
       ZERM  CC
       A-MZ  ("W
       ZERZ
       JMP  SEND
       A-MZ  ("C
       ZERZ
       JMP  RESET
     ERR,  MEMA  ("?  /ILLEGAL CHAR
       JMS  @TYPE
       JMP  START

     SEND,  MEMA  CC
       LLSH  3
       ACCM  CC
     SEND1,  JMS  ECHO
       JMS  VALID  /NORMAL RETURN ONLY IF VALID HEX CHAR
       LLSH  4
       ACCM  SCHAR
       JMS  ECHO
       JMS  VALID  /8 BITS IS 2 HEX CHARS
       A+MMA  SCHAR
       A+MA   CC  /ADD READ WRITE BIT
       JMS  @SCTL
       CTLCF
       MEMZ  CC
       JMP  SEND2
       MEMZ  TBIT
       JMP  SEND2
       ONEM  TBIT
```

```
  MEMZ  SCHAR  /IF W00 THEN SEND DATA TO CTL
  JMP  #+5
  JMS  @WCTL
  100000
  2000
  JMP  @CALLS
  MEMA  ("-
  JMS  @TYPE
  JMP  SEND1
SEND2,  ACCM  SSIG
  RASH  4  /SHIFT FOR HIGH NIB
  JMS  HEXT  /CONVERT IT TO ASCII
  MEMA  SSIG
  JMS  HEXT  /CONVERT 2ND CHAR
  JMS  @CRLF
  JMP  START

SCHAR,  0
SSIG,  0
CC,  0
TBIT,  0

ECHO,  0
  JMS  READ
  A-MZ  (221  /^Q
  ZERZ
  JMP  @ABORT
  A-MZ  (207  /^G
  ZERZ
  JMP  @CALLS
  JMS  @TYPE
  JMP  @ECHO
READ,  0
  TTYRF
  JMP  #-1
  RDTTY
  JMP  @READ


RESET,  CTLRS
  JMP  START



VALID,  0  /PUTS OCTAL EQUIVALENT FOR HEX ASCII IN ACC
  ACCM  VC  /LEGAL CHARS ARE 260-272 (DIGITS)
  A-MA  (260  /AND 301-310 (A-F)
  EXCT  AC19
  JMP  ERR  /< 0
```

```
  MEMA  VC
  A-MA  (272
  EXCT  AC19
  JMP  NUM  /CHAR IS LEGAL
  MEMA  VC
  A-MA  (300
  EXCT  AC19
  JMP  ERR  /< A
  MEMA  VC
  A-MA  (310
  SKIP  AC19
  JMP  ERR  /> F
LETTER,  MEMA  VC  /ITS A LETTER
  A-MA  (267  /A-300+11
  JMP  @VALID
NUM,  MEMA  VC
  A-MA  ("0  /STRIP OFF BIAS
  JMP  @VALID
  VC,  0

HEXT,  0  /MASKS AND TYPES AS HEX
  ACCM  HT  /SAVE ACC
  MEMA  HC1  /INITIALIZE POINTER
  ACCM  HP
  MEMA  HT  /RECALL ACC
  ZERM  HC  /COUNTER
  ANDA  K17  /MASK LOWER NIBBLE
  A-MZ  HC  /MATCH COUNTER ?
  ZERZ
  JMP  HTYPE
  MPOM  HC  /NO, BUMP COUNTER
  MPOM  HP
  JMP  #-5
HTYPE,  MEMA  @HP
  JMS  @TYPE
  JMP  @HEXT

/ TEMP STORAGE AND TABLE
K17,  17
HC,  0
HC1,  HTOP
HP,  0
HT,  0
HTOP,  260
  261
  262
  263
  264
  265
```

```
   266
   267
   270
   271
   301
   302
   303
   304
   305
   306
/
/ EXTERNALS (INCLUDES COMMON AND TABLES)
UNP,  2650
CRLF,  2736
PRTOCT,  2605
SYSTRT,  7600
TYPE,  2731
ABORT,  4700
CALLS,  71
TABLE,  0
SCTL,  1650
SCTLC,  1663
```

```
// I/O SUBROUTINES OPENW,OPENR,CLOSE,WRITE,READD
     //SUBROUTINE OPENW
     / REVISION -- JANUARY 26,1981
      /  AUTHOR    - BARRETT,TB
      /  PURPOSE  -- OPENS A FILE BY LOCATING THE NEXT AVAILABLE
      /             TRACK AND AMOUNT OF SPACE AAILABLE.
      /
      /  PARAMETERS -- NONE. TRACK AND SPACE ARE RETURNED IN OARG1
      /               (7770) AND OARG2 (7771) RESPECTIVELY,IF OARG2=0.
     /  IF OARG2 IS SET TO THE NO. OF WORS IN THE FILE,OPENW
     / NOTE -- SET OARG1 TO 0 BEFORE CALLING.
     /  WILL FIND THE FIRST AVAILABLE SPACE.
     *2420
     OPENW,  0
       JMS  DIRFIN
       MONM  @DISOLV
       JMS  @DIRFUN
       1
       2
       NOFIL  /POINTS TO A VALUE OF 0
       ACCA
       JMS  DIROUT
       JMP  @OPENW
     /
     // SUBROUTINE OPENR(FILNAM)
      REVISION -- DECEMBER 30,1980
      AUTHOR   --BARRETT,TB
      PURPOSE  -- OPENS A FILE FOR READING BY RETURNING THE STARTING
     /  TRACK,FILE SIZE AND LOCATION IN CORE FOR
     /  STORAGE (AS GIVEN BY THE DIRECTORY)
     / ARGUMENTS --
     /  FILNAM - 2 WORD PACKED FILE NAME OF THE FILE TO BE OPENNED.
     / LOCATIONS OARG1,OARG2,OARG3 CONTAIN TRACK,SIZE AND CORE LOCATION FOR
     / THE FILE RESPECTIVELY. IF THE FILE IS NOT FOUND, OARG1 CONTAINS -1.
     OPENR,  0
       MEMA  @OPENR  /FILENAME
       ACCM  FILNM
       MPOM  OPENR
       MEMA  @OPENR
       ACCM  FILNM+1
       MPOM  OPENR  /SET RETURN ADDRESS
       JMS  DIRFIN
       JMS  @DIRFUN
       1
       2
       FILNM
       MONM  @OARG1  /FILE DOES NOT EXIST
       JMS  DIROUT  /RESTORE
       JMP  @OPENR
      /
```

```
 //  SUBROUTINE CLOSE(FILNAM)
 /  PURPOSE -- ADD A FILE TO THE DIRECTORY
 /  PARAMETERS -
 /  FILNAM - 6 CHAR. (PACKED FORM) FILE NAME (2 WORDS)
 /     CONTROL RETURNS AFTER THE FILE NAME
 /  BEFORE CALLING PUT THE STARTING TRACK IN 7770 AND THE
 /  FILE SIZE (WORDS) IN 7771. THE CODE ADDRESS CAN BE PUT INTO
/  OARG3 AND THE STARTING ADDRESS IN SYSTRT.
CLOSE,  0
  MEMA  @CLOSE  /TRANSFER FILENAME
  ACCM  FILNM
  MPOM  CLOSE
  MEMA  @CLOSE
  ACCM  FILNM+1
  MPOM  CLOSE
  JMS   DIRFIN
  JMS   @DIRFUN
  1
  1
  FILNM
  ACCA
  JMS   DIROUT
  JMP   @CLOSE
 /
 //  SUBROUTINE WRITE(IT,SIZE,ISTART)
 / PURPOSE -- SIMPLE WRITE TO DISK USING DEMON II DISK
 /  PARAMETERS --
 /  IT - STARTING TRACK
 /  SIZE - NO. PF WORDS IN BUFFER (STARTS AT ISTART)
/   ISTART - STARTING ADDRESS OF BLOCK TO TRANSFER.
WRITE,  0
  MEMA  @WRITE
  A+MA  DNO
  ACCM  IT
  MPOM  WRITE
  MEMA  @WRITE
  ACCM  SIZE
  MPOM  WRITE
  MEMA  @WRITE
  ACCM  ISTART
  MPOM  WRITE  /RETURN ADDRESS
  JMS   @DISK
IT,  0
SIZE,  0
ISTART,  0
  JMP   @WRITE
  DNO,  100000
// SUBROUTINE READD(IT,SIZE)
/ REVISION -- NOVEMBER 29,1980
```

```
/ AUTHOR   -- BARRETT,TB
/ PURPOSE  -- READ TRACK IT OF SIZE WORDS INTO BUFFER
/    STARTING AT 100000. NOTE THAT IF SIZE IS GREATER
/    THAN A TRACK, MORE THAN 1 TRACK WILL BE READ.
READD,  0
  MEMA  @READD
  A+MA  DNO  /ADD THE DISK NO.
  ACCM  ITT
  MPOM  READD
  MEMA  @READD
  ACCM  SIZZ
  MPOM  READD  /SET RETURN
  ZERA  /SIGNALS READ
  JMS   @DISK
ITT,  0
SIZZ,  0
  100000
  JMP  @READD
 /
DIRFIN,  0  /READ OUT 3000-7600,READ IN DIRFUN
  ONEA
  JMS  @DISK
  100001
  4600
  3000
  ZERMA  @DERRF
  JMS  @DISK
  100007
  600
  7000
  ACCA
  ZERM  @DEVDET
  JMP  @DIRFIN
 //
DIROUT,  0  /READ BACK 3000-7600
  ZERMA  @DERRF
  JMS  @DISK
  100001
  4600
  3000
  ACCA
  JMP  @DIROUT
 /
 / DEMON II REFERENCES
DIRFUN,  7000
DISK,  7612
DERRF,  7704
DISOLV,  7751
DEVDET,  7764
```

```
OARG1,  7770
OARG2,  7771
OARG3,  7772
/ SCRATCH STORAGE
FILNM,  BLOCK  2
NOFIL,  0
```

```
//   FUNCTION PRTOCT(X)
     / REVISION -- JANUARY 22,1981
      /  AUTHOR    -- BARRETT,TB
      /  PURPOSE   -- PRINT THE OCTAL VALUE OF THE CONTENTS OF ACC
      /
      /
     *2605
     PRTOCT,   0
       LLSH   2
       ACCM   TEMP
       ANDA   (3
       A+MA   (260
       JMS   TYPE
       MEMA   (7   /SET COUNTER
       ACCM   COUNT
     PRTO1,   MMOMZ   COUNT
       JMP   #+2
       JMP   PRTO2
       MEMA   TEMP
       LLSH   3
       ACCM   TEMP
       ANDA   (7
       A+MA   (260
       JMS   TYPE
       JMP   PRTO1
     PRTO2,   MEMA   (215
       JMS   TYPE
       MEMA   (212
       JMS   TYPE
       JMP   @PRTOCT
```

```
// SUBROUTIN  UNP(NC,INDIC,TEXT)
     /REVISION --NOVEMBER 22,1980
     /AUTHOR   -- BARRETT,TB
     /PURPOSE   -- UNPAK PACKED ASCII AND SENDS TO TTY FOR PRINTING.
     /  AN OPTIONAL CR/LF IS SENT ALSO.
     /
     /PARAMETERS --
     /  NC  NO. OF CHARACTERS IN THE PACKED TEXT. IF 0, THE
     /  TEXT IS ASSUMED TO BE TERMINATED WITH 77 (%) AND NC IS
     /  RETURNED AS THE NO. OF TEXT CHARACTERS (NOT INCLUDING %)
     /  INDIC  0 => NO CR/LF, 1 => CR/LF AT END OF TEXT.
     /  TEXT  THE PACKED TEXT.
     /
     *2650
     UNP  ,0
       MEMA  UNP  /STORE ADDRESS OF NC
       ACCM  NC
       MPOM  UNP  /STORE INDIC
       MEMA  @UNP
       ACCM  INDIC
       ONEM  INDIX  /SET PRINT/NOPRINT INDICATOR
       ZERM  NCC  /SET CHARACTER COUNTER TO 0
       ONEM  COUNT  /INITIALIZE 1,2,3 COUNTER
     LOOP,  MMOMZ  COUNT  /DECREMENT COUNTER. IF 0 GET NEXT WORD
       JMP  L1  /IF NOT 0, TYPE CHARATER
       MEMA  (3  /REINITIALIZE COUNTER
       ACCM  COUNT
       MPOM  UNP  /POINT TO NEXT WORD IN TEXT
       MEMA  @UNP  /GET WORD AND SHIFT IT
       LLSH  8
       ACCM  WORD  /STORE IT FOR FURTHWR WORK
     L1,  ANDA  (77  /MASK 6 LSDS
       A-MZ  (77  /CHECK FOR END OF TEXT
       JMP  #+2
       JMP  END
       MEMZ  @NC
       JMP  #+2
       JMP  L3
       ACCM  TEMP
       MEMA  @NC
       A-MZ  NCC  /CHECK TO SEE IF NC CHARS. SENT
       JMP  #+2
       ZERM  INDIX  /IF INDIX IS 0, CHARACTERS ARE NOT PRINTED
       MEMA  TEMP
     L3,  MEMZ  INDIX  /IF 0, DON'T PRINT
       JMP  #+2
       JMP  #+4
       MPOM  NCC
       A+MA  (240  /CONVERT TO UNPACKED ASCII
       JMS  TYPE
```

```
  MEMA  WORD
  LLSH  6  /SHIFT &STORE FOR NEXT CHAR.
  ACCM  WORD
  JMP  LOOP  /GET NEXT CHAR.
END,  MEMAZ  INDIC
  JMS  CRLF
  MPOM  UNP  /SET FOR RETURN
  MEMA  NCC
  ACCM  @NC  /RETURN CHAR. COUNT
  JMP  @UNP
NCC,  0
NC,  0
INDIX,  0
/
/////// SUBROUTINE TYPE //////
TYPE,  0
  TTYPF
  JMP  #-1
  PRTTY
  JMP  @TYPE
/
////// SUBROUTINE CRLF  //////////
CRLF,  0
  MEMA  (212
  JMS  TYPE
  MEMA  (215
  JMS  TYPE
  JMS  TYPE
  JMP  @CRLF
```

```
// SUBROUTINE UNPF(SENDF,NBYTES,FLAG,STARTA)
     / REVISION -- DECMEBER 24,1980
     / AUTHOR   -- BARRETT,TB
     / PURPOSE  -- TRANSFER DATA FROM CORE TO A DESTINATION SPECIFIED BY
     /  SENDF. THE DATA MAY BE UNPACKED IN THE PROCESS.
     /
     / PARAMETERS --
     /  SENDF - ENTRY POINT FOR ACCEPTING A WORD IN ACC (E.G. TYPE)
     /  NBYTES - NO. OF BYTES TO BE TRANSFERRED. (FOR PACKED DATA
     /    THERE ARE 2.5 BYTES/NIC WORD)
     /  FLAG -  1 => DO NOT UNPACK
     /    0 => UNPACK
     /    -1 => UNNIBBLE (5 NIBBLES/WORD)
     /  STARTA - STARTING ADDRESS AT WHICH TO OBTAIN DATA. IF
     /  SET TO -1, UNPF WILL USE THE POINTER FROM THE PREVIOUS CALL.
     *2750
     UNPF,  0
       MEMA  @UNPF
       ACCM  SENDF
       MPOM  UNPF
       MEMA  @UNPF
       ACCM  COUNT
       MPOM  UNPF
       MEMA  @UNPF
       ACCM  UNPFLG
       MEMA  (17
       MPOZ  UNPFLG
       MEMA  (377
       ACCM  MASK
       MPOM  UNPF
       MEMA  @UNPF
       EXCT  AC19  /IF NEG. THEN DONT INITIALIZE
       JMP  UNPFX
       ACCM  POINT
       JMP  UNPF1
     UNPFX,  MMOZ  UNPFLG  /TEST FLAG
       JMP  UNPFZ
     UNPF1,  MEMA  @POINT
       MPOM  POINT
       MMOZ  UNPFLG
       JMP  UNPFY
       JMS  @SENDF
       MMOMZ  COUNT
       JMP  UNPF1
       JMP  UNPFE
     UNPFY,  ACCM  TEMP
       MEMA  (6
       ACCM  BCOUNT
       JMP  #+4
     UNPF2,  MMOMZ  COUNT
```

```
  JMP   #+2
  JMP   UNPFE
UNPFZ,   MMOMAZ   BCOUNT
  JMP   #+2
  JMP   UNPF1
  A-MZ   (5
  JMP   UNPF3
UNPF22,   MEMA   TEMP
  MPOZ   UNPFLG
  LLSH   4
  LLSH   4
  ACCM   TEMP
UNPF21,   ANDA   MASK
  JMS   @SENDF
  JMP   UNPF2
UNPF3,   A-MZ   (4
  JMP   UNPF4
UNPF33,   MPOZ   UNPFLG
  ZERZ
  JMP   UNPF22
  MEMA   TEMP
  RISH   4
  ANDA   (360
  ACCM   TEMP1
  JMP   UNPF22
UNPF4,   A-MZ   (3
  JMP   UNPF5
  MPOZ   UNPFLG
  ZERZ
  JMP   UNPF22
  MEMA   @POINT
  MPOM   POINT
  LLSH   4
  ACCM   TEMP
  ANDA   (17
  A+MA   TEMP1
  JMS   @SENDF
  JMP   UNPF2
UNPF5,   JMP   UNPF22
UNPFE,   MPOM   UNPF   /RETURN
  JMP   @UNPF
SENDF,   0
COUNT,   0
UNPFLG,   0
POINT,   0
TEMP,   0
BCOUNT,   0
TEMP1,   0
TEMP2,   0
```

```
MASK,  0
////////////////////////////////////////////////////////////////////////
/
// SUBROUTINE PAKF(RECVF,NBYTES,FLAG,STARTA,NBYTR)
/ REVISION -- DECEMBER 31,1980
/ AUTHOR   -- BARRETT,TB
/ PURPOSE  -- TRANSFER DATA GIVEN BY RECVF TO CORE.
/  THE DATA MAY BE PACKED IN THE PROCESS. (THIS IS THE INVERSE
/  OF UNPF). USE OF ARGUMENTS IS THE SAME AS IN UNPF EXCEPT-
/  RECVF GETS A DATA BYTE AND GIVES IT TO PACKF VIA ACC. NOTE
/  THAT IT MAY BE NECESSARY FOR RECVF TO CONTROL THE NUMBER OF
/  BYTES TRANSFERRED BY STICKING A 1 IN COUNT1 WHEN THE LAST
/  BYTE HAS BEEN RECEIVED (E.G. AN EOF MARK IS DETECTED)
/  NBYTR - NO. OF BYTES RECEIVED.
////////////////////////////////////////////////////////////////////////
PAKF,  0
  MEMA  @PAKF
  ACCM  RECVF
  MPOM  PAKF
  MEMA  @PAKF
  ACCM  COUNT1
  MPOM  COUNT1
  MPOM  PAKF
  MEMA  @PAKF
  ACCM  PAKFLG
  MPOM  PAKF
  MEMA  @PAKF
  EXCT  AC19
  JMP   PAKFX
  ACCM  POINT1
  MEMA  (6
  ACCM  BCNT
  ZERM  NBYTES
  ZERM  WORD
PAKFX,  MMOZ  PAKFLG
  JMP   PAKF2
PAKF1,  MMOMZ  COUNT1
  JMP   #+2
  JMP   PAKFF
  JMS   @RECVF
  MPOM  NBYTES
  ACCM  @POINT1
  MPOM  POINT1
  JMP   PAKF1
PAKF2,  MMOMZ  COUNT1
  JMP   #+2
  JMP   PAKFE
  MMOMZ  BCNT
  JMP   #+3
```

```
   MEMA  (5
   ACCM  BCNT
   JMS   @RECVF
   ACCM  TMP
   MPOM  NBYTES
   MEMA  BCNT
   A-MZ  (5
   JMP   PAKF3
PAKF22,  MEMA  TMP
   RLSH  10
   A+MM  WORD
   JMP   PAKF2
PAKF3,   A-MZ  (4
   JMP   PAKF4
   MEMA  TMP
   LLSH  4
   A+MM  WORD
   JMP   PAKF2
PAKF4,   A-MZ  (3
   JMP   PAKF5
   MEMA  TMP
   RISH  4
   A+MA  WORD
   ACCM  @POINT1
   MPOM  POINT1
   MEMA  TMP
   ANDA  (17
   RLSH  4
   ACCM  WORD
   JMP   PAKF2
PAKF5,   A-MZ  (2
   JMP   PAKF6
   MEMA  TMP
   LLSH  10
   A+MM  WORD
   JMP   PAKF2
PAKF6,   MEMA  TMP
   A+MA  WORD
   ACCM  @POINT1
   MPOM  POINT1
   ZERM  WORD
   JMP   PAKF2
PAKFE,   MMOZ  BCNT
   ZERZ
   JMP   #+3
   MEMA  WORD
   ACCM  @POINT1
PAKFF,   MPOM  PAKF
   MEMA  NBYTES
```

```
  ACCM  @PAKF
  MPOM  PAKF
  JMP  @PAKF  /NOTE THAT WE HAVE TO STORE THE LAST UNFILLED WORD
/      WHICH MAY BE LATER OVER WRITTEN.
RECVF,  0
COUNT1,  0
POINT1,  0
PAKFLG,  0
NBYTES,  0
TMP,  0
WORD,  0
BCNT,  0
```

```
// SUBROUTINE PKR(FILNAM,NF)
     / REVISION -- JANUARY 22,1981
     / AUTHOR   -- BARRETT,TB
     / PURPOSE  -- PACK USER GIVEN CHARACTERS INTO A 2-WORD
     /    "FILE-NAME". THE 2 MOST SIGNIFICANT BITS
     /     OF THE FILENAME ARE 00.
     / ARGUMENTS --
     /  FILNAM - ADDRESS OF THE FIRST WORD OF THE FILENAME
     /  NF - (RETURNED) THE NO. OF CHARACTERS IN THE FILENAME.
     *3240
     PKR,  0
       MEMA  @PKR
       ACCM  ADDR  /STORE THE ADDRESS OF THE FILENAME
       ZERM  @ADDR
       MPOAM  PKR
       ACCM  NF  /STORE ADDRESS FOR RETURNING NF
       ZERM  @NF
       MEMA  (6
       ACCM  COUNT
       ZERM  INDIC  /SET CR INDICATOR
     PKR1,  A-MZ  (3
       JMP  #+3
       MPOM  ADDR
       ZERM  @ADDR
       MEMZ  INDIC  /IF INDIC HAS BEEN SET THEN JUST SHIFT
       JMP  PKR2
       JMS  @ECHO
       A-MZ  (215
       JMP  #+4
       ONEM  INDIC
       ZERM  TEMP
       JMP  PKR2
       MPOM  @NF
       A-MA  (240
       ACCM  TEMP
     PKR2,  MEMA  @ADDR
       LLSH  6
       A+MA  TEMP
       ACCM  @ADDR
       MMOMAZ  COUNT
       JMP  PKR1
       MPOM  PKR  /INCREMENT FOR RETURN
       JMP  @PKR
     ADDR,  0
     NF,  0
     INDIC,  0
     /
     / EXTERNALS
     ECHO,  2257
```

```
// SUBROUTINE DEC
     / REVISION -- DECEMBER  30,1980
     / AUTHOR   --BARRETT,TB
     / PURPOSE  -- TRANSFORM A PACKED ASCII HEX STRING TO BINARY AND PACK
     /  5 NIBBLES PER NIC WORD
     / ARGUMENTS -- ALL ARGUMENTS ARE PASSED THROUGH A COMMON AREA
     /  WITH THE FOLLOWING VARIABLES IN THE ORDER SHOWN-
     /  WPNT - POINTS TO STORAGE LOCATION OF THE LAST WORD
     /    STORED. THIS IS INCREMENTED WHENEVER A WORD IS
     /    COMPLETE SO IT SHOULD BE SET ACCORDINGLY
     /    ON INITIAL ENTRY.
     /  WORD - CONTAINS THE NIBBLES OR PORTIONS THEREOF TO
     /    BE STORED AT WPNT+1. IT SHOULD BE SET TO 0 ON
     /    INITIAL CALL.
     /  WCNT - A COUNTER FOR WORD. WHEN WORD IS EMPTY,WCNT
     /    =5, WHEN FULL WCNT=0. WHEN WCNT GOES TO 0, WORD IS
     /    STORED AT WPNT+1 AND WPNT IS INCREMENTED. SET TO 6
     /    AT INITIAL CALL TO DEC.
     /  COUNT - THE NUMBER OF NIBBLES+1 TO BE PACKED. NOTE THAT COUNT
     /    IS DECREMENTED TO 0 BY DEC.
     /  BCNT - BYTE COUNTER. SET TO 1 FOR INITIAL CALL.
     /  APNT - POINT TO CURRENT STRING WORD. SET TO 1 LESS THAN THE
     /    START OF THE STRING INITIALLY.
     /  CHKSUM - NIBBLE VALUES ARE ADDED & STORED IN CHKSUM. SET TO
     /    0 ON EACH CALL TO DEC (USUALLY).
     /  NWORD - NO. OF WORDS STORED. (USUALLY SET TO 0 AT EACH CALL.
     /  NOTE----
     /  IN ORDER TO FORCE A WORD OUT OF DEC, SET WCNT TO COUNT.
     /
     ////////////////////////////////////////////////////////////////////////////
     *3310
     DEC,  0
     START,  MMOMZ  BCNT
       JMP  #+7
       MEMA  (5
       ACCM  BCNT
       MPOM  APNT
       MEMA  @APNT
       LASH  1
       ACCM  TEMP
       MMOMZ  WCNT
       JMP  #+10
       MEMA  (5
       ACCM  WCNT
       MPOM  WPNT
       MEMA  WORD
       ACCM  @WPNT
       MPOM  NWORD
       ZERM  WORD
       MMOMZ  COUNT
```

```
      JMP   #+4
      MPOM   BCNT
      MPOM   WCNT
      JMP   @DEC   /NOTE EXIT
      MEMA   BCNT
      A-MZ   (3
      JMP   DEC1
      MPOM   APNT
      MEMA   @APNT
      ACCM   TEMP1
      RISH   3
      A+MA   TEMP
      JMP   DEC2
DEC1,   A-MZ   (2
      JMP   DEC3
      MEMA   TEMP1
      LASH   5
      ACCM   TEMP
      JMP   DEC2
DEC3,   MEMA   TEMP
DEC2,   EXCT   AC19
      A+MA   TRAN
      LASH   3
      ACCM   TEMP
      ANDA   MASK
      ACCM   NIBBLE
      LLSH   4
      A+MM   CHKSUM
/ GET READY FOR NEXT WORD
      MEMA   TEMP
      LASH   5
      ACCM   TEMP
      MEMA   (6
      ACCM   SHIF1
      MEMA   WCNT
      M-AM   SHIF1
      MEMA   NIBBLE
      MMOMZ   SHIF1
      ZERZ
      JMP   #+3
      RISH   4
      JMP   #-4
      A+MM   WORD
      JMP   START
/ SCRATCH STORAGE
TEMP,   0
NIBBLE,   0
TEMP1,   0
SHIF1,   0
```

```
/ MASKS
TRAN,   220000
MASK,   3600000
/COMMON STORAGE
WPNT,   0
WORD,   0
WCNT,   0
COUNT,   0
BCNT,   0
APNT,   0
CHKSUM,   0
NWORD,   0
```

```
// SUBROUTINE NIBBIN(VALUE,NNIB,C,BIN)
     / REVISION -- DECEMBER 30,1980
     / AUTHOR   -- BARRETT,TB
     / PURPOSE  -- CONVERT PACKED BCD OR BCH TO BINARY.
     / ARGUMENTS --
     /  VALUE - PACKED BCD OR BCH; MOST SIGNIFICANT NIBBLE
     /    AT UPPER ORDER LOCATION IN VALUE;LEFT JUSTIFIED.
     /  NNIB  - NO.OF 4 BIT NIBBLES TO BE CONVERTED
     /  C     - OCT 12 IF THIS IS A BCD VALUE
     /          OCT 20 IF THIS IS A BCH VALUE
     /  BIN   - RETURNS BINARY VALUE HERE
     /
     ////////////////////////////////////////////////////////////////////////
     *3470
     NIBBIN,  0
       MEMA  (6
       ACCM  SHIF
       MEMA  @NIBBIN
       ACCM  VALUE
       MPOM  NIBBIN
       ZERM  RESULT
       MEMAZ  @NIBBIN
       ZERZ
       JMP  NIB2
       ACCM  NNIB
       M-AM  SHIF
       MEMA  VALUE
       MMOMZ  SHIF
       ZERZ
       JMP  #+3
       RISH  4
       JMP  #-4
       ACCM  VALUE
       MPOM  NIBBIN
       MEMA  @NIBBIN
       ACCM  C
       MPOAM  NIBBIN
       ACCM  BIN
       MPOM  NIBBIN
       MEMA  VALUE
       ONEM  MPLCND
     NIB1,  ANDA  (17
       TACMQ
       MULT
     MPLCND,  0
       TMQAC
       A+MM  RESULT
       MMOMZ  NNIB
       JMP  #+2
       JMP  NIB2
```

```
  MEMA   MPLCND
  ACCM   #+4
  MEMA   C
  TACMQ
  MULT
  0
  TMQAC
  ACCM   MPLCND
  MEMA   VALUE
  RISH   4
  ACCM   VALUE
  JMP   NIB1
NIB2,   MEMA   RESULT
  ACCM   @BIN
  JMP   @NIBBIN
VALUE,   0
NNIB,   0
C,   0
BIN,   0
RESULT,   0
SHIF,   0
```

```
// SUBROUTINE TEKHEX(CNT,NBYTE)
     / REVISION -- JANUARY 24,1981
     / AUTHOR   -- BARRETT,TB
     / PURPOSE  -- DECODES (TO BINARY) A TEKTRONIX HEX FILE AND PUTS
     /     THE RESULT IN PACKED FORM STARTING AT LOCATION
     /    100000. THE INPUT FILE IS ASSUMED TO BE IN
     /    PACKED FORM STARTING AT 100000.
     / ARGUMENTS -- CNT - TOTAL NO. OF PACKED WORDS IN THE OUTPUT.
     /    NOTE THAT THE LAST PACKED WORD MAY HAVE 0 FILL.
     /    ALSO NOTE THAT DEC PACKS WORDS SUCH THAT NIBBLE1
     /    OCCUPIES 19-16....,NIBBLE5, 3-0.
     /    NBYTE - TOTAL NO. OF BYTES (2 NIBS) STORED.
     /  - NOTE THAT THIS SUBROUTINE ASSUMES THAT THE TEK. 8002 AND
     /  COMM-STOR ARE CONFIGURED SUCH THAT EACH "LEADING" SLASH IS
     /  PREFACED WITH XOF (ASCII 223). ALSO NOTE THAT THERE ARE
     /  17 (21 OCT) SURPLUS CHARS. AFTER THE LAST CR.
     /  BE SURE TO USE MX MODE WHEN STORING THE HEX FILE.
     *3565
     TEKHEX,  0
       MEMA  TEKHEX
       ACCM  CNT
       ZERM  @CNT
       MPOMA  TEKHEX
       ACCM  NBYTE
       ZERM  @NBYTE
       MPOM  TEKHEX
       ONEM  BCNT  /INITIALIZE DEC
       ZERM  WORDO
       MEMA  (6
       ACCM  WCNTO
       MEMA  BUFS
       AMOA
       ACCM  APNT
       ACCM  WPNTO
     TEK1,  MEMA  (13
       ACCM  COUNT
       MEMA  (6
       ACCM  WCNT
       ZERM  WORD
       MEMA  HEADA  /ADDRESS OF HEADER START
       ACCM  WPNT
       JMS  DEC
       MEMA  HEAD  /CHECK FOR "/"
       LLSH  10
       ANDA  (17
       A-MZ  (17
       ZERZ
       JMP  TEK2
       JMS  @CRLF
       JMS  @UNP
```

```
   12
   1
   TEXT % TEKHEX ER%
   JMP  TEKE  / GOTO TEKHEX END
TEK2,  MEMA  HEAD+1
   LLSH  4
   ACCM  #+2
   JMS  NIBBIN
   Ø
   2
   2Ø
   Ø
   MEMA  #-1
   A+MM  @NBYTE
   LASH  1  /*2 FOR TOTAL NO. OF NIBBLES
   ACCMZ  COUNT
   JMP  #+2
   JMP  TEK3  /NORMAL TERMINATION-CHECK FOR WHETHER
/  THIS WAS LAST FULL BYTE.
   MPOM  COUNT
   MEMA  WPNTO
   ACCM  WPNT
   MEMA  WORDO
   ACCM  WORD
   MEMA  WCNTO
   ACCM  WCNT
   ZERM  CHKSUM
   ZERM  NWORD
   JMS  DEC
   MEMA  CHKSUM
   ANDA  (377  /THIS MOD 256
   ACCM  CHKS
   MEMA  NWORD
   A+MM  @CNT
   MEMA  WPNT  /STORE STATE FOR NEXT CALL
   ACCM  WPNTO
   MEMA  WORD
   ACCM  WORDO
   MEMA  WCNT
   ACCM  WCNTO
   MEMA  (4  /SET FOR GETTING TRAILER (3 NIBBLES)
   ACCM  COUNT
   MEMA  (4
   ACCM  WCNT
   ZERM  WORD
   MEMA  HEADA
   ACCM  WPNT
   JMS  DEC
   MEMA  HEAD
```

```
  LLSH  10
  ACCM  #+2
  JMS  NIBBIN
  0
  2
  20
  0
  MEMA  #-1
  A-MZ  CHKS
  JMP  #+2
  JMP  TEK1
  JMP  @CRLF
  JMS  @UNP
  16
  1
  TEXT %CHECKSUM ER%
  JMP  TEKE
TEK3,  MEMA  WCNTO  /POSSIBLE EXTRA WORD TO ADD
  A-MZ  (6  /IF WCNT IS 6 THE LAST WORD WAS STORED.
  JMP  #+2
  JMP  TEKE
  MPOM  WPNTO
  MEMA  WORDO
  ACCM  @WPNTO
  MPOM  @CNT
TEKE,  JMP  @TEKHEX
/ SCRATCH STORAGE
WPNTO,  0
WORDO,  0
WCNTO,  0
CNT,  0
HEAD,  0
  0  /2 WORD FOR PACKED HEADER/TRAILER
CHKS,  0
NBYTE,  0
/DEFINITIONS
  HEADA,  HEAD-1
BUFS,  100000
/EXTERNALS
UNP,  2650
CRLF,  2736
```

```
// SUBROUTINE NICFIL(NW,FILNAM)
     / REVISION -- JANUARY 26,1980
     / AUTHOR   -- BARRETT,TB
     / PURPOSE  -- TRANSFER A CORE "FILE" TO DISK FILE. THE CORE FILE
     /    STARTS AT 100000.
     / ARGUMENTS --
     /  NW - LENGTH OF THE FILE (WORDS)
     /  FILNAM - ADDRESS OF THE NAME TO BE ASSIGNED TO THE FILE.
     /
     ////////////////////////////////////////////////////////////////
     *4010
     NICFIL,  0
       MEMA  @NICFIL
       ACCM  @OARG2
       ACCM  NW
       ZERM  @OARG1
       MPOM  NICFIL
       MEMA  @NICFIL
       ACCM  AFIL
       MPOM  NICFIL  /SET FOR ERROR RETURN
       JMS   @OPENW
       MEMA  @OARG2  /MAKE SURE HAVE ENOUGH SPACE
       A-MZ  NW
       EXCT  AC19  /IF NEG. JUMP TO ERROR EXIT
       JMP   ERR
       MEMA  @OARG1
       ACCM  IT
       JMS   @WRITE
     IT,  0
     NW,  0
       100000
       MEMA  NW
       ACCM  @OARG2
       MEMA  @AFIL
       ACCM  #+5
       MPOM  AFIL
       MEMA  @AFIL
       ACCM  #+3
       JMS   @CLOSE
       0
       0
       JMP   NIC1
     ERR,  JMS  @UNP
       17
       1
       TEXT % NO FILE SPACE-%
       JMP   @NICFIL
     NIC1,  MPOM  NICFIL
       JMP   @NICFIL
     / SCRATCH STORAGE
     AFIL,  0
```

```
// SUBROUTINE SEARCH(MNEM,CODE,VALUE,NBYTE,TABLA,FLAG)
     / REVISION -- JANUARY 22,1981
     / AUTHOR   -- BARRETT,TB
     / PURPOSE  -- SEARCH A TABLE FOR CODE&VALUE&NO. OF BYTES IN THE
     /    VALUE (WHERE VALUE IS A SINGLE WORD WITH UP TO
     /    5 NIBBLES) OR INSERT VALUE & NO. OF NIBBLES IN VALUE
     / PARAMETERS --
     /  MNEM - 3 LETTER MNEMONIC (PACKED) WHICH IDENTIFIES AN ENTRY
     /  CODE - THE CORRESPONDING CODE (CAMERA "SECONDARY ADDRESS")
     /  VALUE - DATA WORD ASSOCIATED WITH THE CODE (MAY BE 0)
     /    TYPICALLY THIS IS A BCD CODE. FOR EXAMPLE 1203
     /    WOULD HAVE NIBBLES 1,2,0,3,0 IN THAT ORDER FOR
     /    AN OCTAL WORD = 0220060.
     /  NBYTE - NO. OF NIBBLES IN VALUE (MAY BE 0)
     /  TABLA - ADDRESS OF THE START OF THE TABLE
     /  FLAG - ON ENTRY, FLAG IS USED TO INDICATE WHETHER THIS
     /    IS A RETURN (0) OR REPLACE (1) OPERATION. ON
     /    SEARCH OPERATIONS, FLAG IS ALSO RETURNED AS 0 FOR
     /    A SUCCESSFUL SEARCH AND AS 1 FOR NO-FIND.
     /  NOTE ---
     /  TABLE HAS THE FORM -
     /  MNEMONIC (3 PACKED LETTERS,R-JUSTIFIED)
     /  CODE  RIGHT JUSTIFIED 8-BIT CODE
     /  N-NIBBLES NO. OF NIBBLES IN VALUE
     /  VALUE PACKED NIBBLES
     /  ETC.
     /  THE TABLE SHOULD BE TERMINATED WITH @.
     *4070
     SEARCH,  0
       MEMA  @SEARCH
       ACCM  MNEM
       MPOMA  SEARCH
       ACCM  CODE
       MPOMA  SEARCH
       ACCM  VALUE
       MPOMA  SEARCH
       ACCM  NBYTE
       MPOM  SEARCH
       MEMA  @SEARCH
       ACCM  TABLA
       MPOM  SEARCH
       JMP  #+5
     SEA1,  MPOM  TABLA
       MPOM  TABLA
       MPOM  TABLA
       MPOM  TABLA
       MEMA  @TABLA
       A-MZ  (300
       JMP  #+2
       JMP  SEA2  /CANT FIND
```

```
    A-MZ  MNEM
    JMP  SEA1
    MEMZ  @SEARCH
    JMP  SEA3  /STORE
    MPOM  TABLA
    MEMA  @TABLA
    ACCM  @CODE
    MPOM  TABLA
    MEMA  @TABLA
    ACCM  @NBYTE
    MPOM  TABLA
    MEMA  @TABLA
    ACCM  @VALUE
    JMP  SEA4
SEA3,  MPOM  TABLA
    MPOM  TABLA
    MEMA  @NBYTE
    ACCM  @TABLA
    MPOM  TABLA
    MEMA  @VALUE
    ACCM  @TABLA
    JMP  SEA2
SEA4,  ZERMZ  @SEARCH
SEA2,  ONEM  @SEARCH
    MPOM  SEARCH
    JMP  @SEARCH
/ SCRATCH STORAGE
MNEM,  0
CODE,  0
NBYTE,  0
TABLA,  0
```

```
// SUBROUTINE ZERTAB(TABLE)
     / PURPOSE  -- ZEROES THE COMMAND TABLE
     / ARGUMENTS -- TABLE - START ADDRESS OF TABLE
     *4160
     ZERTAB,  0
       MEMA  (23
       ACCM  COUNT
       MEMA  @ZERTAB
       ACCM  POINT
     ZE1,  ZERM  @POINT
       MPOM  POINT
       MMOMZ  COUNT
       JMP  ZE1
       MPOM  ZERTAB
       JMP  @ZERTAB
     / SCRATCH
     COUNT,  0
     POINT,  0
     /
     ////////////////////////////////////////////////////////////////
     /
     // FUNCTION MULTP(X)
     / PURPOSE  -- MULTIPLIES ACC BY X AND RETURNS RESULT IN ACC (LOWER
     /             20 BITS OF THE RESULT. THE HIGH ORDER BITS ARE PUT
     /    IN THE MQ REGISTER.
     MULTP,  0
       TACMQ  /TRANSFER ACC TO MQ REGISTER
       MEMA  @MULTP
       ACCM  #+2
       MULT
       0
       ACCM  VALUE
       TMQAC  /TRANSFERS MQ (LOW ORDER) TO ACC
       ACCM  VALUE1
       MEMA  VALUE
       TACMQ  /PUT HIGH ORDER IN MQ
       MEMA  VALUE1  /LOW ORDER IN ACC FOR RETURN
       MPOM  MULTP  /SET FOR RETURN
       JMP  @MULTP
     /
     ////////////////////////////////////////////////////////////////
     /
     // FUNCTION DIVDE(X)
     / PURPOSE -- DIVIDE MQ+ACC BY X AND RETURN THE RESULT IN ACC.
     /  RETURN THE REMAINDER IN X. NOTE THAT MQ CONTAINS THE HIGH
     /  ORDER BITS AND ACC THE LOW ORDER BITS OF THE DIVIDEND.
     DIVDE,  0
       CLL  /CLEAR LINK
       EXCT  AC19
       STL
```

```
   LASH  1  /LEFT SHIFT DIVIDEND
   ACCM  VALUE  /STORE TEMPORARILY
   TMQAC  /GET HIGH ORDER BITS
   LASH  1
   EXCT  L
   APOA
   ACCM  VALUE1  /STORE TEMPORARILY
   MEMA  VALUE
   TACMQ  /LOAD IT INTO MQ
   MEMA  @DIVDE  /GET DIVISOR
   ACCM  D1
   MEMA  VALUE1  /PUT HIGH ORDER IN ACC
   DIVD
D1,  0
   RISH  1  /RESTORE THE REMAINDER
   ACCM  @DIVDE  /AND STORE FOR RETURN
   TMQAC  /QUOTIENT TO ACC
   MPOM  DIVDE  /FOR RETURN
   JMP  @DIVDE
/
////////////////////////////////////////////////////////////////////////
/
// SUBROUTINE SENDF
/ PURPOSE -- GIVEN A "WORD" TO BE PRINTED IN ACC, SENDF PRINTS THE
/  WORD ACCORDING TO THE FOLLOWING RULE-
/ (1) IF SMODE=0 THEN PRINT DIRECTLY AFTER CONVERTING TO
/  NIC ASCII
/ (2) IF SMODE=1 THEN CONVERT TO DOUBLE HEX AND PRINT. ALSO COUNT
/  CHARACTERS (OR WORDS) AND AT END OF WCNT0 WORDS DO A CR. THE
/  WORD COUNTER COUNTER SHOULD BE SET TO WCNT0 INITIALLY.
/ SMODE,WCNT  AND WCNT0 CAN BE CONSIDERED IN COMMON DECLARED IN SENDF.
SENDF,  0
   MEMZ  SMODE
   JMP  HEXM
   A+MA  (200
   JMS  @TYPE
   JMP  HEXM2
HEXM,  ONEM  LCNT  /SET NIBBLE COUNT TO 2
   MPOM  LCNT
   RLSH  4
HEXM1,  ACCM  VALUE
   MEMA  (260
   ACCM  PREF
   MEMA  VALUE
   ANDA  (17
   ACCM  VALUE1
   A-MA  (12
   EXCT  AC19
   JMP  #+4
```

```
  APOM  VALUE1
  MEMA  (300
  ACCM  PREF
  MEMA  VALUE1
  A+MA  PREF
  JMS  @TYPE
  MEMA  VALUE
  LLSH  4
  MMOMZ  LCNT
  JMP  HEXM1
HEXM2,  MMOMZ  WCNT
  JMP  SENDE
  MEMA  WCNT0
  ACCM  WCNT
  JMS  @CRLF
SENDE,  JMP  @SENDF
/ SCRATCH STORAGE
VALUE,  0
VALUE1,  0
PREF,  0
LCNT,  0
/ COMMON
WCNT,  0
SMODE,  0
WCNT0,  0
/
////////////////////////////////////////////////////////////////////
/
// SUBROUTINE GETFIL
/ PURPOSE -- ASKS FOR FILE WANTED AND GETS THE FILE INTO NIC
/  CORE STARTING AT 100000. IF THE FILE IS UNAVAILABLE,
/    AN ERROR MESSAGE IS PRINTED AND THE
/  ERROR EXIT IS TAKEN. (NORMAL EXIT IS 2 BEYOND CALL POINT.)
GETFIL,  0
  JMS  @UNP
  14
  0
  TEXT % FILE NAME?=%
  JMS  @PKR
  FILNAM
  0
  JMS  @CRLF
  JMS  @OPENR
FILNAM,  BLOCK  2
  MEMA  @OARG1
  SKIP  AC19
  JMP  GETF1
  JMS  @UNP  /FILE DOES NOT EXIST
  24
```

```
  1
  TEXT % FILE DOES NOT EXIST%
  JMP  @GETFIL
GETF1,  MEMA  @OARG1
  ACCM  ITG
  MEMA  @OARG2
  ACCM  SIZE
  JMS  @READD
ITG,  0
SIZE,  0
  MPOM  GETFIL
  JMP  @GETFIL
/
/ SUBROUTINE PAUSE
/ PURPOSE -- VARIABLE PAUSE
PAUSE,  0
  MEMA  @PAUSE
  ACCM  COUNT
  MMOMZ  COUNT
  JMP  #-1
  MPOM  PAUSE
  JMP  @PAUSE
/
/ SUBROUTINE OCT(X)
/ PURPOSE -- PACKS USER GIVEN OCTAL VALUE (UP TO 7 DIGITS) INTO LOCATION
/ X. THE VALUE IS RIGHT JUSTIFIED,ZERO FILL (E.G.1 => 0000001).
OCT,  0
  MEMA  @OCT
  ACCM  VALUE
  ZERM  @VALUE  /ZERO FILL THE NUMBER
OCT1,  JMS  @ECHO  /GET NEXT DIGIT
  A-MZ  (215  /IF CR WE ARE THROUGH
  ZERZ
  JMP  OCTE
  A-MA  (260  /CONVERT FROM ASCII TO OCTAL
```

```
ACCM  VALUE1
      MEMA  @VALUE
      LLSH  3
      A+MA  VALUE1
      ACCM  @VALUE  /STORE NEW VALUE
      JMP  OCT1
    OCTE,  MPOM  OCT
      JMP  @OCT
    /
    / EXTERNALS
    UNP,  2650
    PKR,  3240
    CRLF,  2736
    OPENR,  2432
    READD,  2511
    TYPE,  2731
    OPENW,  2420
    WRITE,  2470
    CLOSE,  2451
    ECHO,  2257
    / DEFINITIONS
    OARG1,  7770
    OARG2,  7771
    /
    ////////////////////////////////////
    /
    $
```

APPENDIX D

The Tektonix "WHEX" File

The source code for the CTL was developed on a Tektronix 8002A uP Laboratory using assembly language and Zilog mnemonics. The program was then assembled and stored as a "load" file. The load file was then loaded into memory and finally stored as a Hexadecimal file using the Tektronix WHEX command. In this form the file can be transferred, for example, to the Sykes Comm-Stor. This appendix shows the structure of the WHEX file and the configuration of Comm-Stor which was used when receiving the WHEX file.

I. Tektronix Hexadecimal File Structure

General Format

(Header)	Location Counter	Byte Count	First Checksum	Data	Second Checksum	CR (EOL)

Format Description

Name	No. of ASCII Characters	Content Description
Header	1	Always a slash (/).
Location Counter	4	Four hexadecimal digits representing the starting memory location of the block.
Byte Count	2	Two-digit hexadecimal value specifying the number of data bytes in the data field of the block.
First Checksum	2	Two-digit hexadecimal number representing the hexadecimal sum of the values of the six digits that make up the location counter and the byte count.
Data	2*N	N data bytes, each represented as two hexadecimal digits. Each hex digit is coded as an ASCII character 0-9 or A-F. There can be a maximum of thirty data bytes (sixty hexadecimal digits) per block.
Second Checksum	2	Two-digit hexadecimal number representing the sum, modulo 256, of the hexadecimal values of the digits that make up the N data bytes.
CR	1	Always a carriage return (CR) indicating the end of the block.

II. Sykes Comm-Stor Configuration for communications with the Tektronix 8002 Development Laboratory. (Comm-Stor should be in the MX mode while receiving a WHEX file.)

```
2:  [illegible] TYPE  (0)
3:  END OF LINE CHARACTER  ([CR])
4:  SYSTEM TO ADD LINE FEED AFTER CARRIAGE RETURN?  (YES)
5:  LINE FEED CHARACTER  ([LF])
6:  CARRIAGE RETURN CHARACTER FOR SYSTEM MESSAGES  ([CR])
7:  LINE FEED CHARACTER FOR SYSTEM MESSAGES  ([LF])
8:  END OF TEXT CHARACTER  ([↑Z])
9:  TRANSMIT "END OF TEXT" CHARACTER?  (YES)
10:  TRANSMIT "END OF TRANSMISSION" CHARACTER?  (NO)
11:  "END OF TRANSMISSION" CHARACTER  ([↑D])
12:  "RESET" CHARACTER  ([↑T])
13:  SPACE CHARACTER FOR SYSTEM MESSAGES  ( )
14:  FIRST CHARACTER OF ESCAPE SEQUENCE  ([↑P])
15:  PARITY ERROR SYMBOL  (?)
16:  "STOP SEND/START SEND" OPTION  (0)
17:  "STOP SEND" CHARACTER (TO COMPUTER)  ([↑S])
18:  "START SEND" CHARACTER (TO COMPUTER)  ([↑Q])
19:  "HOLD" CHARACTER (FROM COMPUTER/TERMINAL)  ([↑S])
20:  "RESUME" CHARACTER (FROM COMPUTER/TERMINAL)  ([↑Q])
21:  OUTPUT NUMERICAL ERROR MESSAGES?  (NO)
22:  SEND ERROR MESSAGES TO MODEM?  (NO)
23:  OUTPUT DELAY CHARACTER #1  ([CR])
     PORTS  (1)
     DELAY FACTOR  (30)
24:  OUTPUT DELAY CHARACTER #2  ([CR])
     PORTS  (NONE)
     DELAY FACTOR  (10)
25:  OUTPUT DELAY CHARACTER #3  ([LF])
     PORTS  (1)
     DELAY FACTOR  (5)
26:  OUTPUT DELAY CHARACTER #4  ([CR])
     PORTS  (1)
     DELAY FACTOR  (5)
27:  "DELETE" CHARACTER ENTERED  ([BS])
28:  "DELETE" CHARACTER ECHOED  ([BS])
29:  "LINE CANCEL" CHARACTER  ([↑X])
30:  NUMBER OF LINES ON PAGE  (24)
31:  STOP DISPLAY AFTER EACH PAGE?  (NO)
32:  SUBSTITUTE TERMINAL PARITY ERRORS WITH ERROR SYMBOL?  (
33:  MODEM OFF-LINE ALERT CHARACTER  ([↑G])
34:  INHIBIT ECHO TO TERMINAL?  (NO)
35:  IGNORE "NULL" CHARACTER FROM TERMINAL?  (YES)
36:  TERMINAL "NULL" CHARACTER  ([NULL])
37:  NORMAL TERMINAL DATA
     PARITY - EVEN/ODD/NONE  (NONE)
     NUMBER OF DATA BITS
     INCLUDING FIXED BITS (IF ANY)  (8)
     EIGHTH DATA BIT  (0)
38:  BINARY TERMINAL DATA
     PARITY - EVEN/ODD/NONE  (NONE)
```

```
41:  HALF DUPLEX MODEM?  (NO)
42:  **DOES NOT APPLY TO FULL-DUPLEX MODEMS**
43:  **DOES NOT APPLY TO FULL-DUPLEX MODEMS**
44:  **DOES NOT APPLY TO FULL-DUPLEX MODEMS**
45:  **DOES NOT APPLY TO FULL-DUPLEX MODEMS**
46:  **DOES NOT APPLY TO FULL-DUPLEX MODEMS**
47:  WAIT AFTER EOL FOR PROMPT OR AFTER ETX FOR ACK?  (NO)
48:  CHARACTER TO INITIATE WAIT PERIOD  (ETX)
49:  WAIT FOR PROMPT BEFORE STARTING TRANSMISSION?  (NO)
50:  "PROMPT" OR "ACKNOWLEDGE" CHARACTER  (^F)
51:  "RETRANSMIT MESSAGE" CHARACTER  (^U)
52:  IGNORE "NULL" CHARACTER FROM MODEM?  (YES)
53:  MODEM "NULL" CHARACTER  (NULL)
54:  IS ATTACHED MESSAGE ECHOED?  (YES)
55:  SUBSTITUTE MODEM PARITY ERRORS WITH ERROR SYMBOL?  (NO)
56:  CHECK FOR MODEM FRAMING ERRORS?  (NO)
57:  INACTIVITY TIMEOUT  (INACTIVE)
58:  NORMAL MODEM DATA
     PARITY - EVEN/ODD/NONE  (EVEN)
     NUMBER OF DATA BITS EXCLUDING PARITY AND
     INCLUDING FIXED BITS (IF ANY)  (7)
59:  BINARY MODEM DATA
     PARITY - EVEN/ODD/NONE  (NONE)
     NUMBER OF DATA BITS
     INCLUDING FIXED BITS (IF ANY)  (8)
60:  DOES MODEM PROVIDE "DATA SET READY"?  (NO)
61:  PRINTER PORT INSTALLED?  (NO)
62:  PRINTER DATA
     PARITY - EVEN/ODD/NONE  (EVEN)
     NUMBER OF DATA BITS EXCLUDING PARITY AND
     INCLUDING FIXED BITS (IF ANY)  (7)
63:  PRINTER INTERFACE CONTROL  (STANDARD)
64:  DOES PRINTER PROVIDE "DATA TERM READY"?  (NO)
65:  DOES PRINTER PROVIDE "SEC. REQUEST TO SEND"?  (NO)
66:  DOES PRINTER REQUIRE "LINE FEED" AFTER "CARRIAGE RETURN"?  (
67:  CHARACTER TO SEPARATE COMMAND AND ARGUMENT  ( )
68:  CHARACTER TO SEPARATE MESSAGE NAMES  (/)
69:  CHARACTER TO START "ENTER AUTOMATIC" INCREMENTING FIELD  (
70:  CHARACTER TO END "ENTER AUTOMATIC" INCREMENTING FIELD  (
71:  CHARACTER TO SEPARATE MESSAGE NAME AND EXTENSION  (+)
72:  CHARACTER TO SEPARATE COMMAND AND ATTACHED MESSAGE  (#)
73:  DIRECTORY BOUNDARY SPECIFICATION CHARACTER  (*)
74:  "DON'T CARE" CHARACTER FOR EXTENSION  (?)
75:  SYNTAX ERROR RESPONSE  (?)
76:  USER RESPONSE TO "SURE?" MESSAGE  (Y)
77:  SYSTEM COMMAND CHARACTER - TERMINAL AND MODEM  (!)
78:  SYSTEM COMMAND CHARACTER - MODEM ONLY  (.)
79:  "SELECT DRIVE 1" CHARACTER  (1)
80:  "SELECT DRIVE 2" CHARACTER  (2)
81:  "BAUD MODEM" COMMAND  (BM)
82:  "BAUD PRINTER" COMMAND  (BT)
83:  "BAUD TERMINAL" COMMAND  (BT)
84:  "COPY" COMMAND  (C)
85:  "CANCEL" COMMAND  (CN)
```

```
"ECHO MODE" COMMAND  (EM)
"ECHO EXIT" COMMAND  (EX)
"INCLUDE MODE" COMMAND  (IM)
"INCLUDE MODE EXIT" COMMAND  (IX)
"LOAD EXTENSION" COMMAND  (LE)
"LOAD INITIAL VALUE" COMMAND  (LI)
"MONITOR MODE" COMMAND  (MM)
"MONITOR MODE EXIT" COMMAND  (MX)
"PRINT" COMMAND  (P)
 "PRINT DIRECTORY" COMMAND  (PD)
 "RECEIVE" COMMAND  (R)
 "RECEIVE AUTOMATIC" COMMAND  (RA)
 "SEND" COMMAND  (S)
 "SEND DIRECTORY" COMMAND  (SD)
 "SEND STATUS" COMMAND  (SS)
 "SEQUENTIAL MODE" COMMAND  (SM)
 "ALPHA MODE" COMMAND  (AM)
 "STANDBY MODE" COMMAND  (SB)
 "WRITE-ENABLE DISK" COMMAND  (WE)
 "WRITE-PROTECT DISK" COMMAND  (WO)
 EDIT OPTION INSTALLED?  (YES)
 EDITOR LINE NUMBER SEPARATOR  (.)
 EDITOR CHARACTER STRING DELIMITER  (/)
 EDIT COMMAND CHARACTER  (:)
 "MOVE FILE" COMMAND  (MV)
 "SAVE FILE" COMMAND  (SV)
 EDITOR "APPEND" COMMAND  (A)
 EDITOR "DELETE" COMMAND  (D)
 EDITOR "INSERT" COMMAND  (I)
 EDITOR "LINE COUNT" COMMAND  (=)
 EDITOR "FILL" COMMAND  (O)
 EDITOR "LIST" COMMAND  (L)
 EDITOR "LIST-NUMBERED" COMMAND  (N)
 EDITOR "REPLACE" COMMAND  (R)
 EDITOR "SEARCH" COMMAND  (T)
 FORMS OPTION INSTALLED?  (YES)
 CHARACTER TO START FORMS VARIABLE FIELD  ([↑A])
 CHARACTER TO END FORMS VARIABLE FIELD  ([↑B])
 "FORMS COMPLETE" COMMAND  (FC)
 "FORMS VARIABLE" COMMAND  (FV)
 "FORMS EXIT" COMMAND  (FX)
 FORMS MODE STRING SEARCH CHARACTER  ([↑Y])
 FORMS MODE-CLEAR FORM CHARACTER  ([↑L])
 FORMS "LINE RE-ENTER" CHARACTER  ([↑Z])
 FORMS MODE - FILL IN FROM DRIVE 1  ([↑O])
 FORMS MODE - FILL IN FROM DRIVE 2  ([↑N])
 FORMS "TAB" CHARACTER  ([TAB])
 ANSWERBACK MESSAGE  (**NONE**)
 CHARACTER TO INITIATE ANSWERBACK MESSAGE  ([↑E])
 USER COMMAND TABLE  ([↑P]R CHR:#COPY*CHR:*CON0[CR])
 SELF-STARTING SYSTEM?  (NO)
```

APPENDIX E

Using the CTL with the Hewlet-Packard 9825A Calculator

As mentioned in the body of this report, the CTL "operating system" is designed to allow the CTL to partake in data transfers only if it is the Controller-in-charge of the bus. It will "wake-up" in this state if the SYC-switch is placed in the on position prior to power-on or reset. However, it is also designed to have control transferred to it from another Controller, provided the SYC-switch is placed in the off position. (It will remain in control until reset.) This transfer of control can be done from an HP 9825A Calculator using the program statement

```
pct 701
```

where 701 is the select code (7) of the 98024A Interface Module and the CTL device address (01). This assume, of course, that the CTL address switches have been set for address 1.

The following annotated listing illustrates 9825A code which can be used to service the Hamamatsu camera while the Calculator is Controller-in-charge (i.e., prior to sending the pct command).

(a) serial poll

```
if bit (7,rds(7))=0 ; goto +0  (wait for SRQ)
rds (702) --> A                (get status byte)
```

(b) set camera "input" format

```
wti 0,7          (specifies that select code 7 is implicit in all succeeding wti operations)
wtb 70202,49     (send ASCII 1 (decimal 49) to camera with secondary address 02)
wti 7,144        (write to register 7, binary 10010000. This causes EOI to be sent with the next data byte.)
wtb 731,13       (send CR (decimal 13) with EOI)
wti 7,128        (write to register 7, binary 10000000. This clears EOI)
```

(c) read camera video data (each pixel consists of 4 ASCII characters - 3 numerals and 1 space)

```
dim  F$[600,4]
for  J=1 to 600; red 702, F$[J]
fmt  f0xc4
dsp  J, F$[J]; wait 50
```

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

Printed by
United States Air Force
Hanscom AFB, Mass. 01731


FILMED
3-8